MODULE 1

CHAPTER 1

Introduction to Web Technology

Syllabus

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 : Fundamental syntax and semantics, 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, Animation, Basics of Bootstrap.

1.1 Client- Server Communication 1-3
1.2 The Internet 1-4
1.2.1 Features of the Internet 1-4
1.3 Basic Internet Protocol 1-5
1.3.1 Features of HTTP protocol 1-5
1.3.2 HTTP flow 1-5
1.3.3 HTTP Messages 1-6
1.4 World Wide Web 1-8
1.4.1 Types of Website 1-8
1.4.2 Static Vs. Dynamic Web Sites 1-9
1.5 Web Clients 1-9
1.6 Web Servers 1-10
1.6.1 Role of web servers in web hosting 1-10
1.6.2 Features of web servers 1-10
1.6.3 How Web servers work? 1-10
1.6.4 Web browser Vs. Web servers 1-11
1.7 HTML5 1-11
1.7.1 Features of HTML5 1-12
1.7.2 Difference between HTML and HTML5 1-12
1.7.3 Syntax of HTML5 Document 1-13
1.7.4 HTML5 Elements 1-13
1.7.5 HTML5 Attributes 1-16
1.7.6 HTML5 Document 1-17
1.8 Formatting Text in HTML 1-18
1.8.1 Formatting 1-18

1.8.2 Fonts 1-19
1.9 Organizing Text in HTML 1-20
1.9.1 Headings 1-20
1.9.2 Paragraph 1-21
1.9.3 Line Breaks 1-21
1.9.4 DIV tag 1-21
1.10 Tables in HTML 1-22
1.10.1 Table Tag with Attributes, TABLE, TR, TH, TD Tags 1-22
1.10.2 Borders, Cell Spacing, Cell Padding, Width, Align, Bgcolor Attribute 1-22
1.10.3 Colspan and Rowspan Attributes 1-23
1.11 HTML Lists 1-25
1.12 HTML Image 1-27
1.13 FORMS in HTML 1-29
1.13.1 Interactive Elements 1-30
1.13.2 HTML <fieldset >and <legend> tag 1-32
1.14 HTML5 controls 1-33
1.14.1 Semantic Elements in HTML5 1-33
1.14.2 Media Elements 1-37
1.14.3 HTML5 New <input> types 1-38
1.15 HTML Drag and Drop 1-38
1.16 Introduction of CSS 1-40
1.16.1 Advantages of CSS 1-40
1.16.2 Inserting CSS in an HTML Page 1-40
1.17 Inheritance 1-41
1.18 Backgrounds 1-42
1.19 Borders 1-43
1.20 Border Images 1-44
1.21 Colors 1-45
1.22 CSS Shadow Effects 1-46
1.22.1 CSS Text Shadow 1-46
1.22.2 CSS box Shadow 1-46
1.23 Text Formatting 1-49
1.24 CSS3 Transformations 1-53
1.24.1 CSS 3D Transforms 1-59
1.25 CSS Transitions 1-61
1.26 CSS Animations 1-64
1.27 Basics of Bootstrap 1-68
1.27.1 Advantages of Bootstrap 1-69
1.27.2 Bootstrap Containers 1-69
1.27.3 Bootstrap Grid System 1-70
1.28 Multiple Choice Questions 1-72
• **Chapter Ends** 1-74

1.1 CLIENT-SERVER COMMUNICATION

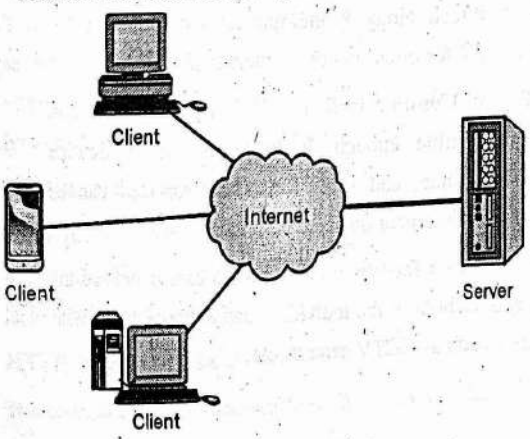
GQ. Define term client and server. Explain client-server communication in detail.

Clients

- When we talk the word Client, it means to talk of a person or an organization using a particular service.
- Similarly in the digital world a Client is a computer (Host) i.e. capable of receiving information or using a particular service from the service providers (Servers).

Servers

- When we talk the word Servers, It means a person or medium that serves something.
- Similarly in this digital world a Server is a remote computer which provides information (data) or access to particular services.
- Client-server architecture works when the client computer sends a resource or process request to the server over the network connection, which is then processed and delivered to the client. A server computer can manage several clients simultaneously, whereas one client can be connected to several servers at a time, each providing a different set of services.
- Clients and servers exchange messages in a request-response messaging pattern. The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol.



(1A1)Fig. 1.1.1 : Client-Server Communication

How the clients interacts with the servers?

- **Step 1:** User enters the URL (Uniform Resource Locator) of the website or file. The Browser then requests the DNS (DOMAIN NAME SYSTEM) Server.
- **Step 2 :** DNS Server lookup for the address of the web server.
- **Step 3 :** DNS Server responds with the IP address of the web server.
- **Step 4 :** Browser sends over an HTTP/HTTPS request to web server's IP (provided by DNS server).
- **Step 5 :** Server sends over the necessary files of the website.
- **Step 6 :** Browser then renders the files and the website is displayed. This rendering is done with the help of DOM (Document Object Model) interpreter, CSS interpreter and JS Engine collectively known as the JIT or (Just in Time) Compilers.

Example

- When a bank customer accesses online banking services with a web browser (the client), the client initiates a request to the bank's web server. The customer's login credentials may be stored in a database, and the web server accesses the database server as a client.
- An application server interprets the returned data by applying the bank's business logic, and provides the output to the web server. Finally, the webserver returns the result to the client web browser for display.

Advantages of Client-Server model

1. Centralized system with all data in a single place.
2. Cost efficient requires less maintenance cost and data recovery is possible.
3. The capacity of the Client and Servers can be changed separately.

Disadvantages of Client-Server model

1. Clients are prone to viruses, Trojans and worms if present in the server or uploaded into the server.
2. Server are prone to Denial of Service (DOS) attacks.
3. Data packets may be spoofed or modified during transmission.
4. Phishing or capturing login credentials or other useful information of the user are common and MITM (Man in the Middle) attacks are common.

1.2 THE INTERNET

GQ. Explain the term Internet.

Definition : The Internet is nothing but a global system which consists of inter-connected networks of computers that uses the Internet protocol suite named TCP/IP (Transmission Control Protocol/Internet Protocol) to link the devices throughout the world. It is a network of networks which contains various types of networks like private, public, government, business, and academic from local to global scope.

- The Internet is a global network of billions of computers and other electronic devices.
- With the Internet, it's possible to access almost any information, communicate with anyone else in the world, and do much more.
- The Internet provides a broad range of information resources and services like the World Wide Web (WWW) applications, telephony, email, sharing of files and connected hypertext documents.
- In 1960, the United States Federal Government form a research commission to create robust, fault-tolerant communication through computer networks.
- The primary form of network called ARPANET was formed in the year of 1980 which provides interconnection between the regional academic and military networks.
- In 1980, the National Science Foundation Network provides funds to develop new networking technologies as well as the integration of various networks.
- In 1990 the commercial networks and enterprises get linked which facilitate the transition to modern internet and creates an effective growth in network technology as different generations of personal, mobile and institutional computers were connected in the network.
- In 1990, the commercial networks and different enterprises are connected which leads to the beginning of transition to advanced modern Internet. This effectively connects the different computers like personal, institutional and mobile in the network.
- Internet reshaped and even redefined the previous traditional communication media like newspaper, radio, television, etc.
- The new Internet technology is adapted by books and newspapers. These communication media are even reshaped in new concepts like blogging, web feeds and online news aggregators. The new forms of personal interactions get accelerated through social networking, instant messaging and internet forums.
- Selling and buying of goods through Internet (online shopping) gives big opportunities to both small businesses as well as major retailers. It also gives a big comfort to consumers.
- Different uses of internet are:
  1. Social networking - chat
  2. Email
  3. Sharing of information
  4. Getting live updates - news around the world
  5. Virtual classrooms
  6. Online Jobs
  7. Remote access

1.2.1 Features of the Internet

1. **Accessibility :** An Internet is a global service and accessible to all. Today, people located in a remote part of an island or interior of Africa can also use Internet.
2. **Easy to Use :** The software, which is used to access the Internet (web browser), is designed very simple; therefore, it can be easily learned and used. It is easy to develop.
3. **Interaction with Other Media :** Internet service has a high degree of interaction with other media. For example, news and other magazine, publishing houses have extended their business with the help of internet services.
4. **Low Cost :** The development and maintenance cost of Internet service are comparatively low.
5. **Extension of Existing IT Technology :** This facilitates the sharing of IT technology by multiple users in organizations and even facilitates other trading partners to use.
6. **Flexibility of Communication :** Communication through internet is flexible enough. It facilitates communication through text, voice, and video too. These services can be availed at both organizational and individual levels.
7. **Security :** Internet facility has to a certain extent helped the security system both at the individual and national level with components such as CCTV camera, etc.

1.3 BASIC INTERNET PROTOCOL

- **Protocol** is a set of rules and regulations to transfer data from one machine to another in network or internet.
- From 1990, the Hypertext Transfer Protocol (HTTP) is the base of data communication process in the WWW (World Wide Web).
- **HTTP protocol** is used for exchanging or delivering the hypertext over the network.
- The HTTP is a communication protocol based on TCP/IP, which transfers the information over the Internet. The information can be in the form of HTML files, images, audios, videos, or query results, etc.
- HTTP can use different ports for communication, but usually, the TCP 80 port is used.
- HTTP specification denotes that how the data to be requested by web client will be formed and delivered to the web server; and after receiving the request for the data at web servers, how the web servers will answer to the requests.
- HTTP is also said to be the request/response protocol which depends on the client-server architecture where the HTTP clients are: Web browsers, and the HTTP Servers are: Web Servers.

1.3.1 Features of HTTP protocol

GQ. Explain in detail HTTP protocol.

1. HTTP is Connectionless Protocol

- In HTTP protocol, after sending the HTTP request by HTTP client (e.g. a Web browser), the connection between HTTP client & HTTP server is disconnected and the HTTP client waits for HTTP server's response.
- Then, after receiving the HTTP client's request the HTTP server processes the request and for sending back the response to the HTTP client, connection is re-established between the HTTP client & the HTTP server.

2. Media independent

- HTTP protocol can handle any type of data over the Internet, if both the HTTP client and the HTTP server know how to handle particular type of data specified by MIME-type.

3. HTTP is Stateless Protocol

- The connectionless characteristic of HTTP makes the HTTP as a stateless protocol.
- This means that the HTTP server and HTTP client are only aware of each other during the processing of the current HTTP request, and that after processing the current HTTP request, both of them forget about each other.
- That's why the HTTP client or the web browser can't hold the information between two different HTTP requests across the web pages.

4. HTTP Protocol is Extensible

- In HTTP 1.0, HTTP header helps in extending the HTTP protocol, where introducing the new functionalities becomes easier.
- The newer functionality is added in HTTP protocol by a simple agreement between the HTTP client and the HTTP server about new header's semantics.

1.3.2 HTTP flow

The following steps are carried out to communicate HTTP client i.e. browser with an HTTP server i.e. an intermediate proxy or the server :

1. Specify URL

- The user needs to specify URL in their browser

2. Open a TCP connection

- Then the browser opens a TCP connection to the server using port 80.
- The TCP connection is used to send one or more request / response.
- The HTTP client can either open a new connection or reuse an existing one, or open number of TCP connections to the servers.

3. Send an HTTP message

- After connection is established the client sends the HTTP request through this connection to the server for getting the appropriate contents.
- Example, following statement is used to request a web page index.html from www.loophole.com.

GET www.loophole.com/index.html HTTP/1.0

4. Read the response sent by the server

- After getting the HTTP response sent by server through the connection, the browser renders the content on the screen according to the data given in HTTP packets and then user can read it easily.

- Example, following statement indicates the response returned by sever for above request.

  HTTP/1.1 200 OK

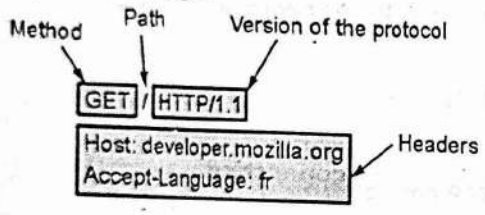
5. Close or reuse the connection for further requests

- The HTTP pipelining helps for sending various HTTP requests one after another through a TCP connection regardless of whether the responses of previously sent requests are received completely or not.
- It is difficult to implement HTTP Pipelining practically so in HTTP 2 this feature is supersede by the more robust concept called as multiplexing requests within a frame.

1.3.3 HTTP Messages

There are two types of HTTP messages, requests and responses.

1. Request Message



(1A2)Fig. 1.3.1

Requests includes following elements :

- An HTTP Request method : (e.g. GET, POST) specifies the operation to be carried out.
- The URI defines complete path of the resource to be fetched.
- The HTTP protocol version being used.
- Optional headers including additional information needed by the HTTP servers or a body needed by some methods like POST.
- The first three elements together referred as **request line**.

Request line

- The Request-line starts with a Request-method followed by the Request-URI followed by the HTTP protocol version. All these parts are separated by a space character.

  e.g. GET index.html HTTP/1.1

- Here, the Request-method is GET, the Request-URI is index.html and the HTTP protocol version is HTTP/1.1.
- The parts of Request-Line are explained below :

(a) Request Method

- The Request-method specifies the operation to be performed on the resource given by Request-URI.
- The Request-method should be written in uppercase letters and it is case-sensitive.
- Following table lists all the Request-methods which are supported by HTTP/1.1.

Sr. No.	Methods	Description
1.	GET	The GET method is used for fetching the information from the server indicated by given request-URI. Requests with GET method only retrieve data from specific server and don't apply any effect on the data.
2.	HEAD	Same as GET, means used for retrieving data from specific server, but it transmit the status-line and the header section only.
3.	POST	Opposite of GET, the POST method is used to send data to the server. The request using POST method leads the uploading of data on the requested server. For example, sending the customer information, files, etc. using HTML forms.
4.	PUT	It is used to replace all the current target resource contents indicated by given request URI with the uploaded content.
5.	DELETE	It is used to deletes all the current target resource contents indicated by given request URI.
6.	CONNECT	It is used to establish a path to the server indicated by given request URI for communication purpose.
7.	OPTIONS	It is used to illustrate various options of communication for the target resource indicated by given request URI.
8.	TRACE	It is used for tracking purpose. It is achieved by performing a message loop back test through the path to the target resource.

(b) Request-URI

- The URI is short form of Uniform Resource Identifier. The Request-URI indicates the requested resource.

- Following statement specifies the most used form of an URI :

  "*" | absoluteURI | abs_path | authority

- Note that the **absoluteURI** is used when the HTTP request is made to a proxy.
- Actually the proxy is requested to forward the service or the request from a valid cache, and then return the response back.
- **For example :** Following statement illustrates the Request-URI which indicates specific resource on an origin server or gateway specified in the path.

GET http://www.Example.com/Web/index.html HTTP/1.1

- Here a client wants to get back a resource (i.e. index.html web page) from the server (server of the site www.Example.com) directly.
- The above statement creates a TCP connection to port 80 of the host "www.Example.com" to forward the following statement :

  GET /Web/index.html HTTP/1.1

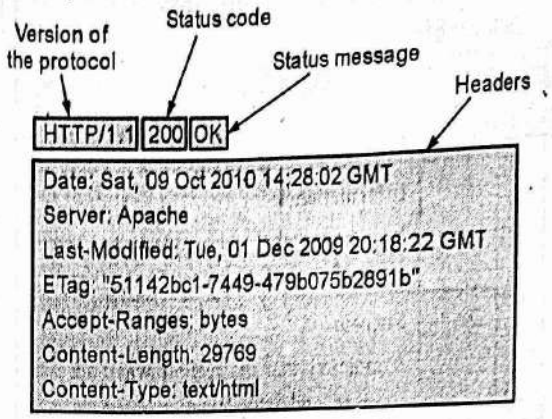
(a) Request Header Fields

The fields of request-header are used to transfer additional information about the request and the client itself, to the server.

Following list contains some important Request-header fields which are used on demand :

- 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

2. Response Message



(1A3)**Fig. 1.3.2**

Responses includes following elements :

- The HTTP protocol version.
- A status code, i.e. success or error code indicating that the request has been successful, or not, respectively. If not it also tells why the error is occurred.
- A status message indicates the short description of the status code.
- HTTP headers similar to the headers for requests.
- Optional, a body containing the fetched resource.
- The first three elements together referred as **status line**

Message Status-Line

- In response, the Status-Line formed by the HTTP protocol version followed by a status code (i.e. a 3-digit number) followed by its associated textual phrase. All the parts are separated by a space character.
- For example, HTTP/1.1 200 OK
- Here the HTTP/1.1 is HTTP-Version, and the Status-Code is 200 and OK is the Reason-Phrase
- The parts of Status-Line are explained below :

(a) HTTP Version

- It indicates which HTTP protocol version is supported by the server.
- The HTTP/1.1 indicates that the server supports HTTP 1.1 version of the protocol.

(b) Status Code

- It is a 3-digit integer value in which only the first digit defines the class of response and the remaining two digits are not used for categorization purpose.
- There are 5 values for the first digit as shown in following table :

Sr. No.	Code	Description
1.	1xx	**Informational** It indicates that the request was received successfully and the process is continuing.
2.	2xx	**Success** It indicates that the request was successfully received, understood, and accepted.
3.	3xx	**Redirection** It indicates that to complete the request further action should be followed.
4.	4xx	**Client Error** It indicates that the request is not successfully completed due to the presence of incorrect syntax.
5.	5xx	**Server Error** It indicates that the server is failed to fulfil a valid request.

- HTTP status-codes can be extended. It is not necessary for HTTP applications to know the meaning of all registered-status-codes.

(c) Response Header Fields

- The fields of response-header are used to send additional information about the response, which can't be put in the Status-Line.
- These response-header fields are used to give information about the server and further accessibility to the resource identified by the Request-URI.
- Following list contains some important response-header fields which are used on demand :

(i) Accept-Ranges (ii) Age
(iii) ETag (iv) Location
(v) Proxy-Authenticate (vi) Retry-After
(vii) Server (viii) Vary
(ix) WWW-Authenticate

1.4 WORLD WIDE WEB

GQ. What is World Wide Web? With the help of suitable diagram discuss the Client-Server Structure of the web.

- WWW stands for **World Wide Web**. The World Wide Web sometimes only pronounced simply as Web, is a way of retrieving information over the Internet. It is model of information-sharing which is built on top of the Internet.
- HTTP protocol is used by the web to transmit data over the internet. Web service which allows applications to communicate with each other uses HTTP. These applications communicate to exchange business logic and use the web to share information.
- A website is made of number of web pages. These web pages which are also known as web documents are interconnected with each other with the help of hyperlinks. Such web pages can be accessed using various web browsers like Internet Explorer, Firefox, Google chrome etc. The web documents can contain simple text, graphics, audio and videos.
- A **website** is made of number of web pages. Simply a website is a file which is accessible anywhere in the world Websites are usually created for commercial purpose.
- There are various extensions of websites depending upon their prime purpose. Some of these are as follows :
  - .com – Commercial
  - .org – Organization
  - .net – Network
  - .biz – Business
  - .edu – Educational
- Also there are extensions depending upon the countries :
  - .in – India
  - .us – America
  - .uk – United Kingdome
  - .nz – New Zealandthrough internet.

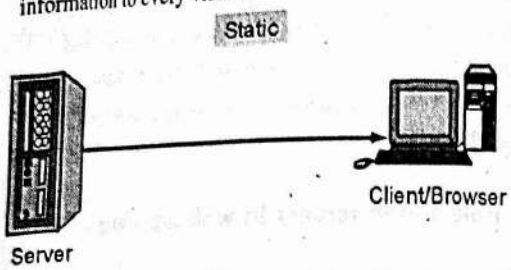
1.4.1 Types of Website

There are two types of website as given below:

1. Static Website
2. Dynamic Website

1. Static website

- Static website is one of the basic types of website that is easy to create.
- In static webpage, codes are fixed for each and every page so the information contained in the page does not change.
- A static website contains web pages with fixed and stable content.
- Its web pages are written in html code and shows the same information to every visitor.



(1A4)Fig. 1.4.1 : Static Website

Advantages of static websites

- Static websites are easier and fast to develop.
- Static websites are low-priced for development.
- Static websites are cheap to host.

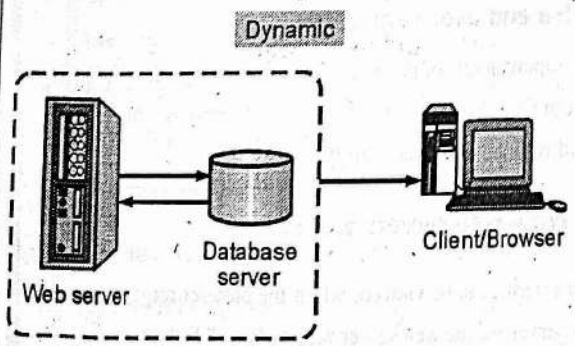
Disadvantages of static websites

- For any changes in website, it requires web development expertise.
- Content of static website are inactive.

2. Dynamic website

Dynamic website is a group of dynamic web pages. The content of dynamic pages can be changed frequently.

Dynamic websites can accesses the required data from the database. Whenever changes are made to the database then the content of the site is also gets changed.



(1A5)Fig. 1.4.2 : Static Website

Advantages of dynamic websites

1. Dynamic websites contains more functionality.
2. Dynamic websites are easier to update.
3. Dynamic websites brings people back to site to search for more information.

Disadvantages of dynamic websites

1. Dynamic websites are slower and more costly to develop.
2. Cost of hosting is more as compared to static website.

1.4.2 Static Vs. Dynamic Web Sites

Sr. No.	Static websites	Dynamic websites
1.	Static website is a set of web pages whose contents are static means it doesn't change dynamically. The contents are changes only when developer updates the web page file and send it to the web server.	Dynamic website is a set of web pages that has dynamically changeable contents called as dynamic web pages.
2.	These websites are easier to develop.	These websites are comparatively difficult to develop than static websites.
3.	It does not need any web programming and database design to create a static website. It simply uses HTML.	It requires web programming and database design to develop. It uses PHP, SERVLET, JSP, ASP.NET etc.
4.	For each request, Static website sends the same response.	According to the requests the responses send by dynamic websites may be different.
5.	To make the website interactive the static website uses hyperlinks and doesn't use any scripting.	To make the website interactive the dynamic website uses client side scripting or server side scripting.
6.	Static websites are flexible.	Dynamic websites are not flexible.

1.5 WEB CLIENTS

- A web client typically refers to the web browser in the user's machine or mobile device.

- A web client is an application that communicates with a web server, using Hypertext Transfer Protocol (HTTP).
- A web client contains two parts: dynamic web pages and the web browser. Dynamic web pages are produced by components that run in the web tier, and a web browser delivers web pages received from the server.
- A web client is also known as a thin client because it does not execute heavy-duty operations such as querying databases, performing complex business tasks, or connecting to legacy applications.
- The web browser asks for data from the website, which happens when the user clicks on a particular link or clicks to download a document that has been displayed on the browser.
- **Web browser** is an example of a web client.

Popular Web Browsers

There are number of web browsers available. They are :

1. Mozilla Firefox
2. Google Chrome
3. Microsoft Internet Explorer
4. Apple Safari
5. Opera etc.

1.6 WEB SERVERS

- A web server is nothing but software and hardware that uses the Hypertext Transfer Protocol, commonly known as HTTP, and some other protocols that respond to request from clients made on the World Wide Web.
- The primary function of a web server is to store, process and deliver web pages to clients.
- Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.
- The web server also supports Simple Mail Transfer Protocol or SMTP and File Transfer Protocol or FTP and HTTP. These are used to transfer files for emailing and even for storage.
- While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.
- Many generic web servers also support server-side scripting using Active Server Pages (ASP), PHP, or other scripting languages. This means that the behaviour of the web server can be scripted in separate files, while the actual server software remains unchanged. Usually, this function is used to generate HTML documents dynamically as opposed to returning static documents.
- The Web server is used for retrieving or modifying information from databases. Web servers are not only used for serving the World Wide Web. They can also be found embedded in devices such as printers, routers, webcams and serving only a local network. The web server may then be used as a part of a system for monitoring or administering the device in question.

1.6.1 Role of web servers in web hosting

The websites hosting means making available the space to place the websites on web servers so that whenever a client wants to access website it is easy to get that website with the help of Internet.

1.6.2 Features of web servers

Following are functionalities which are done by Web servers in website hosting :

1. Stores and secures website data

The website data which is hosted by a server is stored on the place given by server and the server ensures that these data is secure from unauthorized users.

2. Provides web database access

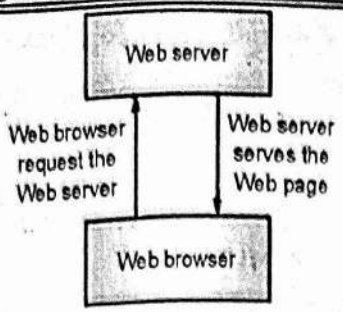
Most of the Web servers provide access of the website which is hosted on it for the user. As well as some web servers are able to facilitate some other services like, backend database servers.

3. Serve the end user requests

The most important role of the web server is to accept client's requests coming through the Internet and process those request and respond back accordingly.

1.6.3 How Web servers work?

A page on internet can be viewed, when the browser requests it from the web server and the web server responds with that page.



(1A6)Fig. 1.6.1: Working of web server

Simple process consists of 4 steps, they are :

1. **Obtaining the IP Address from domain name :** Our web browser first obtains the IP address the domain name (for e.g., for this page the domain name is www.geeksforgeeks.org) resolves to. It can obtain the IP address in 2 ways-
   - By searching in its cache.
   - By requesting one or more DNS (Domain Name System) Servers.

Note : Any website is assigned an IP address when it is first created on web server.

2. **Browser requests the full URL :** After knowing the IP Address, the browser now demands a full URL from the web server.
3. **Web server responds to request :** The web server responds to the browser by sending the desired pages, and in case, the pages do not exist or some other error occurs, it will send the appropriate error message.

   For example :

   You may have seen **Error 404**, while trying to open a webpage, which is the message sent by the server when the page does not exist. Another common one is **Error 401** when access is denied to us due to incorrect credentials, like username or password, provided by us.
4. **Browser displays the web page :** The Browser finally gets the webpages and displays it, or displays the error message.

Popular Web Servers

There are number of web browsers available. They are :

1. Apache HTTP Server
2. Apache Tomcat
3. GlassFish
4. Microsoft-IIS
5. Nginx etc.

1.6.4 Web browser Vs. Web servers

Sr. No.	Parameters	Web Browser	Web Server
1	Purpose	Web Browser is a software which is used to browse and display pages available over internet.	Web server is a software which provides these documents when requested by web browsers.
2	Process	Web Browser is a software which is used to browse and display pages available over internet.	Web server is a software which provides these documents when requested by web browsers.
3	Processing Model	Web browser has no processing model.	Web server follows three major processing models: process based, thread based or hybrid.
4	Data Storage	Web browsers stores user data in cookies in local machine.	Web server provide an area to store the website.
5	Installation	Web Browser is installed on user's machine.	Web server can be installed anywhere but it need to be on a network or on local computer.
6	Example	Google Chrome	Apache Server

1.7 HTML5

HTML5 is a standard for structuring and presenting content on the World Wide Web.

- HTML5 is a markup language used for structuring and presenting content on the World Wide Web.
- It is the fifth and current major version of the HTML standard.
- It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by

humans and consistently understood by computers and devices such as web browsers, parsers, etc.

- HTML5 is intended to include not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.
- It extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.
- Many new syntactic features are included. To natively include and handle multimedia and graphical content, the new <video>, <audio> and <canvas> elements were added.
- It has also support for scalable vector graphics (SVG) content and MathML for mathematical formulas.
- To enrich the semantic content of documents, new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav> and <figure>, are added.
- New attributes are introduced, some elements and attributes have been removed, and others such as <a>, <cite> and <menu> have been changed, redefined or standardized.

1.7.1 Features of HTML5

GQ. Explain features of HTML5.

- **HTML5 Provides Error Handling:** The HTML5 specification offers new rules for lexing and parsing in order to generate the same results by the browsers, when parsing incorrect syntax.
- Audio and Videos are provided in HTML5, e.g. <audio> and <video>.
- Some new elements are introduced in HTML 5 like :
  <article>, < aside>, <audio>, <bdi>, <canvas>, <command>, <data>, <datalist>, <details>, <embed>, <figcaption>, <figure>, <footer>, <header>, <keygen>, <mark>, <meter>, <nav>, <output>, <progress>, <rp>, <rt>, <ruby>, <section>, <source>, <summary>, <time>, <track>, <video>, <wbr>.
- Vector graphics is by default supported by HTML 5. e.g. SVG and canvas
- JS GeoLocation API in HTML5 is used to identify location of user browsing the website (with permission of user).
- HTML5 supports local storage instead of cookies.
- It is easy to draw different shapes such as circle, rectangle and triangle in HTML5.
- All new browsers support HTML5.
- DOCTYPE declaration in Html5 is very simple like, "<!DOCTYPE html>".
- Character encoding in Html5 is very simple like, <meta charset="UTF-8">.
- Handle figure captions more flexibly.
- Show or hide extra information.

1.7.2 Difference between HTML and HTML5

GQ. Differentiate between HTML and HTML5.

Parameter	HTML	HTML5
Audio and Video	Audio and Video tags are not provided in HTML4	Audio and Videos are provided in HTML5 e.g. <audio> and <video>.
Vector Graphics	Vector Graphics is only possible by use of technologies like VML, Silverlight, Flash etc.	Vector graphics is by default supported by HTML5 e.g. SVG and canvas
Tracing User Location	It is difficult to trace true GeoLocation of end user browsing the website particularly when it comes to mobile devices.	JS GeoLocation API in HTML5 is used to identify location of user browsing the website (with permission of user)
Cookies	Html use cookies.	It supports local storage instead of cookies.
Shapes	The different shapes such as circle, rectangle and triangle are not possible.	It is easy to draw different shapes such as circle, rectangle, and triangle.
Browser Support	Supported by old browsers	Supported by all new browsers.
Syntax	Doctype declaration in Html is very long <!DOCTYPE HTML PUBLIC "-//Loophole//DTD HTML 4.01//EN" "http://www.Loophole.com/TR/html4/strict.dtd">	DOCTYPE declaration in Html5 is very simple "<!DOCTYPE html>"
Character Encoding	Character encoding in Html is very long <!DOCTYPE HTML PUBLIC "-// Loophole//DTD HTML 4.0 Transitional//EN">	Character encoding is very simple <meta charset="UTF-8">

1.7.3 Syntax of HTML5 Document

GQ. Explain syntax of HTML5 document.

HTML5 does not have the same syntax rules as XHTML where we need the lower case tag names, quoting attributes, an attribute which has a value and close all the empty elements.

HTML5 provides flexibility and support the followings :

- Uppercase tag names.
- Quotes are optional for attributes.
- Attribute values are optional.
- Closing empty elements are optional.

HTML5 Structure

The DOCTYPE

- In previous versions of HTML, the declaration of DOCTYPEs were longer since the HTML language was SGML based and hence needs a reference to a DTD.
- In HTML 5 simple syntax has been used to specify DOCTYPE :

```
<!DOCTYPE html>
```

- The syntax is case-insensitive.

Character Encoding

- A simple syntax can be used as follows for character encoding :

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

- The syntax is case-insensitive.

The <script> tag

- One of the most common practices that was followed while writing an HTML code was that of adding a type attribute with a value of "text/javascript" to the script element as follows :

```
<script type= "type/javascript"
src="scriptfile.js"> </script>
```

- The extra information is eliminated in HTML5 and the syntax can be as follows :

```
<script src="scriptfile.js"></script>
```

The <link> tag

- The syntax used for link so far was as follows :

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

- The extra information is eliminated in HTML5 and the syntax can be as follows :

```
<link rel ="stylesheet" href="stylefile.css">
```

- The markup for an HTML 5 document would look like the following :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset = "utf-8">
    <title>...</title>
  </head>
  <body>
    <header>...</header>
    <nav>...</nav>
      <article>
    <section>
      ...
    </section>
    </article>
    <aside>...</aside>
      <footer>...</footer>
  </body>
</html>
```

1.7.4 HTML5 Elements

- HTML5 elements are basically marked up with the help of start tags and end tags.
- Tags are delimited with the help of angle brackets with the enclosed tag name.
- The dissimilarity between start tags and end tags is that the end tag includes a slash prior to tag name.
- Example of HTML5 element : ...
- HTML5 tag names are not case sensitive and sometimes written in uppercase or mixed case, even though the most preferable convention is to write with lower case.
- Most of the elements contain some contents in open and close tags.
- Some elements, however, do not have any content at all and these are called as void elements.
- For example, br, hr, link and meta etc.

List of HTML5 Elements

Tag	Description
<!--...-->	States a comment
<!DOCTYPE>	States the document type
<a>	States an anchor
<abbr>	States an abbreviation
<acronym>	Deprecated:States an acronym
<address>	States an address element
<applet>	Deprecated: States an applet
<area>	States an area inside an image map
<article>	New Tag: States an independent piece of content of a document, such as a blog entry or newspaper article
<aside>	New Tag:States a piece of content that is only slightly related to the rest of the page.
<audio>	New Tag:States an audio file.
<base>	States a base URL for all the links in a page
<basefont>	Deprecated: States a base font
<bdo>	States the direction of text display
<bgsound>	States the background music
<blink>	States a text which blinks
<blockquote>	States a long quotation
<body>	States the body element
 	Inserts a single line break
<button>	States a push button
<canvas>	New Tag:This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<caption>	States a table caption
	Deprecated: States centered text
<col>	States attributes for table columns
<colgroup>	States groups of table columns
<command>	New Tag:States a command the user can invoke.
<comment>	Puts a comment in the document
<datalist>	New Tag:Together with the a new list attribute for input can be used to make comboboxes

Tag	Description
<dd>	States a definition description
<del>	States deleted text
<details>	New Tag:States additional information or controls which the user can obtain on demand.
<dir>	Deprecated: States a directory list
	States a section in a document
<dl>	States a definition list
<dt>	States a definition term
<embed>	New Tag:Defines external interactive content or plugin.
<fieldset>	States a fieldset
<figure>	New Tag:States a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.
<b>	States bold text
<big>	Deprecated:States big text
<i>	States italic text
<small>	States small text
<tt>	Deprecated:States teletype text
<font>	Deprecated: States text font, size, and color
<footer>	New Tag:States a footer for a section and can contain information about the author, copyright information, et cetera.
<form>	States a form
<frame>	Deprecated:States a sub window (a frame)
<frameset>	Deprecated:States a set of frames
<head>	States information about the document
<header>	New Tag:States a group of introductory or navigational aids.
<hgroup>	New Tag:States the header of a section.
<h1> to <h6>	States header 1 to header 6
<hr>	States a horizontal rule
<html>	States an html document
<isindex>	Deprecated: States a single-line input field
<iframe>	States an inline sub window (frame)
<ilayer>	States an inline layer

Tag	Description
<img>	States an image
<input>	States an input field
<ins>	States inserted text
<keygen>	New Tag:States control for key pair generation.
<keygen>	Generate key information in a form
<label>	States a label for a form control
<layer>	States a layer
<legend>	States a title in a fieldset
<li>	States a list item
<link>	States a resource reference
<map>	States an image map
<mark>	New Tag:States a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
<marquee>	Create a scrolling-text marquee
<menu>	Deprecated: States a menu list
<meta>	States meta information
<meter>	New Tag:States a measurement, such as disk usage.
<multicol>	States a multicolumn text flow
<nav>	New Tag:States a section of the document intended for navigation.
<nobr>	No breaks allowed in the enclosed text
<noembed>	States content to be presented by browsers that do not support the <embed>tag
<noframes>	Deprecated:States a noframe section
<noscript>	States a noscript section
<object>	States an embedded object
<ol>	States an ordered list
<optgroup>	States an option group
<option>	States an option in a drop-down list
<output>	New Tag:States some type of output, such as from a calculation done through scripting.
<p>	States a paragraph
<param>	States a parameter for an object

Tag	Description
<cite>	States a citation
<code>	States computer code text
<dfn>	States a definition term
	States emphasized text
<kbd>	States keyboard text
<samp>	States sample computer code
	States strong text
<var>	States a variable
<plaintext>	Deprecated: Render the raminder of the document as preformatted plain text
<pre>	States preformatted text
<progress>	New Tag:States a completion of a task, such as downloading or when performing a series of expensive operations.
<q>	States a short quotation
<ruby>	New Tag:Together with <rt> and <rp> allow for marking up ruby annotations.
<script>	States a script
<section>	New Tag:Represents a generic document or application section.
<select>	States a selectable list
<spacer>	States a white space
	States a section in a document
<s>	Deprecated: States strikethrough text
<strike>	Deprecated: States strikethrough text
<style>	States a style definition
<sub>	States subscripted text
<sup>	States superscripted text
<table>	States a table
<tbody>	States a table body
<td>	States a table cell
<textarea>	States a text area
<tfoot>	States a table footer
<th>	States a table header
<thead>	States a table header

Tag	Description
<time>	New Tag:States a date and/or time.
<title>	States the document title
<tr>	States a table row
<u>	Deprecated: States underlined text
<ul>	States an unordered list
<video>	New Tag:States a video file.
<wbr>	New Tag:States a line break opportunity.
<wbr>	Indicate a potential word break point within a <nobr> section
<xmp>	Deprecated: States preformatted text

1.7.5 HTML5 Attributes

GQ. List HTML5 attributes.

- Attributes are provided to elements to set their various properties.
- Attributes are globally or specifically defined. The global attributes can be used for any element while some attributes are for specific elements only.
- All attributes have a name and a value pair.

Example

```
<div class="example">...</div>
```

- Attributes are specified in start tags and never used in end tags.
- Sometimes written in uppercase or mixed case, even though the most preferable convention is to write with lower case.

List of HTML 5 Attributes

Attribute	Options	Function
accesskey	User Defined	Used to state keyboard shortcut to access an element.
align	right, left, center	Horizontally aligns tags
background	URL	Sets background image behind an element
bgcolor	numeric, hexidecimal, RGB values	Sets a background color behind an element
class	User Defined	Used to Classify an element so as to use with CSS.
contenteditable	true, false	Used to state if the user can update the element's content or not.
contextmenu	Menu id	Used to state the context menu for an element.
draggable	true,false, auto	Used to state whether or not a user can drag an element.
height	Numeric Value	Used to state the height of tables, images, or table cells.
hidden	hidden	Used to state whether element should be visible or not.
id	User Defined	Names an element for use with CSS.
item	List of elements	Group elements.
itemprop	List of items	Group items.
spellcheck	true, false	Used to state if the element must have its spelling or grammar checked.
style	CSS Style sheet	Used to state an inline style for an element.
subject	User define id	Used to state the element's corresponding item.
tabindex	Tab number	Used to state the tab order of an element.
title	User Defined	"Pop-up" title for the elements.
valign	top, middle, bottom	Vertically aligns tags in an HTML element.
width	Numeric Value	Used to state the width of tables, images, or table cells.

1.7.6 HTML5 Document

HTML5 documents structure can contain following tags :

- **Section** - A generic document or application section is represented by this tag. It can be used with <h1> - <h6>.
- **Article** - This tag indicates a separate piece of content of a document, for example a blog entry or newspaper article.
- **aside** - This tag indicates a separate piece of content that is only partly related to the rest of the page.
- **Header** - This tag indicates the header of a section.
- **Footer** - This tag indicates a footer for a section and can information such as details of author, copyright information, etc.
- **Nav** - This tag indicates a section of the document proposed for navigation.
- **Dialog** - A conversation can be marked up by this tag.
- **Figure** - This tag is helpful to link a caption with some embedded content, like a graphic or video.

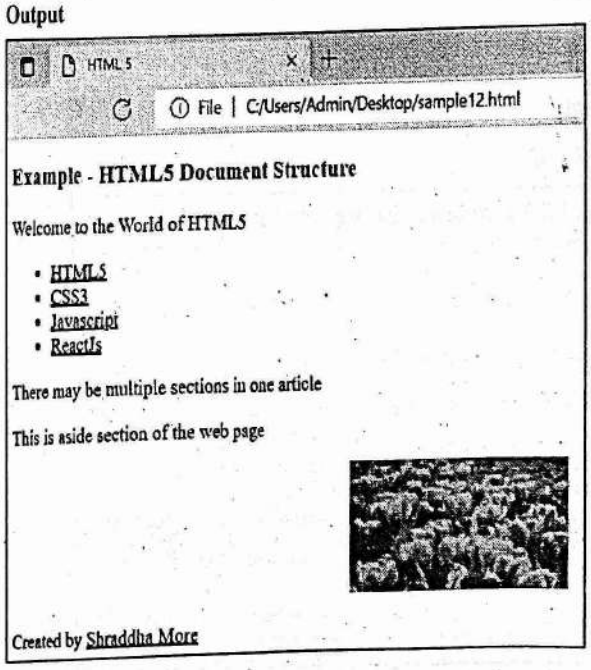
Structure of HTML5 Document

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Title Here</title>
  </head>
      <body>
    <header>Header Content</header>
    <nav>nav Content </nav>
    <article>
      <section>
            Content
      </section>
    </article>
    <aside> Content </aside>
    <figure> Content </figure>
    <footer> Content </footer>
  </body>
</html>
```

Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>HTML 5</title>
  </head>
  <body>
    <header role="banner">
     <h3>Example - HTML5 Document Structure</h3>
     <p>Welcome to the World of HTML5</p>
    </header>
    <nav>
     <ul>
      <li><a href="#">HTML5</a></li>
          <li><a href="#">CSS3</a></li>
      <li><a href="#">Javascript</a></li>
      <li><a href="#">ReactJs</a></li>
     </ul>
    </nav>
    <article>
     <section>
<p>There may be multiple sections in one article</p>
     </section>
    </article>
    <aside>
     <p>This is aside section of the web page</p>
    </aside>
    <figure align="center">
<img src="tulip.jpg" alt="Tulip" width="200"
height="100">
    </figure>
    <footer>
<p>Created by <a href="#">Shraddha More</a></p>
    </footer>
  </body>
</html>
```

Output



1.8 FORMATTING TEXT IN HTML

1.8.1 Formatting

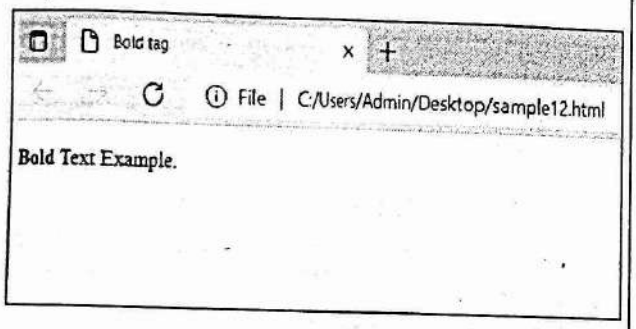
1. Bold Tag

- HTML provides tag to bold the text.
- The text surrounded by ... tags, is displayed on webpage in bold.

Program 1.8.1 : Write a simple code to demonstrate the use of <bold> tag.

```
<!DOCTYPE html>
<html>
<head>
<title>Bold tag</title>
</head>
<body>
<p> <b> Bold</b> Text Example.</p>
</body>
</html>
```

Output



2. Italic

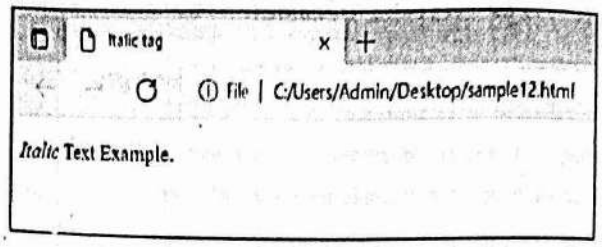
HTML provides tag to italic the text.

The text surrounded by ... tags, is displayed on webpage in italic form.

Program 1.8.2 : Write a script using italic tag.

```
<!DOCTYPE html>
<html>
<head>
<title> Italic tag </title>
</head>
<body>
<p><i> Italic</i> Text Example.</p>
</body>
</html>
```

Output



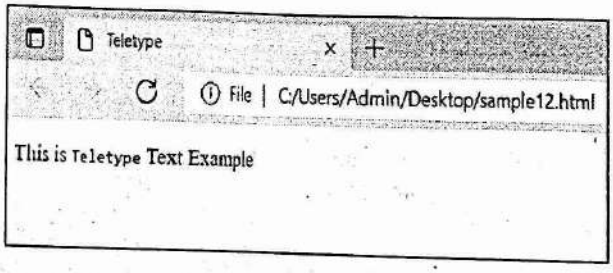
3. Teletype

- The HTML <tt> tag specifies teletype text. This is not supported in HTML5.
- The text surrounded by <tt> and </tt> tag is displayed in moonscape font.
- In a moonscape font each letter has the same width.

Program 1.8.3 : Write a code to display the teletype text.

```
<!DOCTYPE html>
<html>
<head>
<title>Teletype</title>
</head>
<body>
    <p>This is <tt> Teletype</tt> Text Example</p>
</body>
</html>
```

Output



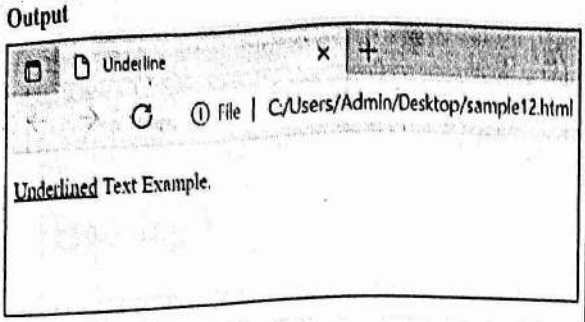
4. Underline

- HTML provides <u> tag to underline the text.
- The text surrounded by <u>...</u> tags, is displayed on webpage in underline.

Program 1.8.4 : Write a program to show the underlined text.

```
<!DOCTYPE html>
<html>
<head>
<title>Underline</title>
</head>
<body>
<p><u> Underlined</u> Text Example. </p>
</body>
</html>
```

Output



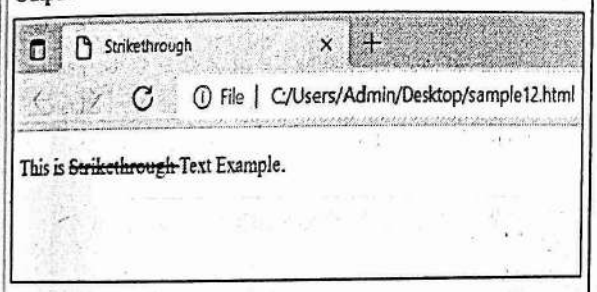
5. Strikethrough

- HTML provides <strike> tag to strikethrough the text.
- The text surrounded by **< strike >...</ strike >** tags, is displayed on webpage with strikethrough means a thin line through the text.

Program 1.8.5 : Write a code to display the strikethrough text.

```
<!DOCTYPE html>
<html>
<head>
<title> Strikethrough </title>
</head>
<body>
<p> This is <strike> Strikethrough </strike> Text
Example. </p>
</body>
</html>
```

Output



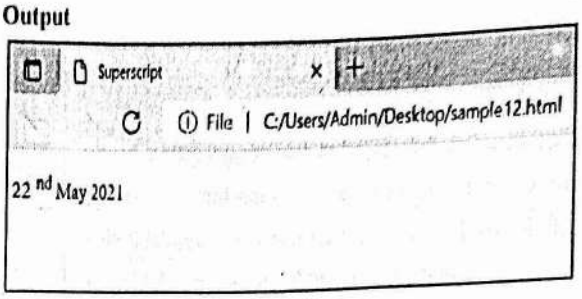
6. Superscripts

- HTML provides <sup> tag to strikethrough the text.
- The text surrounded by **<sup >...</sup >** tags, is displayed on webpage in superscript. It appears half a character above the other characters. Superscript text can be used for text, like 22nd May 2021.

Program 1.8.6 : Write a program to show the use of <sup> tag.

```
<!DOCTYPE html>
<html>
<head>
<title> Superscript </title>
</head>
<body>
<p> 22 <sup> nd, </sup> May 2021 </p>
</body>
</html>
```

Output



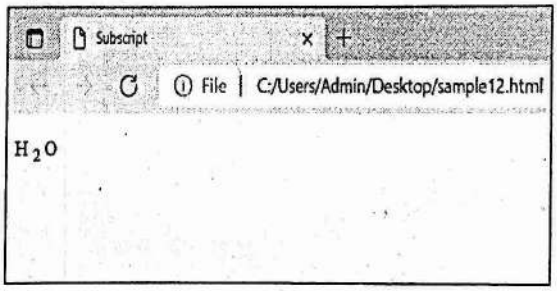
7. Subscript

- The text surrounded by **< sub>...</sub>** tag is written in subscript.
- It appears half a character below the baseline. Subscript text can be used for chemical formulas, like H_2O.

Program 1.8.7 : Write a program to display the subscript text.

```
<!DOCTYPE html>
<html>
<head>
<title> Subscript </title>
</head>
<body>
<p> H <sub> 2 </sub> O </p>
</body>
</html>
```

Output



1.8.2 Fonts

The text in a webpage can be formatted by setting the < font > ..... < /font > tag and various font attributes.

The <font> tag has following attributes :

1. Face

The Face attribute is used to specify the name of font for the text.

```
<font face="Arial"> Arial Font </font>
```

- The specified font must be installed on the machine where the web page is running; otherwise the text will be displayed in default font of the web browser.
- To the face attribute, we can give multiple font names separated by a comma.

```
<font face="Sans serif, Comic Sans MS, Lucida Console">
Multiple fonts</font>
```

- The first specified font is taken into consideration by the browser. But if it is not available on the machine then second font will be applied and if second font is not available then third font will be applied. Even this is also not available then the default font is applied to the text.

2. Size

- The size of a font can be set using the size attribute. The allowed range of values for font size is from 1(smallest) to 7(largest). 3 is the default font size.

```
<font size=5>Size is 5</font>
```

- It is also possible to set the relative font size. That means how many sizes greater or smaller than the current font size.

```
<font size="-2">Less by 2</font>
<font size="+2">Greater by 2</font>
```

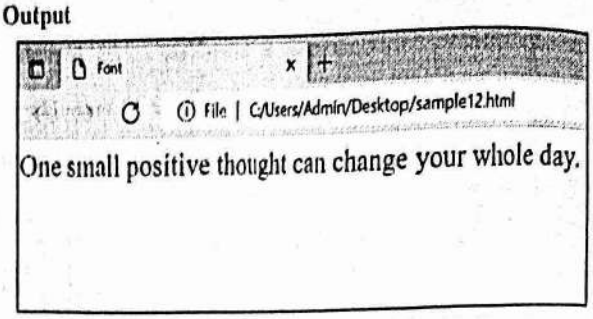
3. Color

- Used to set the color of font. This attribute can accept value as standard name of a color or color code.

Program 1.8.8 : Write a program to display font size, font face and color.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Font </title>
  </head>
  <body>
<font face = "Times New Roman,Sans serif" size = "5"
color="red">
  One small positive thought can change your whole day.
</font>
  </body>
</html>
```

Output



1.9 ORGANIZING TEXT IN HTML

HTML provides various tags to organize text on the web page.

1.9.1 Headings

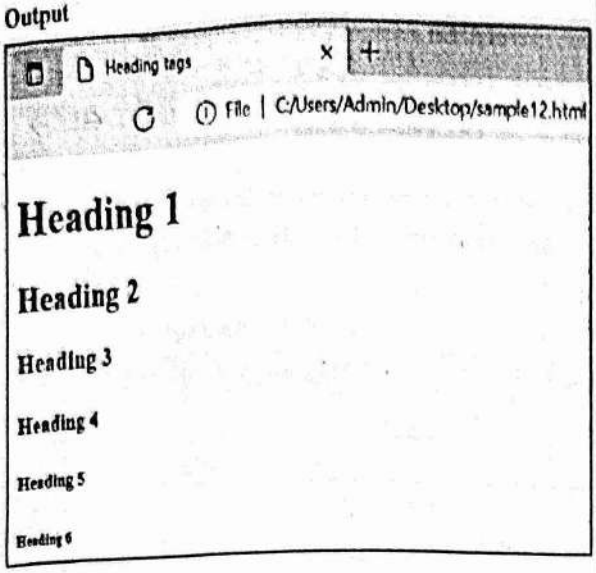
GQ. Explain heading element in HTML.

- In general a document starts with heading. The heading elements are used to give title or subtitles to be displayed on the webpage. The different headings may require different sizes as per their importance.
- HTML has six levels of headings. Six elements namely <h1>, <h2>, <h3>, <h4>, <h5>, and <h6> are used to give headings. When any of the heading tag is used, one line before and one line after that heading is added by the browser.
- <h1> is the biggest heading tag while <h6> is the smallest heading tag. Hence h1 should be used for the most important heading text and h6 should be used for least important heading text.

Program 1.9.1 : Write a program to display all heading tags from <h1> to <h6>.

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

Output



1.9.2 Paragraph

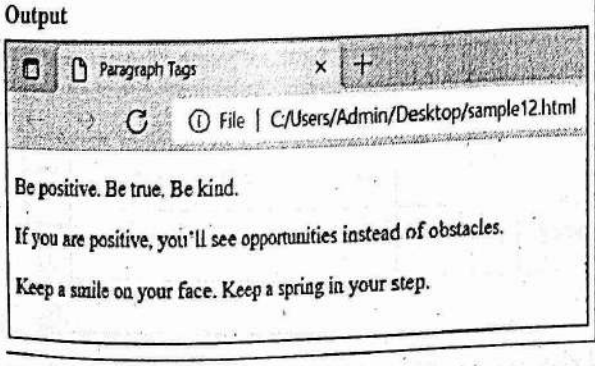
GQ. Explain paragraph element in HTML.

- The tag is used to define paragraph in HTML. This tag structures our text in different paragraphs.
- The paragraph of text is enclosed in opening and closing tags.

Program 1.9.2 : Write a simple program to define the paragraph tag.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Paragraph Tags</title>
  </head>
  <body>
    <p>Be positive. Be true. Be kind.</p>
    <p>If you are positive, you'll see opportunities instead of
obstacles.</p>
    <p>Keep a smile on your face. Keep a spring in your
step.</p>
  </body>
</html>
```

Output



1.9.3 Line Breaks

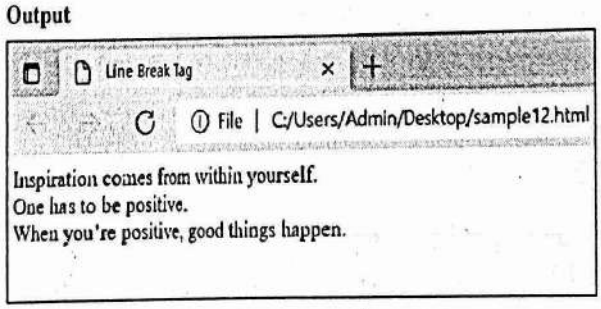
GQ. Explain line break element in HTML.

- The <br> tag is used to give the line break. It works just like '\n' of C programming language.
- This tag is called as empty element as it does not need any closing tag.

Program 1.9.3 : Write a simple code for line break example.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Line Break Tag</title>
  </head>
  <body>
  Inspiration comes from within yourself.
<br>
One has to be positive.
<br>
When you're positive, good things happen.
  </body>
</html>
```

Output



1.9.4 DIV tag

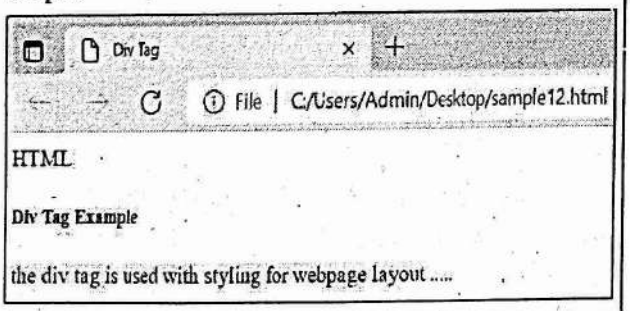
- HTML provides the tag to render the division or section in an HTML document.
- The tag is a block level tag which is used to group block-elements to format them with styles.
- tag is used to create webpage layout where we define different parts (Left, Right, Top etc.) of the page and apply styling to each so they will appear with special style rules.
- The tag supports the following standard attributes.

Attribute	Value	Description
class	Class_name	Class name for an HTML element
dir	rtl ltr	Indicates the text direction for the text included within HTML element
id	id	Gives a unique id for a HTML element
lang	language_code	Indicates a language code for the text in a HTML element
style	style_definition	Indicates an inline style for a HTML element and the text included in it.
title	text	Gives more information about a HTML element
xml:lang	language_code	Indicates a language code for the text in an element, in XHTML documents

Program 1.9.4 : Write a program to display the use of tag.

```
<!DOCTYPE html>
<html>
<head>
<title>Div Tag </title>
</head>
<body>
<div id="title" align="left" > HTML </div>
<div id="content" dir="ltr" >
<h5> Div Tag Example </h5>
<p>the div tag is used with styling for webpage layout
.....</p>
</div>
</body>
</html>
```

Output



1.10 TABLES IN HTML

- HTML provides the ability to developers to add the data in table format so that it will help the visitors to understand the information quickly.
- HTML provides the <table> tag for adding the table into webpage. Let's see some information about this tag.

1.10.1 Table Tag with Attributes, TABLE, TR, TH, TD Tags

The <table> tag is used to present the data in tabular format, means in rows and columns. <table> is the main tag which is used to create the table.

Following Table shows the list of table related tags

Tag	Description
<table>	It defines a table.
<tr>	Table row - It defines a row in a table.
<th>	Table heading - It defines column headings in the table.
<td>	Table Data - It defines a cell in a table.
<caption>	It defines the caption of the table.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used in combination with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

1.10.2 Borders, Cell Spacing, Cell Padding, Width, Align, Bgcolor Attribute

Also some other attributes are provided to format the table such as specifying the color, border, height, width, etc.

Attributes of <table> tag

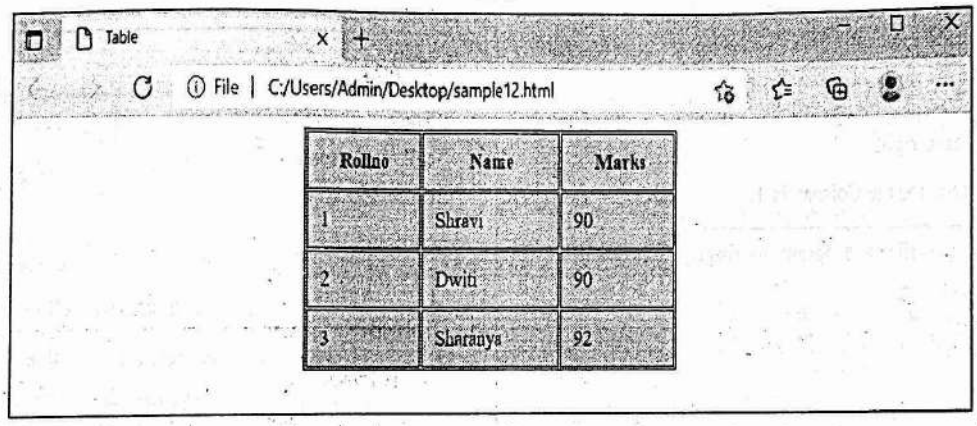
Attribute	Value	Description
align	left center right	Specifies the alignment of a table with respect to surrounding text
bgcolor	rgb(x,x,x) #xxxxxx colorname	Specifies the color for a table background
border	Any number	Specifies size of border
cellpadding	pixels	Specifies the space between the cell wall and the cell content
cellspacing	pixels	Specifies the space between cells

Attribute	Value	Description
width	pixels %	Specifies the width of a table
height	pixels %	Specifies the height of a table
Bordercolor	rgb(x,x,x) #xxxxxx colorname	Sets color of the table border

Program 1.10.1 :Write HTML code to display student data in table format.

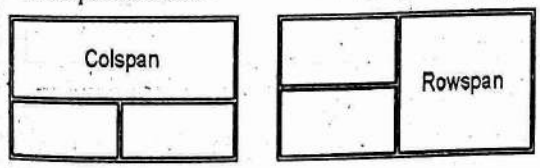
```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
      <body>
<table border=1 bordercolor="red" width=40% height =
40% cellspacing=2 cellpadding=10  bgcolor="lightgray"
align="center">
<tr>
<th>Rollno</th>
<th>Name</th>
<th>Marks</th>
</tr>
<tr>
<td>1</td>
<td>Shravi</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>Dwiti</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>Sharanya</td>
<td>92</td>
</tr>
</table>
</body>
</html>
```

Output



1.10.3 Colspan and Rowspan Attributes

- A table is divided into rows and every row is divided into cells.
- Sometimes we require the table cells span across (merged) more than one column or row. Here we can use Colspan and Rowspan attributes.



(1A7)

(i) Colspan

- The colspan attribute is used to define the number of columns a cell should span (or merge) horizontally. That means, we like to merge two or more Cells in a row into a single Cell.

```
<td colspan=2 >
```

- This given code will merge two Cells into one Cell horizontally.

Before Colspan

First Cell	Second Cell
Third Cell	Forth Cell

After Colspan

Merged	
Third Cell	Forth Cell

- In the above figure there are two tables. In the first figure 2 rows and 2 columns in each row are present.
- In the second figure, first two cells are merged horizontally with the help of Colspan attribute.

Program 1.10.2 : Write a program to demonstrate use of colspan.

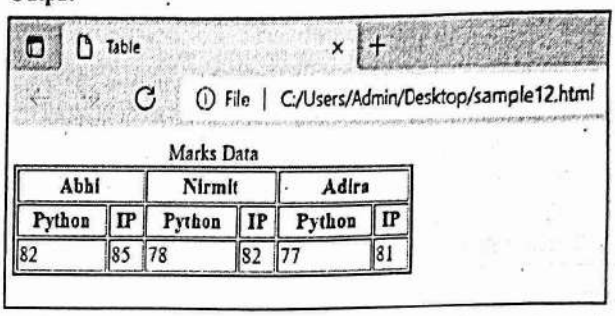
```
html>
<body >
    <table border=1 >
        <tr>
            <td colspan=2 >
                Merged
            </td>
        </tr>
        <tr>
            <td>
                Third Cell
            </td>
            <td>
                Forth Cell
            </td>
        </tr>
    </table>
</body>
</html>
```

- Colspan (Column Span) merges the Cells horizontally that means from left to right.
- The default value for the Colspan is 1.

Program 1.10.3 : Write a program to display the marks of students using colspan.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
  <body>
<table border=2 width="40%">
<caption>Marks Data</caption>
<tr>
<th colspan="2">Abhi</th>
<th colspan="2">Nirmit</th>
<th colspan="2">Adira</th>
</tr>
<tr>
<th>Python</th>
<th>IP</th>
<th>Python</th>
<th>IP</th>
<th>Python</th>
<th>IP</th>
</tr>
<tr>
<td>82</td>
<td>85</td>
<td>78</td>
<td>82</td>
<td>77</td>
<td>81</td>
</tr>
</table>
</body>
</html>
```

Output

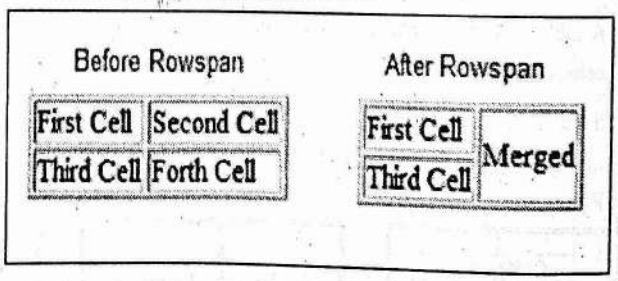


(ii) Rowspan

- The rowspan attribute is used to define the number of rows a cell should span in vertical manner. That means two or more Cells in the same column are merged as a single Cell vertically.

```
<td rowspan=2 >
```

- This given code will merge two Cells into one Cell vertically.
- In the above figure, we can see there are two tables. In the first figure there are 2 rows and each row has 2 columns.
- In the second figure there are 2 rows in the first column and only 1 row in the second column. The Rowspan attribute is used vertically in the second column.



Program 1.10.4 : Write a program to demonstrate use of rowspan.

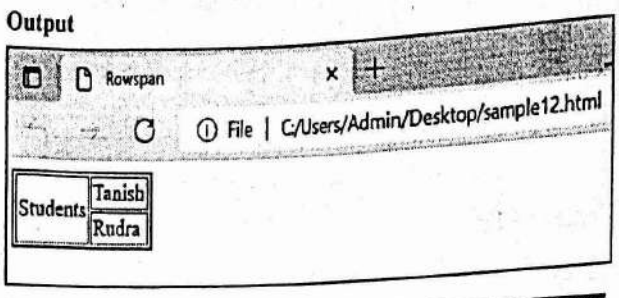
```
<html>
<body >
    <table border=1 >
        <tr>
            <td>
                First Cell
            </td>
            <td rowspan=2 >
                Merged
            </td>
        </tr>
        <tr>
            <td valign=middle>
                Third Cell
            </td>
        </tr>
    </table>
</body>
</html>
```

- Rowspan merged the Cells in vertical manner that means from top to bottom.

Program 1.10.5 : Write a program to merge the data in a row.

```
<html>
<head>
<title>
Rowspan
</title>
</head>
<body>
<table border=2>
  <tr>
    <td rowspan="2">
    Students
    </td>
    <td>
  Tanish
    </td>
  </tr>
  <tr>
    <td>
    Rudra
    </td>
  </tr>
</table>
</body>
</html>
```

Output



1.11 HTML LISTS

- Lists are used everywhere on the websites.
- There are number of things like articles, website navigation menus, and product features on e-commerce websites which makes frequent use of lists.
- HTML provides three types ways to give the lists :
  1. An unordered list <ul>
  2. An ordered list <ol>
  3. A definition list <dl>

A. HTML Unordered Lists

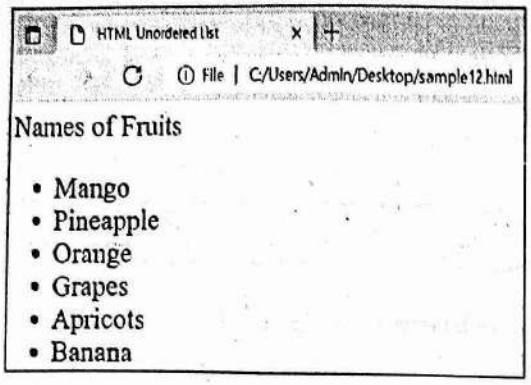
- In this list, plain bullets are used for the list items. Generally this option is used when there is no any standard sequence or order of the list items. The <ul> tag is used to give this list. Every element in the list is marked by a bullet. Different types of bullets are available.
- The <li> which is the sub-tag of <ul> is used for individual list elements.

Program 1.11.1 : Write a code for displaying the list elements.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Unordered List</title>
  </head>
  <body>
<font size=5>
Names of Fruits
    <ul>
      <li>Mango</li>
      <li>Pineapple</li>
      <li>Orange</li>
      <li> Grapes </li>
      <li> Apricots</li>
      <li> Banana</li>
    </ul>
</font>
```

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

Output



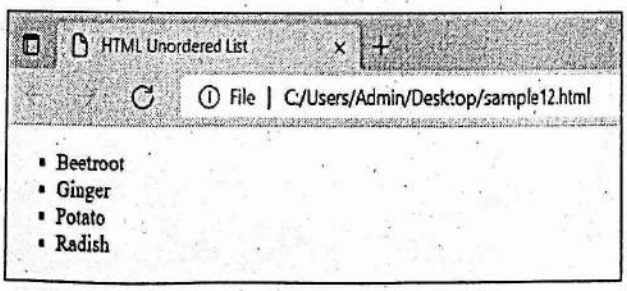
- The <ul> tag has attribute *TYPE* which is used to specify the type of bullet for the list items. The default type is disc which is displayed in above output.
- There following options for the attribute *TYPE* of <ul> tag.

  (1) Square (2) Circle (3) Disc
- See the following Program 1.11.2 which illustrates the use of attribute *TYPE*.

Program 1.11.2 : Write a simple code demonstrating the use of <ul> tag.

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

Output



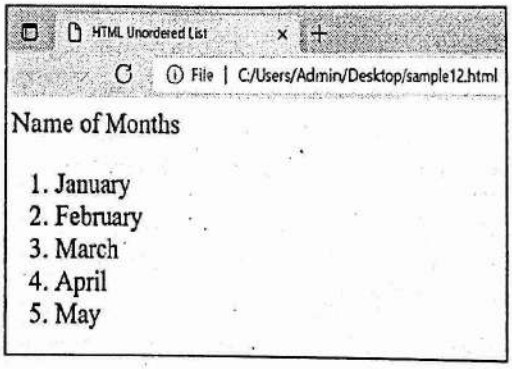
B. HTML Ordered Lists

- Ordered lists are generally used when we want to specify numbers instead of bullets. The <ol> tag is used for such list.
- The number starts from 1 and also incremented by one for each successive list element in the ordered list.

Program 1.11.3 : Write a script to display ordered list.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Unordered List</title>
  </head>
  <body>
<font size=5>
Name of Months
    <ol>
      <li>January</li>
      <li>February</li>
      <li>March</li>
      <li>April</li>
      <li>May</li>
    </ol>
</font>
  </body>
</html>
```

Output



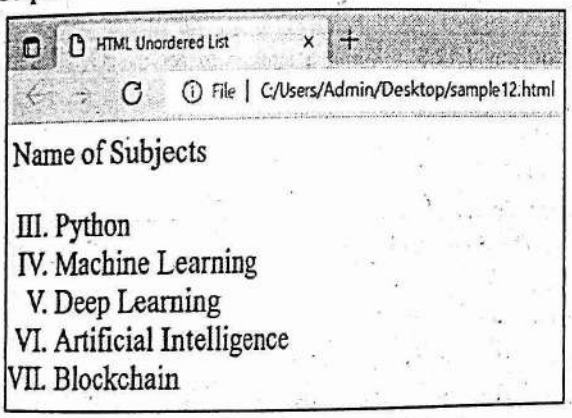
- The <ol> tag has attribute *TYPE* which is used to specify the type of number for the list items. The default type is 1 which is displayed in above output.
- There following options for the attribute *TYPE* of <ul> tag.
  - 1 - Number
  - I – Upper Roman
  - i – Lower Roman
  - A – Upper Alpha
  - a – Lower Alpha

- Following Program 1.11.4 illustrates the use of attribute *TYPE*.
- **The start Attribute :** This attribute of <ol> tag is used to specify the starting point of numbering.

Program 1.11.4 : Display the ordered list using roman type.

```
<!DOCTYPE html>
<html>
 <head>
   <title>HTML Unordered List</title>
 </head>
 <body>
<font size=5>
Name of Subjects
   <ol type="I" start="3">
     <li>Python</li>
     <li>Machine Learning</li>
     <li>Deep Learning</li>
     <li>Artificial Intelligence</li>
     <li>Blockchain</li>
   </ol>
</font>
</body>
</html>
```

Output



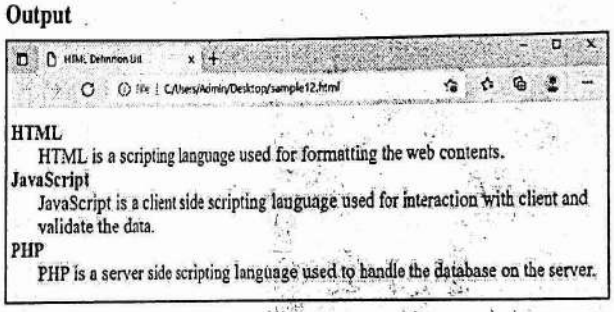
C. HTML Definition Lists

- Definition Lists are used to give the list of definitions just like the dictionary or encyclopedia. In general this is considered as the ideal way to represent a glossary, list of terms, or some kind of name/value pair list.
- Following tags are used in definition list.
  (1) **<dl>** – Defines the beginning of the list
  (2) **<dt>** – Specify a term
  (3) **<dd>** – Specify term definition
  (4) **</dl>** – Defines the end of the list

Program 1.11.5 : Write a script using the HTML definition list.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Definition List</title>
  </head>
  <body>
<font size=5>
    <dl>
      <dt><b>HTML</b></dt>
      <dd>HTML is a scripting language used for formatting
the web contents.</dd>
      <dt><b>JavaScript</b></dt>
      <dd>JavaScript is a client side scripting language used
for interaction with client and
validate the data.</dd>
<dt><b>PHP</b></dt>
<dd>PHP is a server side scripting language used to handle
the database on the server.</dd>
</font>
    </dl>
  </body>
</html>
```

Output



1.12 HTML IMAGE

- Images are very important to improve the design and the appearance of a web page.
- In HTML, images are defined with the **<img>** tag.
- Inserting an image in web page is achieved by using <img> tag.

Syntax

```
<img src="Image_URL" />
```

- Here the **src** attribute specifies the web address of the image. The src attribute's value should contain the correct image file name with its extension. PNG, JPEG or GIF images can be displayed on webpage.

The alt attribute

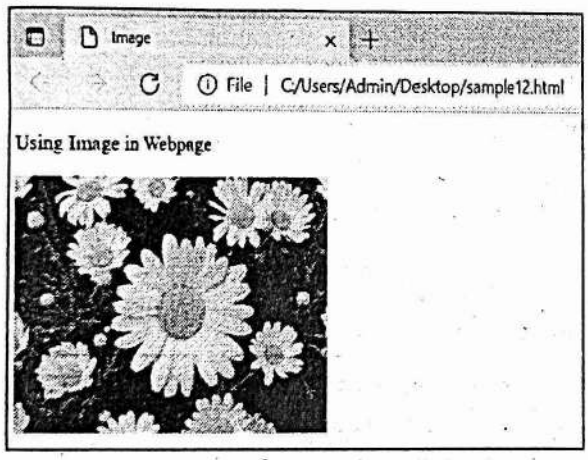
- The **alt** attribute specifies the alternate text for the image. If the image can't displayed on the webpage. So it is a mandatory attribute.
- The value of the alt attribute is a text which describes the image :

```
<img src=" Image_URL" alt="description_text">
```

Program 1.12.1 : Write a program to add an image in HTML document.

```
<!DOCTYPE html>
<html>
<head>
<title>Image </title>
</head>
<body>
<p>Using Image in Webpage</p>
<img src=" flower.jpg" alt="Flower" />
</body>
</html>
```

Output



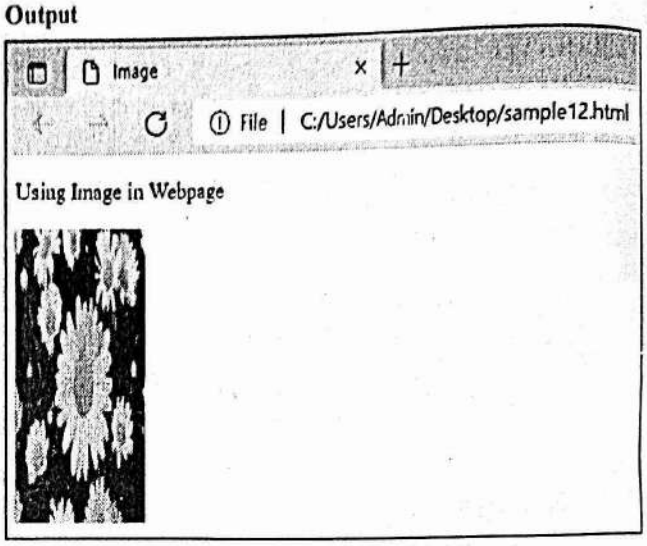
The width and height attributes

- The **width** and **height** attributes are used to set the size of the image to be displayed on the webpage.
- These attributes of <img> tag defines the width and height of the image in pixels.

Program 1.12.2 : Write program to set the size of image using width and height attributes of <img> tag.

```
<!DOCTYPE html>
<html>
<head>
<title>Image</title>
</head>
<body>
<p>Using Image in Webpage</p>
<img src=" flower.jpg " alt="Flower "
width="100"height="200">
</body>
</html>
```

Output



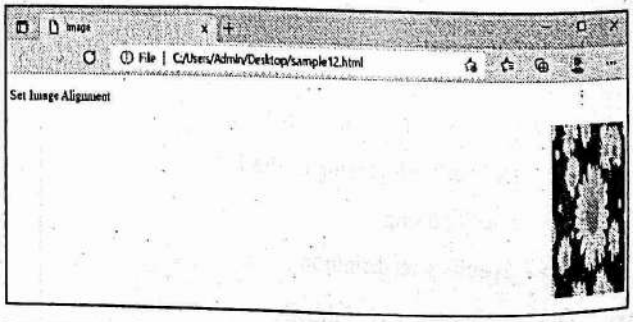
The align attribute

- The **align** attribute is used for setting the alignment of image to center or right, because by default, image will align at the left side of the webpage.
- Following Program shows how to align the image to right.

Program 1.12.3 : Write a program to set the alignment for image.

```
<!DOCTYPE html>
<html>
<head>
<title> Image </title>
</head>
<body>
<p>Set Image Alignment</p>
<img src=" flower.jpg " alt="Flower " width="100"
height="200" align="right">
</body>
</html>
```

Output



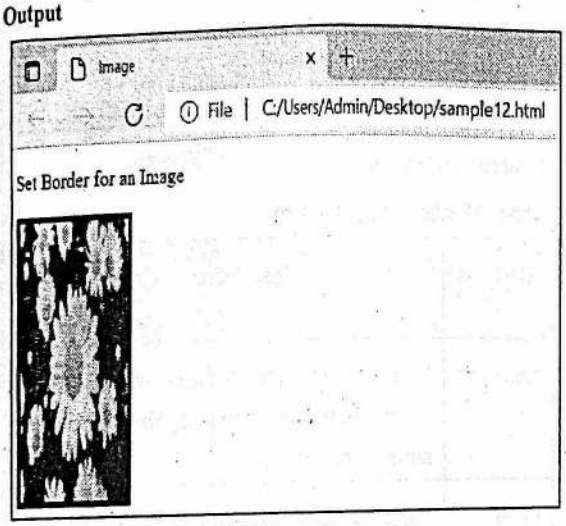
The border attribute

- The border attribute has the value which specifies the border thickness in pixels. By default, image will have a border around it, and the 0 pixel thickness means no border around the image.

Program 1.12.4 : Write a program to add an image and set the border=4 pixel to it.

```
<!DOCTYPE html>
<html>
<head>
<title>Image</title>
</head>
<body>
<p>Set Border for an Image </p>
<img src=" flower.jpg " alt="Flower " width="100"
height="200" border="4">
</body>
</html>
```

Output



Inserting Images as a Page Background

By default the webpage background is white. HTML provides following two ways to beautify the webpage background.

- The **background** attribute is used with HTML elements to set the image as its background.

Syntax

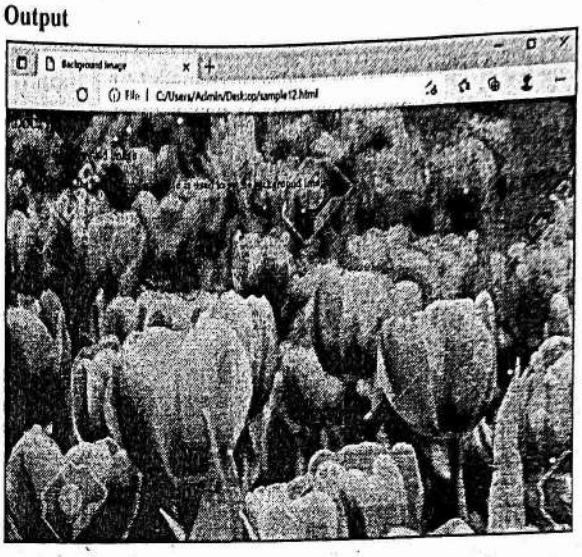
```
<tagname background="Image URL"...>
...
</tagname>
```

- For setting background the JPEG, GIF and PNG image formats are used.

Program 1.12.5 : Write a program to set the background image for body tag using background attribute.

```
<!DOCTYPE html>
<html>
<head>
<title>Background Image</title>
</head>
<body background="tulip.jpg">
<p>Setting the background image </p>
<p>In the body tag the background attribute is used to set
the background image</p>
</body>
</html>
```

Output



- Also using the background-image property of the style attribute we can add the image as background of an HTML element :

```
<body style="background-image:url('tulip.jpg')">
```

- It will produce same result as shown above.
- It is possible to set the background image for other HTML elements such as paragraph or table.
- Following example shows how to set background image on the paragraph as shown below :

```
<body>
<p style="background-image:url('tulip.jpg')">
...
</p>
</body>
```

1.13 FORMS IN HTML

- The HTML form is a part of a document which may have general content, special elements known as controls like textfields, checkboxes, submit button, radio buttons etc.

- Users usually "complete" a form by modifying its elements. This modification may include entering text, selecting controls like checkbox etc. before submitting the form to the next page for processing. This data is posted to an application like CGI, ASP Script or PHP script etc. This application will then perform essential processing on the accepted data based on defined business logics of the application.
- HTML Forms are generally necessary when we want to accept some data from the website visitors. For example e-mail account form, credit card form where we may like to accept information like name, email address, phone number, gender etc.
- The HTML **<form>** tag creates an HTML form

Syntax

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

FormTag attributes

Following is the list of form tag attributes

Sr. No	Attribute	Description
1.	Action	Url of next page(script) to which we want to submit the data
2.	Method	Method which will upload the data. The common methods used are GET and POST.
3.	target	Sets the name of the target window or frame where we want to display the result. It takes values like _blank, _self, _parent etc.
4.	enctype	Specify the way to encode the data before sending it to server

1.13.1 Interactive Elements

HTML provides various controls which can be used to collect the data using HTML form.

1. Text Input Controls

Three types of text input controls are used on forms :

(i) **Single-line Text Input Control**

This control is used for one line of user input. It is created using HTML **<input>** tag.

```
<input type="text" name="txtUser"/>
```

(ii) **Password Input Controls**

This is too a single-line text input control except that it masks the character when user enters it. It is also created using the same <input> tag.

```
<input type="password" name="txtPass"/>
```

Attributes of input tag

Sr. No	Attribute	Description
1.	Type	Specifies the type of control.
2.	Name	Used to assign a name to the control which is sent to the server to be recognized and retrieve the value.
3.	Value	This is used to set default value to the control.
4.	size	Sets the width of the control in terms of characters.
5.	maxlength	Specifies the maximum number of characters which user can give in the text box.

(iii) **Multi-line text input controls**

This is used to accept detailed information from user. It is created using HTML **<textarea>** tag.

```
<textarea rows=5 cols=30/>
```

Attributes of <textarea> tag

Sr. No.	Attribute	Description
1.	name	Used to assign a name to the control which is sent to the server to be recognized and retrieve the value.
2.	Rows	Specifies the number of rows of text area.
3.	Cols	Specifies the number of columns of text area.

2. Checkbox Control

This control is used to give list of items where multi-selection is allowed. It is also created using <input> tag but the value for type attribute is set as **checkbox.**

```
<input type="checkbox" name="chkPhoenix"/>
```

3. Radio Button Control

This control is used to give list of items where single-selection is allowed. It is also created using <input> tag but where the value for type attribute is set as radio.

```
<input type="radio" name="optEngg"/>
```

4. Select Box Control

A select box is also known as drop down box. It provides option to list number of options in the form of drop down list. Here the user can select one or multiple options.

```
<select name="sn">
<option>--</option>
-----
</select>
```

Attributes of <select> tag

Sr. No	Attribute	Description
1.	name	Used to assign a name to the control which is sent to the server to be recognized and retrieve the value.
2.	size	This can be used to specify a scrolling list box.
3.	multiple	If this option is set, it creates listbox and user can select multiple elements. If not then combobox is created where single selection is allowed.

Attributes of <option> tag

Sr. No	Attribute	Description
1.	value	The value that will be used to check whether the option is selected or not.
2.	Selected	It sets the option by default selected
3.	Label	Sets label to options

5. File Upload Box

This is used to give option to user for file upload. This is created using <input> tag. The type attribute is set to file.

Attributes of file upload box

Sr. No	Attribute	Description
1.	accept	Specifies the types of files that the server accepts.

6. Button Controls

HTML provides different types of buttons. The <input> tag is used to create buttons.

Types of Buttons

Sr. No	Type	Description
1.	Submit	This button submits a form.
2.	Reset	This button resets (clears) form controls.
3.	Button	This button calls a client-side script (function).

7. Hidden Form Fields

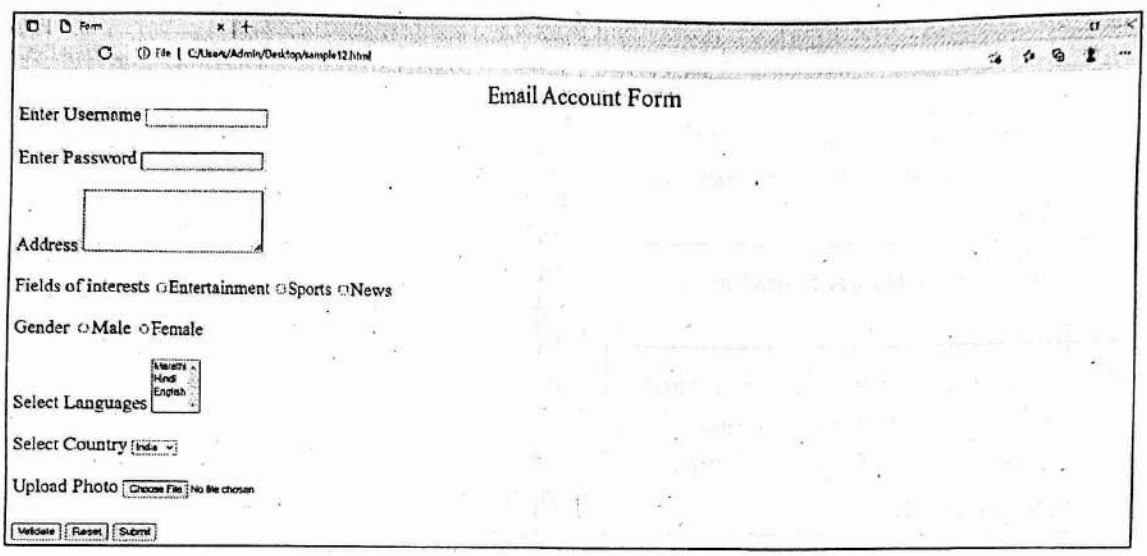
This element is used to hide the data while sending it to the server. This control as name suggests does not display on the form.

Program 1.13.1 : Write a program to display the form containing all controls.

```
<html>
<head>
<title>
Form
</title>
</head>
<body>
<font size=6>
<center>
Email Account Form
</center>
</font>
<font size=5>
Enter Username <input type="text" name ="txtUser"
maxlength=30/><br><br>
Enter Password <input type="password" name
="txtPass"/><br><br>
Address <textarea name="add" rows=5
cols=30></textarea><br><br>
Fields of interests
<input type="checkbox" name ="chk1"/>Entertainment
<input type="checkbox" name ="chk2"/>Sports
<input type="checkbox" name ="chk3"/>News<br><br>
Gender
<input type="radio" name ="optGender"/>Male
<input type="radio" name
="optGender"/>Female<br><br>
Select Languages
<select name="lang" multiple>
<option>Marathi</option>
<option>Hindi</option>
<option>English</option>
</select>
<br><br>
Select Country
```

```
<select name="cntry">
<option>India</option>
<option>USA</option>
<option>Japan</option>
<option>UK</option>
</select>
<br><br>
Upload Photo
<input type = "file" name = "fileupload" accept = "image/*"
/>
<br><br>
<input type="button" name = "vld" value="Validate">
<input type="reset" name = "rst" value="Reset">
<input type="submit" name = "sub" value="Submit">
<input type = "hidden" name = "pagenm" value = "11" />
</font>
</body>
</html>
```

Output



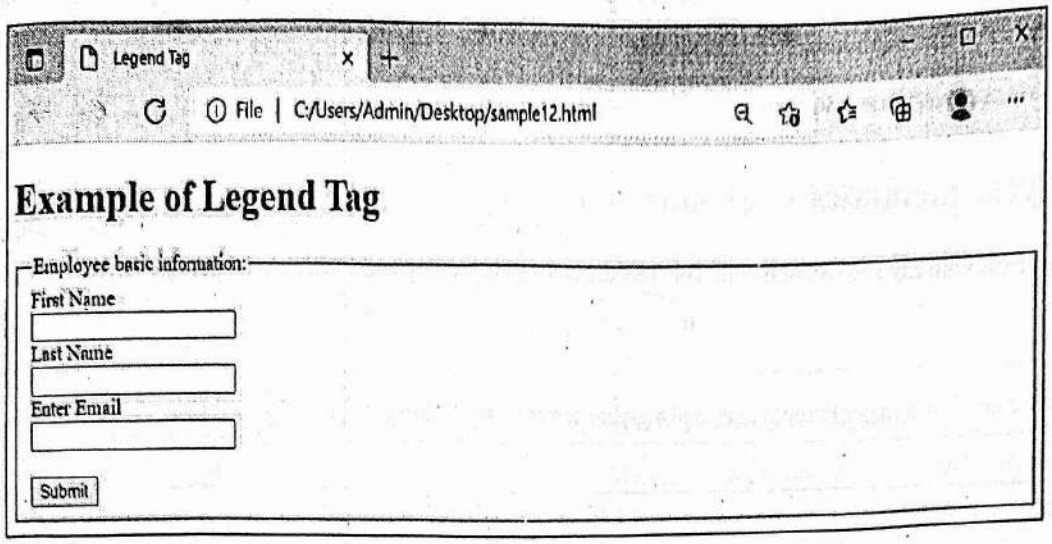
1.13.2 HTML <fieldset >and <legend> tag

- HTML <legend> tag is used to insert a title or caption to its parent element such as <fieldset>. The <legend> element must be the first child of <fieldset > element.
- By using <legend> tag with <form> elements, it is easy to understand the purpose of grouped form elements.

Program 1.13.2

```
<!DOCTYPE html>
<html>
<head>
<title>Legend Tag</title>
</head>
<body>
<h1>Example of Legend Tag</h1>
<form>
    <fieldset>
  <legend>Employee basic information:</legend>
  <label>First Name</label><br>
  <input type="text" name="fname"><br>
  <label>Last Name</label><br>
  <input type="text" name="lname"><br>
  <label>Enter Email</label><br>
  <input type="email" name="email"><br><br>
  <input type="Submit"><br>
    </fieldset>
</form>
</body>
</html>
```

Output



1.14 HTML5 CONTROLS

1.14.1 Semantic Elements in HTML5

- In HTML4, developers have to use their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.
- This is so difficult for search engines to identify the correct web page content. Now in HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier. It now allows data to be shared and reused across applications, enterprises, and communities.
- Semantic elements can increase the accessibility of your website, and also helps to create a better website structure.

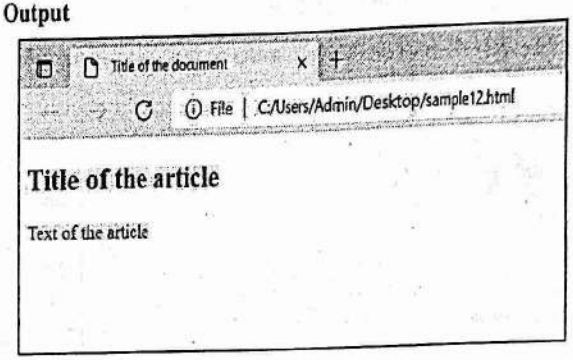
A. HTML5 <article> Element

HTML <article> element defines article content within a document, page, application, or a website. It can be used to represent a forum post, a magazine, a newspaper article, or a big story.

Program 1.14.1

```
<!DOCTYPE html>
<html>
 <head>
  <title>Title of the document</title>
 </head>
 <body>
  <article>
   <h1>Title of the article</h1>
   <p>Text of the article</p>
  </article>
 </body>
</html>
```

Output



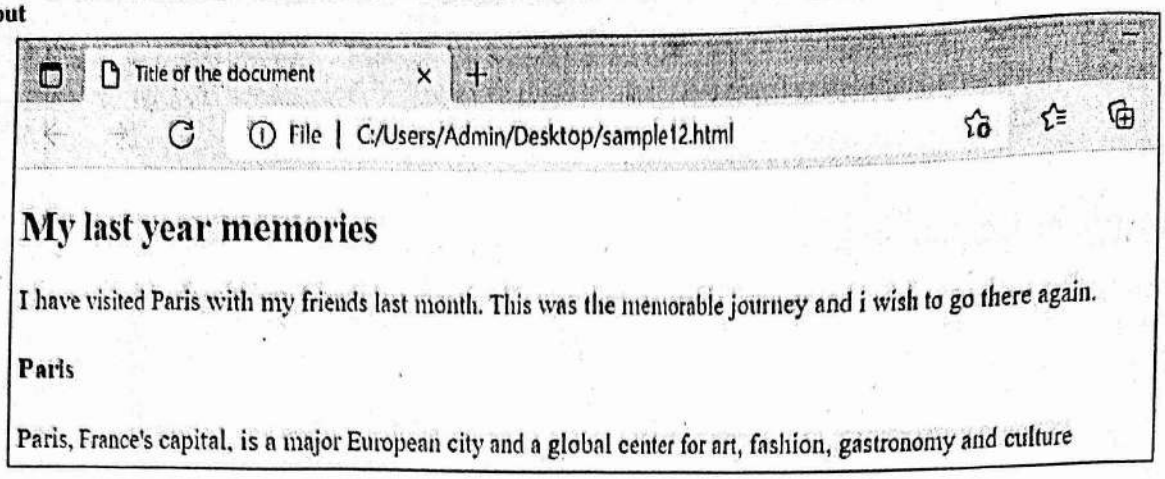
B. HTML5 <aside> Element

The <aside> element represent the content which is indirectly giving information to the main content of the page. It is frequently represented as a sidebar.

Program 1.14.2

```
<!DOCTYPE html>
<html>
 <head>
  <title>Title of the document</title>
 </head>
 <body>
  <h2>My last year memories</h2>
  <p>I have visited Paris with my friends last month. This was the memorable journey and i wish to go there again.</p>
   <aside>
    <h4>Paris</h4>
    <p>Paris, France's capital, is a major European city and a global center for art, fashion, gastronomy and culture</p>
   </aside>
  </body>
</html>
```

Output



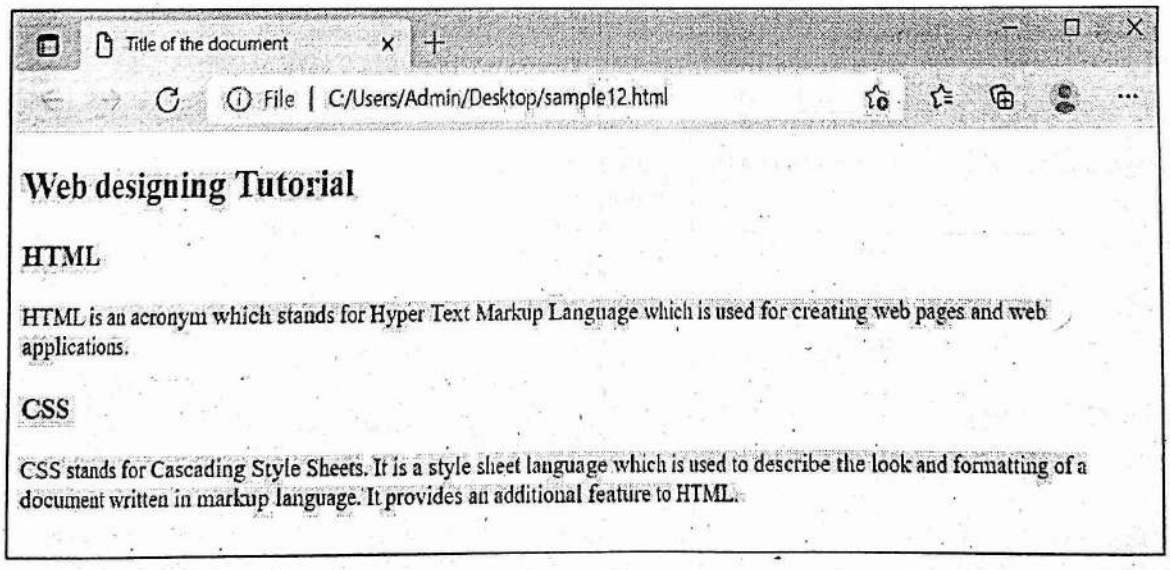
C. HTML5 <section> Element

The <section> element is used to represent the standalone section within an HTML document. A page can have various sections and each section can contain any content, but headings for each section is not mandatory.

Program 1.14.3

```
<!DOCTYPE html>
<html>
 <head>
  <title>Title of the document</title>
 </head>
 <body>
  <h2>Web designing Tutorial</h2>
<section>
  <h3>HTML</h3>
  <p>HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications.</p>
 </section>
 <section>
  <h3>CSS</h3>
  <p>CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML.</p>
 </section>
   </body>
</html>
```

Output



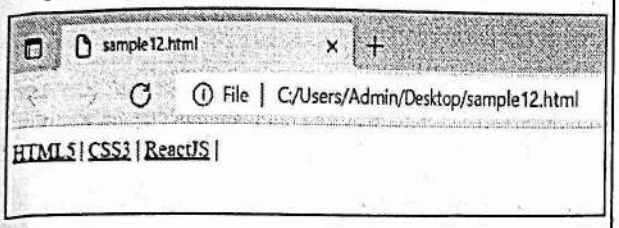
D. HTML5 <nav> Element

The HTML <nav> element is used to define a set of navigation links.

Program 1.14.4

```
<!DOCTYPE html>
<html>
<body>
<nav>
 <a href="HTML.html">HTML5</a> |
 <a href="CSS.html">CSS3</a> |
 <a href="React.html">ReactJS</a> |
</nav>
</body>
  </html>
```

Output



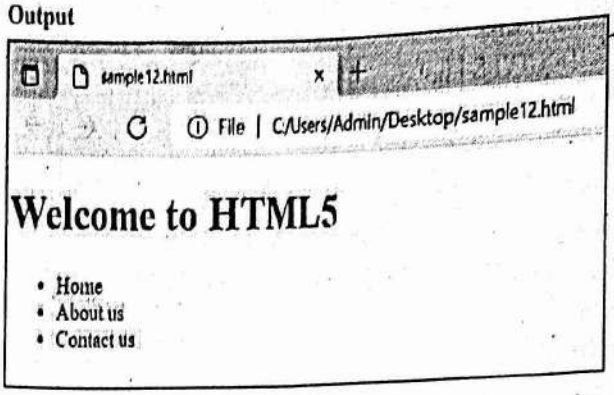
E. HTML5 <header> Element

The <header> element represent the header of the document which can contain introductory content or navigation links.

Program 1.14.5

```
<!DOCTYPE html>
<html>
<body>
 <header>
   <h1>Welcome to HTML5</h1>
   <nav>
     <ul>
       <li>Home </li>
       <li>About us </li>
       <li>Contact us</li>
     </ul>
   </nav>
 </header>
</body>
</html>
```

Output



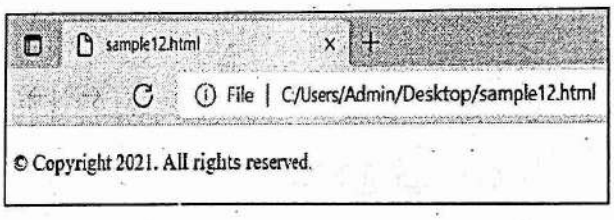
F. HTML5 <footer> Element

The <footer> tag defines the footer of an HTML document or page.

Program 1.14.6

```
<!DOCTYPE html>
<html>
<body>
  <footer>
    <p>© Copyright 2021. All rights reserved. </p>
  </footer>
</body>
</html>
```

Output



G. HTML5 <figure> and <figcaption> tag

- The <figcaption> element is used to provide a caption to an image.
- It is an optional tag and can appear before or after the content within the <figure> tag.
- Only one <figcaption> element can be nested within a <figure> tag although the <figure> element itself may contain multiple other elements like <img> or <code>.
- The <figcaption> element is used with <figure> element and it can be placed as the first or last child of the <figure> element.

Program 1.14.7

```
<!DOCTYPE html>
<html>
<body>
  <figure>
    <img src="tajmahal.jpg" alt="Taj Mahal"/>
    <figcaption>Fig.1.1 - A front view of the great Taj Mahal
in Agra.</figcaption>
    </figure>
</body>
</html>
```

Output

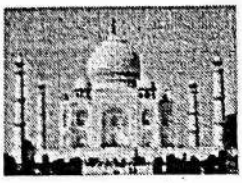


Fig.1.1 - A front view of the great Taj Mahal in Agra.

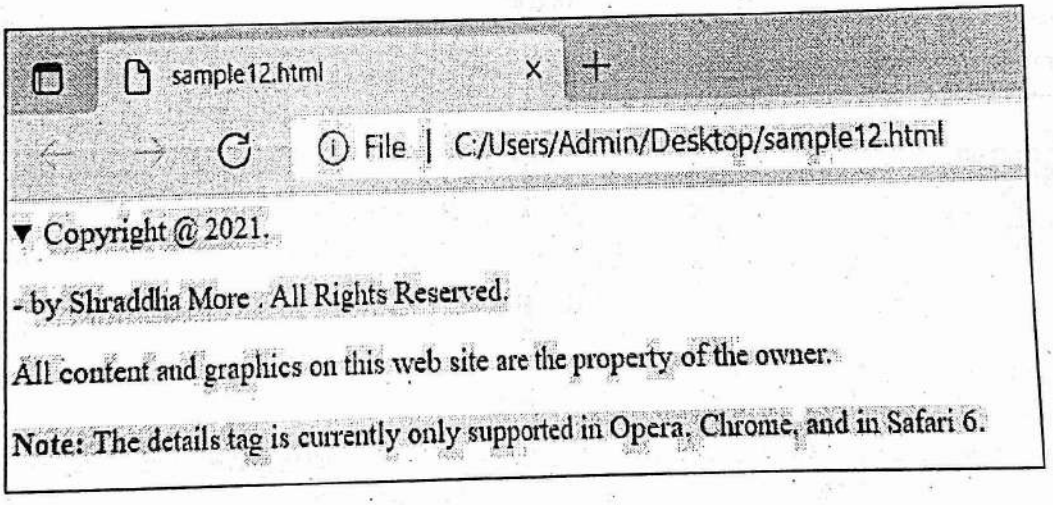
H. HTML <details> Tag

- HTML <details> tag is used to specify the additional details on the web page that the user can view or hide on demand.
- It is used as a disclosure widget from which user can retrieve additional information or control.
- It is used together with a relevant tag known as <summary>. Technically, there is no need of summary tag, but if you ignore this then the browser will use some default text.
- The <details> tag wraps all the content which you want to show or hide and the <summary> tag contains the summary and the title of the section.

Program 1.14.8

```
<!DOCTYPE html>
<html>
<body>
  <details>
    <summary>Copyright @ 2021.</summary>
    <p> - by Shraddha More . All Rights Reserved.</p>
    <p>All content and graphics on this web site are the
property of the owner.</p>
  </details>
    <p><b>Note:</b> The details tag is currently only
supported in Opera, Chrome, and in Safari 6.</p>
</body>
</html>
```

Output



List of Semantic/Structural elements

Controls	Description
<article>	Defines an article.
<aside>	Defines content aside from the page content.
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside of it.
<command>	Defines a command button that a user can invoke.
<details>	Specifies additional details that the user can view or hide on demand.

Controls	Description
<figcaption>	Defines a caption for a <figure> element.
<figure>	Specifies self-contained content.
footer>	Defines a footer for a document or section.
<header>	Defines a header for a document or section.
<hgroup>	Group heading (<h1> to <h6>) elements.
<mark>	Defines marked/highlighted text.
<meter>	Defines a scalar measurement within a known range, or a fractional value.
<nav>	Defines navigation links.
<output>	Represents the result of a calculation.
<progress>	Represents the progress of a task.
<rp>	Defines what to show in browsers that do not support ruby annotations (for East Asian typography).
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography).
<ruby>	Defines a ruby annotation (for East Asian typography).
<section>	Defines a section in a document.
<summary>	Defines a visible heading for the <details> element. The heading can be clicked to view/hide the details.
<time>	Defines a date/time.

1.14.2 Media Elements

- HTML5 provides the feature of native audio and video support without the need of Flash.
- For this purpose HTML5 provide tags such as <audio> and <video> which adds media to a website.
- These tags contain **src** attribute to identify the media source and **controls** attribute so that user will be able to play and pause the media.

A. Embedding Audio

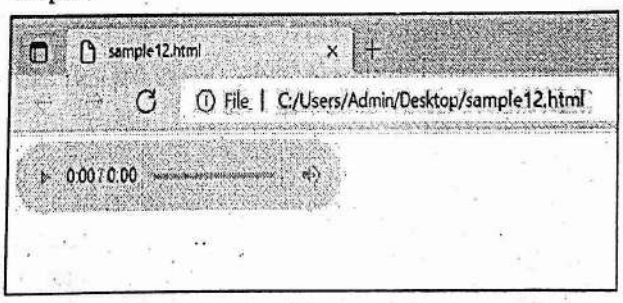
- HTML5 provides <audio> tag which helps to embed content related to sound in an HTML or XHTML document as follows.

```
<audio src="foo.wav" controls autoplay>
 Your browser does not support the <audio> element.
</audio>
```

- The specification of current HTML5 draft does not clearly mention the allowed audio formats browsers should support in the audio tag.
- But frequently preferred audio formats are **ogg, mp3** and **wav**.
- The <source> tag is used to specify media along with media type and various attributes.
- Multiple source elements are allowed in an audio element and the first recognized format will be used by the browser.

```
<!DOCTYPE HTML>
<html>
 <body>
  <audio controls autoplay>
   <source src="/songs/s1.ogg" type="audio/ogg" />
   <source src="/songs/s1.wav" type="audio/wav" />
  </audio>
 </body>
</html>
```

Output



Audio Attribute Specification

The HTML5 audio tag provides following attributes to control the look and feel as well as several functionalities of the control.

Attribute	Description
autoplay	It is of type Boolean. If mentioned, the audio will automatically start to play back as soon as possible without waiting to finish loading the data.
autobuffer	It is of type Boolean. If mentioned, the audio will automatically start buffering even if it's not set to automatically play.
controls	It allows the user to control audio playback, including volume, seeking, and pause/resume playback.
loop	It is of type Boolean. If mentioned, will allow audio to automatically starts again when reach to the end.
preload	If this attribute is mentioned, then the audio will be loaded at page load, and ready to run. It will be ignored in case autoplay is present.
src	The URL of the audio to embed.

B. Embedding Video

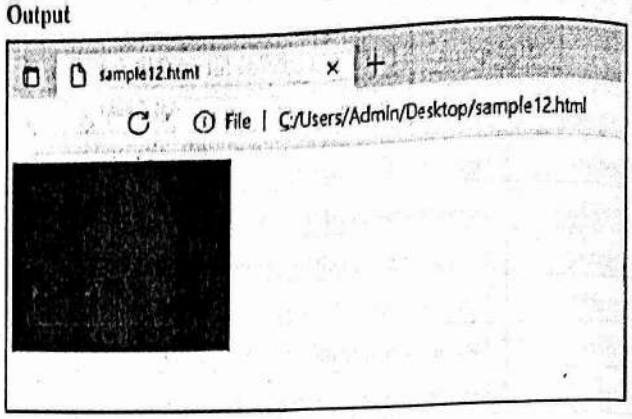
- HTML5 provides <video> tag which helps to embed content related to video in an HTML or XHTML document as follows.

```
<video src="s1.mp4" width="200" height="150" controls>
</video>
```

- The specification of current HTML5 draft does not clearly mention the allowed video formats browsers should support in the audio tag.
- But frequently preferred audio formats are ogg, and mpeg4.
- The <source> tag is used to specify media along with media type and various attributes.
- Multiple source elements are allowed in video element and the first recognized format will be used by the browser.

```
<!DOCTYPE HTML>
<html>
  <body>
    <video width="200" height="150" controls autoplay>
      <source src="/vsong/s1.ogg" type="video/ogg" />
      <source src="/vsong/s1.mp4" type="video/mp4" />
    </video>
  </body>
</html>
```

Output



1.14.3 HTML5 New <input> types

Controls	Description
color	Used for input fields that should contain a color.
date	Used for input fields that should contain a date.
datetime	Allows the user to select a date and time (with time zone).
datetime-local	Allows the user to select a date and time (no time zone).
email	Used for input fields that should contain an e-mail address.
month	Allows the user to select a month and year.
number	Used for input fields that should contain a numeric value.
range	Used for input fields that should contain a value within a range.
search	Used for search fields (a search field behaves like a regular text field).
tel	Used for input fields that should contain a telephone number.
time	Allows the user to select a time (no time zone).
url	Used for input fields that should contain a URL address.
week	Allows the user to select a week and year.

1.15 HTML DRAG AND DROP

The HTML5 drag and drop feature allows the user to drag and drop an element to another location. The drop location may be a different application. While dragging an element a translucent representation of the element is follow the mouse pointer.

Events for Drag and Drop feature

A number of events are fired during the various stages of the drag and drop operation. But mouse events such as mousemove are not fired during a drag operation.

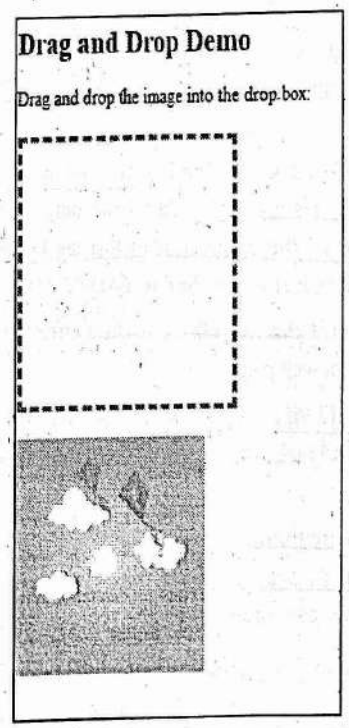
The following table provides you a brief overview of all the drag and drop events.

Event	Description
ondragstart	Fires when the user starts dragging an element.
ondragenter	Fires when a draggable element is first moved into a drop listener.
ondragover	Fires when the user drags an element over a drop listener.
ondragleave	Fires when the user drags an element out of drop listener.
ondrag	Fires when the user drags an element anywhere; fires constantly but can give X and Y coordinates of the mouse cursor.
ondrop	Fires when the user drops an element into a drop listener successfully.
ondragend	Fires when the drag action is complete, whether it was successful or not. This event is not fired when dragging a file to the browser from the desktop.

Program 1.15.1

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drag and Drop Elements in HTML</title>
<script>
 function dragStart(e) {
   // Sets the operation allowed for a drag source
   e.dataTransfer.effectAllowed = "move";
   // Sets the value and type of the dragged data
   e.dataTransfer.setData("Text", e.target.getAttribute("id"));
 }
 function dragOver(e) {
   // Prevent the browser default handling of the data
   e.preventDefault();
   e.stopPropagation();  }
 function drop(e) {
   // Cancel this event for everyone else
   e.stopPropagation();
   e.preventDefault();
   // Retrieve the dragged data by type
   var data = e.dataTransfer.getData("Text");
   // Append image to the drop box
   e.target.appendChild(document.getElementById(data));
 }
</script>
<style>
 #dropBox {
   width: 200px;
   height: 200px;
   border: 5px dashed gray;
   background: lightyellow;
   text-align: center;
   margin: 20px 0;
   color: orange;
 }
 #dropBox img {
   margin: 10px;  }
</style>
</head>
<body>
  <h2>Drag and Drop Demo</h2>
  <p>Drag and drop the image into the drop box:</p>
  <div id="dropBox" ondragover="dragOver(event);"
ondrop="drop(event);">
    <!--Dropped image will be inserted here-->
  </div>
  <img src="/examples/images/kites.jpg" id="dragA"
draggable="true" ondragstart="dragStart(event);"
width="180" height="180" alt="Flying Kites">
</body>
</html>
```

Output



1.16 INTRODUCTION OF CSS

- A Style Sheet is a collection of style rules that that tells a browser how the various styles are to be applied to the HTML tags to present the document.
- CSS - Cascading Style Sheet is a simple design scripting language intended to simplify the process of making the web pages attractive with high level formatting.
- CSS is used to manage the look and feel of the web pages. CSS is used to control the color of the text, the style of fonts, the way columns are sized and laid out, different background images or colors used, the spacing between paragraphs, layout designs, and variations in the display for various devices, screen sizes or resolutions and variety of other effects.
- The CSS is very easy to learn and understand but it has strong control over the presentation of web page. In general, CSS is integrated with the markup languages like HTML or XHTML.
- In simple words, CSS is nothing but declaration of style sheets which can be repeatedly used. It resembles to the function concept of C language. That means define once and use anytime anywhere repeatedly.

1.16.1 Advantages of CSS

GQ. State the advantages of CSS.

1. **CSS Saves Time :** Once written, CSS can be used in multiple HTML documents repeatedly. It helps to define style for different HTML elements and apply it to multiple web pages as per requirement.
2. **Pages Load Faster :** When CSS is used, there is no need to write HTML tag attributes repeatedly. Just write only once and can be applied to all the occurrences of that tag in the webpage. As code is less, it speed up the download process.
3. **Easy Maintenance :** Just change in the style sheet effects all the elements in different web pages.
4. **Superior Styles to HTML :** CSS has a wider range of attributes which helps to make the look of webpage far better than HTML.
5. **Multiple Device Compatibility :** Style sheet contents are compatible for multiple types of devices. With use of same HTML document, we can present various versions of a website.
6. **Offline Browsing :** CSS helps to store the web applications on local machines by using an offline cache. This helps to view offline websites. The cache provides fast loading of the website and improved overall performance.
7. **Platform Independence :** The CSS supports reliable platform independence and also supports all the latest browsers.

1.16.2 Inserting CSS in an HTML Page

GQ. What are the types of CSS? Explain with example.

There are three ways of inserting a style sheet in the webpage :

1. Inline Style Sheet
2. Internal Style Sheet
3. External Style Sheet

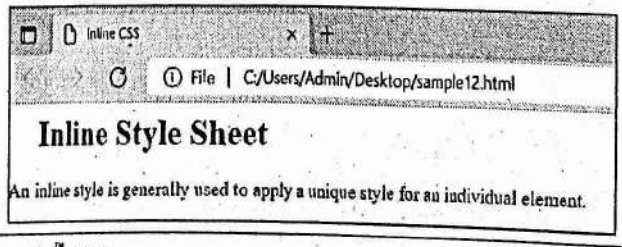
1. Inline Style Sheet

- An inline style is generally used to apply a unique style for an individual element.
- The Inline style is specific for the individual tag. The HTML "style" attribute is used by the inline style to style a particular tag. The Inline style generally useful for an individual CSS change which we do not want to use repeatedly in the site.

Program 1.16.1

```
<html>
<head>
<title>
Inline CSS
</title>
</head>
<body>
<h1 style="color:blue;margin-left:30px;">SJCEM</h1>
<p style="color:red;font-size:18px"> An inline style is
generally used to apply a unique style for an individual
element.
</p>
</body>
</html>
```

Output



The external .css file should not contain any HTML tags

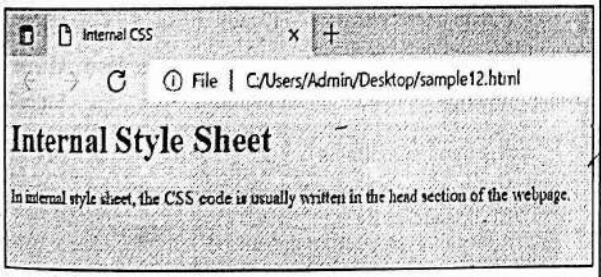
2. Internal Style Sheet

- In internal style sheet, the CSS code is usually written in the head section of the webpage. This simplifies the applications of styles like classes or id's in contrast to repeated use of the code.
- While creating an internal style sheet in the web page, we have to use the <style></style> HTML tags in the Head section of the webpage. The entire code of the Internal CSS style sheet is included between the <head></head> section of the websites.
- Following is an example of how an internal style sheet looks like.

Program 1.16.2

```
<html>
<head>
<title>
Internal CSS
</title>
<style type="text/css">
body {background-color: linen}
h1 {color:blue;margin-left:30px>
p {color:red;font-size:18px>
</style>
</head>
<body>
<h1>Internal Style Sheet</h1>
<p>In internal style sheet, the CSS code is usually written in the head section of the webpage.</p>
</body>
</html>
```

Output



3. External Style Sheet

- The External Style sheet is a file with extension .css which we link to website. This CSS file is used to declare the style sheets. Whenever any change is made in this css file, it gets reflected in all the web pages of the website. This simplifies our work and avoids making change in each and every page where the css is used.
- Now a day most of the websites use external style sheets. These are the styles which are written in a separate document or file and then linked to various web pages. External style sheets affects all the documents which are attached to, this means that if we have a 100-page website where all the pages use the same style sheet, just change in the style sheets will make visual change in all the web pages which are linked to it.

Program 1.16.3

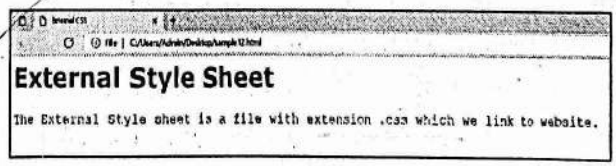
CSS file: style.css

```
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p {
  color: red;
  font-family: courier;
  font-size: 160%;
}
```

HTML file: style.html

```
<html>
<head>
<title>
External CSS
</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<h1>External Style Sheet </h1>
<p> The External Style sheet is a file with extension .css which we link to website. </p>
</body>
</html>
```

Output



1.17 INHERITANCE

GQ. What is Inheritance in CSS?

CSS inheritance works on a property by property basis. When applied to an element in a document, a property with the value 'inherit' will use the same value as the parent element has for that property.

- An elements style values are copied from its parent element.

Property_name : inherit;

background_color

Example

CSS File

```
.foo {
 background-color: white;
 color: black;
}

.bar {
 background-color: inherit;
 color: inherit;
 font-weight: normal;
}
```

HTML file

```
<div class="foo">
 <p class="bar">
  Hello, world. This is a very short paragraph!
 </p>
</div>
```

- Here, background colour of the div element is white, because the background-color property is set to white. The background colour of the paragraph is also white, because the background colour property is set to inherit, and the background colour of the parent element (the div) is set to white.
- The inherit value does not require that the parent element have the same property set explicitly; it works from the computed value. In the above example, the color property of the paragraph has a value of "inherit", but the computed value is "black" because it inherits.

1.18 BACKGROUNDS

GQ. How to set background images in CSS?

- There are several CSS properties used for setting the background images of an element. To set the background for entire page these properties are used with <body> element.
- Following are the CSS background- properties which are used to set an image as background of an element
  - **background-image** : The value of this property is an image URL to be used as background. This is used to specify the source URL for the background image.
  - **background-repeat** : This property is used to specify whether and how the background image should repeat or tile.
  - **background-attachment** : This property is used to specify whether and how the background image should scroll with the Webpage content.

background

  - **background-position** : This property is used to specify the position of background image relative to the element.

color

  - **background-image** : The value of background-image property is a URL of the image as shown below :

```
body { background-image: url(/folder/subfolder/image.jpg); }
```

- We can write image path in url() in either of three ways:
- Relative path based on the root domain of the CSS file : when the image is in root directory of CSS file this path is used. The path begins with a slash (/) indicating the root directory.

e.g.: url(/folder/subfolder/image.jpg)

- Relative path based on the immediate directory of the CSS file : when the image is in the same directory of CSS file this path is used.

e.g.: (url(folder/subfolder/image.jpg))

- Absolute path : when the image stored externally.

e.g.: url(http://example.com/folder/subfolder/image.jpg)

- **background-repeat** : By default this property is set to repeat the image as both horizontally and vertically. Following are the values which can be used for this property :
  - **repeat** : By default this value is set for background-repeat property. This value repeats the background image in both directions.
  - **repeat-x** : This value repeats the background image only on the x-axis i.e. horizontally across the element.
  - **repeat-y** : This value repeats the background image only on the y-axis i.e. vertically down to the element.
  - **no-repeat** : This value doesn't repeats the background image.
- **background-attachment** : This property specifies how the background image responds to scrolling.
  - **fixed** : This specifies that the background image is fixed relative to the browser window means scrolling the entire page or contents of the element doesn't leads to move the background image.
  - **scroll** : This specifies that the background image is fixed relative to the element.
  - **local** : This specifies that the background image is fixed relative to the contents of the image means if the contents of the image are scrolled, then the background image also scroll with them. And also if the entire page

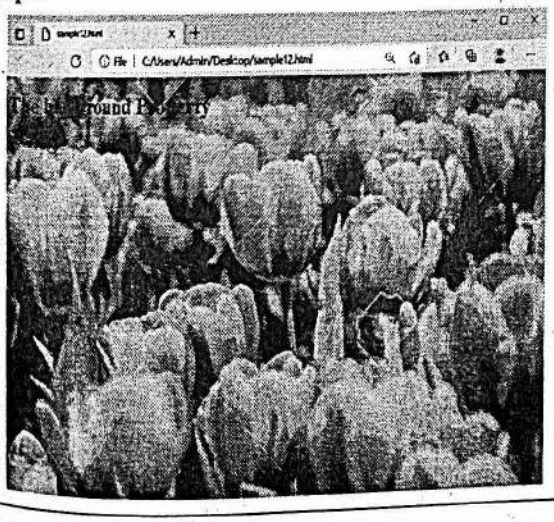
is scrolled by which the element content moved, then the background image also moves too.

- **background-position** : This is used to specify the position of a background image.
- The easiest way to position the background image is to use the positioning keywords left, right, top, bottom, and center.
- Combining this keywords gives nine possibilities like :
  - Left top
  - Left center
  - Left bottom
  - Center top
  - Center center
  - Center bottom
  - Right top
  - Right center
  - Right bottom

Program: 1.18.1 : Set different background properties in one declaration:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
 background: url("tulip.jpg") no-repeat fixed center;
}
</style>
</head>
<body>
<h1>The background Property</h1>
<p>This is some text</p>
</body>
</html>
```

Output



1.19 BORDERS

The CSS border properties allow you to specify the style, width, and color of an element's border.

The border-style property specifies what kind of border to display. The following values are allowed :

- **dotted** - Defines a dotted border.
- **dashed** - Defines a dashed border.
- **solid** - Defines a solid border.
- **double** - Defines a double border.
- **groove** - Defines a 3D grooved border. The effect depends on the border-color value.
- **ridge** - Defines a 3D ridged border. The effect depends on the border-color value.
- **inset** - Defines a 3D inset border. The effect depends on the border-color value.
- **outset** - Defines a 3D outset border. The effect depends on the border-color value.
- **none** - Defines no border.
- **hidden** - Defines a hidden border.

Program 1.19.1

```
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
</style>
</head>
<body>

<h2>The border-style Property</h2>
<p>This property specifies what kind of border to
display:</p>
```

```
<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>
</body>
</html>
```

Output

The border-style Property

This property specifies what kind of border to display:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border.

A ridge border.

An inset border.

An outset border.

No border.

A hidden border.

A mixed border.

1.20 BORDER IMAGES

The CSS border-image property allows you to specify an image to be used instead of the normal border around an element. The property has three parts :

- The image to use as the border
- Where to slice the image
- Define whether the middle sections should be repeated or stretched

For border-image to work, the element also needs the border property set.

Border Image Properties

- **border-image:** A shorthand property for setting all the border-image-* properties.
- **border-image-source:** Specifies the path to the image to be used as a border.
- **border-image-slice:** Specifies how to slice the border image.
- **border-image-width:** Specifies the widths of the border image.
- **border-image-outset:** Specifies the amount by which the border image area extends beyond the border box.
- **border-image-repeat:** Specifies whether the border image should be repeated, rounded or stretched.

Program 1.20.1

```
<!DOCTYPE html>
<html>
<head>
<style>
#borderimg {
 border: 10px solid transparent;
```

```
  padding: 15px;
  border-image: url(border.png) 30 round;
}
</style>
</head>
<body>
<h1>The border-image Property</h1>
<p>Here, the middle sections of the image are repeated to
create the border:</p>
<p id="borderimg">border-image: url(border.png) 30
round;</p>
<p>Here is the original image:</p><img
src="border.png">
</body>
</html>
```

Output

The border-image Property

Here, the middle sections of the image are repeated to create the border:

border-image: url(border.png) 30 round;

Here is the original image:

1.21 COLORS

GQ. Explain colors and properties in CSS.

The color property specifies the color of text.

Syntax

color: color name

Property Values

Value	Description
Color	Specifies the text color.
Initial	Sets this property to its default value.
Inherit	Inherits this property from its parent element.

CSS Color Values

- The color values are specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values :

Format of RGB value

rgb(red, green, blue)

- Here intensity of color is mentioned which is in between 0 to 255.

Format of HEX value

#rrggbb each color

- Specifies the intensity of color (rr-red, gg-green, bb-blue) which is from 00 to ff means (in between 0 to 255 in decimal).

Format of HSL value

hsl(hue, saturation, lightness)

- Color can be specified in the form of hue , saturation and lightness of the color.
- Hue is a degree value on the color wheel which is in between 0 to 360 where 0 indicates red, 120 indicates green, and 240 indicates blue.
- Saturation is a percentage value, 0% indicates a shade of gray, and 100% indicates the pure color.
- Lightness is a percentage value, 0% indicates black, 50% indicates neither light nor dark, 100% indicates white.

Format of RGBA value

rgba(red, green, blue, alpha)

- Where "A" stands for alpha means the transparency of color. The value of alpha 0.0 indicates fully transparent and 1.0 indicates not transparent at all.

Format of HSLA value

hsla(hue, saturation, lightness, alpha)

Following program uses the CSS color property to display the colored text:

Program 1.21.1

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  color: rgb(201, 76, 76);
}
</style>
</head>
<body>
<h1>This is heading 1</h1>
<p>The color of the text can be specified with an RGB
value.</p>
</body>
</html>
```

Output

This is heading 1

The color of the text can be specified with an RGB value.

1.22 CSS SHADOW EFFECTS

With CSS you can add shadow to text and to elements.

1.22.1 CSS Text Shadow

- This property is used to set the text shadow around a text.
- This property is supported by few browsers.
- For example the text-shadow property has following format :

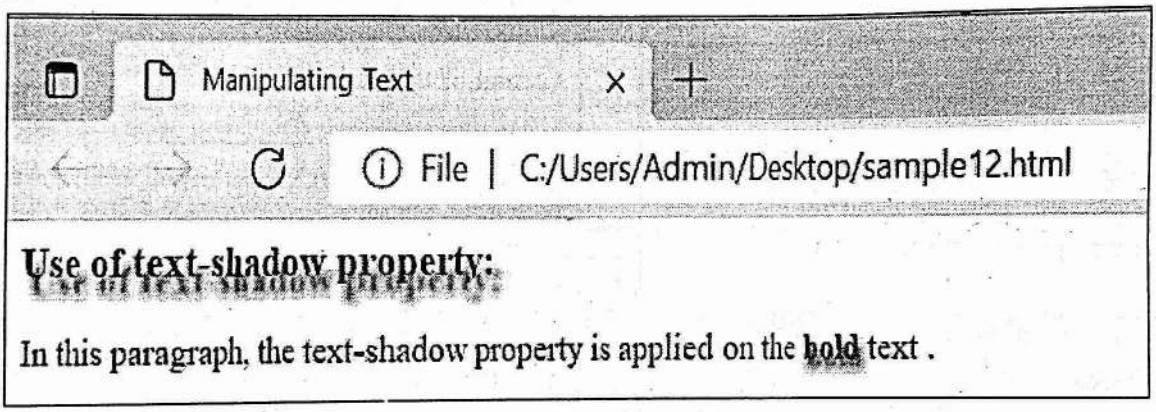
```
text-shadow:4px 4px 8px blue
```

- Where first two values indicate the position of shadow from the text, and the third value indicate the opacity of shadow. 1px indicates dark shadow. And the fourth value indicates the shadow color.

Program 1.22.1 : Code to demonstrate the use of text-shadow property

```
<html>
<title>Manipulating Text</title>
  <head>
     <style>

       h3 { text-shadow:4px 8px 3px red;}
       b { text-shadow:2px 4px 3px maroon;}
       </style>
  </head>
  <body>
    <h3>
      Use of text-shadow property:
    </h3>
    <p >
      In this paragraph, the text-shadow property is applied
on the <b>bold</b> text .
    </p>
  </body>
</html>
```

Output



1.22.2 CSS box Shadow

The CSS box-shadow property applies shadow to elements. That syntax is:

```
box-shadow: [horizontal offset] [vertical offset] [blur radius]
[optional spread radius] [color];
```

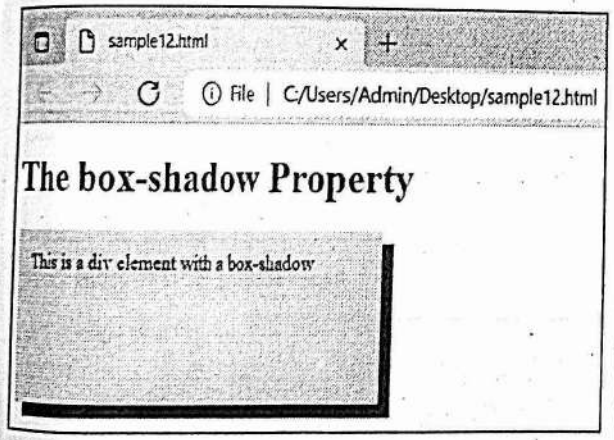
- The horizontal offset (required) of the shadow, positive means the shadow will be on the right of the box, a negative offset will put the shadow on the left of the box.
- The vertical offset (required) of the shadow, a negative one means the box-shadow will be above the box, a positive one means the shadow will be below the box.
- The blur radius (required), if set to 0 the shadow will be sharp, the higher the number, the more blurred it will be, and the further out the shadow will extend. For instance a shadow with 5px of horizontal offset that also has a 5px blur radius will be 10px of total shadow.

- The spread radius (optional), positive values increase the size of the shadow, negative values decrease the size. Default is 0 (the shadow is same size as blur).
- Color (required) – takes any color value, like hex, named, rgba or hsla. If the color value is omitted, box shadows are drawn in the foreground color (text color).

Program 1.22.2

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 padding: 15px;
 background-color: yellow;
 box-shadow: 10px 10px;
}
</style>
</head>
<body>
<h1>The box-shadow Property</h1>
<div>This is a div element with a box-shadow</div>
</body>
</html>
```

Output

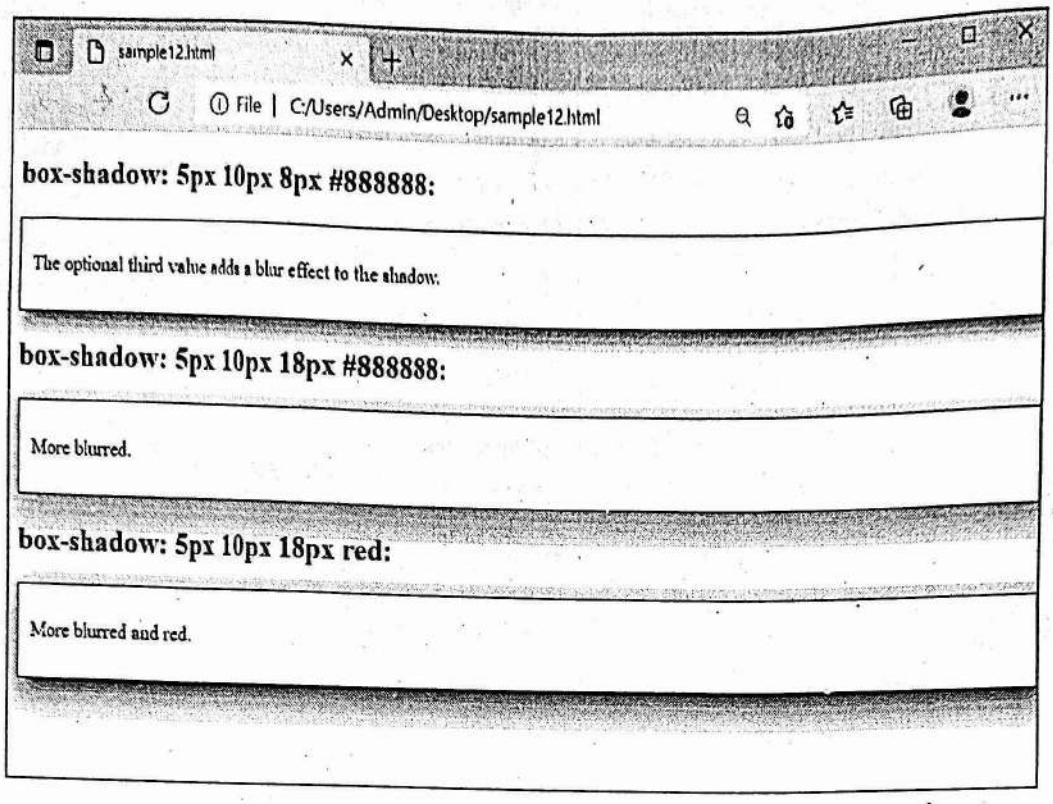


Program 1.22.3 : Add a blur effect to the shadow.

```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {
 border: 1px solid;
 padding: 10px;
 box-shadow: 5px 10px 8px #888888;
}
#example2 {
 border: 1px solid;
 padding: 10px;
 box-shadow: 5px 10px 18px #888888;
}
#example3 {
 border: 1px solid;
 padding: 10px;
 box-shadow: 5px 10px 18px red;
}
</style>
</head>
<body>
<h2>box-shadow: 5px 10px 8px #888888:</h2>
<div id="example1">
 <p>The optional third value adds a blur effect to the
shadow.</p>
</div>
<h2>box-shadow: 5px 10px 18px #888888:</h2>
<div id="example2">
 <p>More blurred.</p>
</div>

<h2>box-shadow: 5px 10px 18px red:</h2>
<div id="example3">
 <p>More blurred and red.</p>
</div>
</body>
</html>
```

NOTES

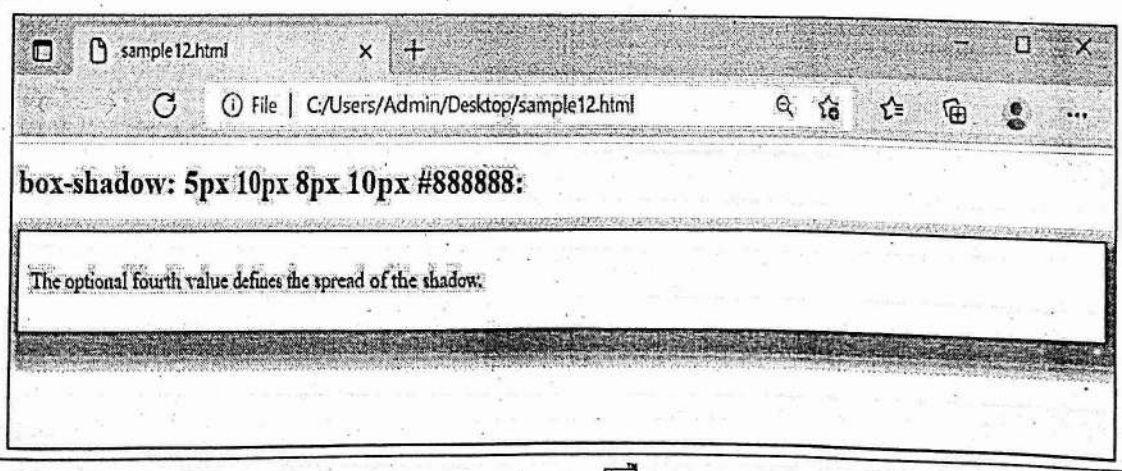
Output



Program 1.22.4 : Define the spread radius of the shadow.

```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {
 border: 1px solid;
 padding: 10px;
 box-shadow: 5px 10px 8px 10px #888888;
}
</style>
</head>
<body>
<h2>box-shadow: 5px 10px 8px 10px #888888;</h2>
<div id="example1">
  <p>The optional fourth value defines the spread of the
shadow.</p>
</div>
</body>
</html>
```

Output



1.23 TEXT FORMATTING

GQ. Write note on manipulating text in CSS with example.

Text Properties are given as below :

Property	Description
letter-spacing	Used for incrementing or decrementing the letter-spacing between characters in a text
line-break	Used to break the lines
line-height	Used for setting the height of line
tab-size	Used to specify the tab-character length
text-align	Used to specify the horizontal-text-alignment
text-indent	Used to specify the text-indentation of the first line of text block.
text-justify	Used to specify text-align is "justify"
text-transform	Used to Control the text-capitalization
white-space	Used to handle the white-space within an element
Hyphens	Used to separate the words
hanging-punctuation	Used to specify whether a punctuation character can be placed outer of the line box
overflow-wrap	Used to handle the overflow by breaking lines within words when a string is large enough and can't fit in given box.
word-break	Used to Specify the rule for line breaking in non-CJK scripts
word-spacing	Used to specify whether to increase or decrease the space between words in a text
word-wrap	Used to adjust the words so as to handle the overflow by breaking the long and unbreakable words to wrap them into next line

Text decoration properties

To decorate the text following Text Decoration Properties are used.

Property	Description
text-decoration	Used to specify the decoration applied on the text.
text-decoration-color	Used to specify the color for decorating the text.
text-decoration-line	Used to specify the line-type in a text-decoration
text-decoration-style	Used to specify the style for decorating the text.
text-shadow	Used to add shadow to text
text-underline-position	Used to specify the position of the underline set by text-decoration property.

Following are the CSS properties used to manipulate text :

1. Color

This property is used for setting the color of a text. We have studied this property in previous section.

2. Direction

- This property is used for setting the text direction.
- The values of this property are ltr or rtl means left to right or right to left respectively.

3. Letter-spacing

- This property is used to insert or remove spaces between the letters of a word.
- The values of this property are normal or a number specifying the space.

4. Word-spacing

- This property is used to insert or remove spaces between the words of a sentence.
- The values of this property are normal or a number specifying space

5. Text-indent

- This property is used to indent the text of a paragraph.
- The values of this property are % or a number specifying indent space.

6. Text-align

- This property is used to align the text of a document.
- The values of this property are left, right, center, justify

7. Text-decoration

- This property is used to decorate the text using underline, over-line, and strikethrough.
- The values of this property are none, underline, over-line, line-through, and blink.

8. Text-transform

- This property is used to set the text cases to uppercase or lowercase letters.
- The values of this property are none, capitalize, uppercase, lowercase.

9. White-space

- This property is used handling the white space present inside an element.
- The values of this property are normal, pre, nowrap.

10. Text-shadow

- This property is used to set the text shadow around a text.
- This property is supported by few browsers.
- For example the text-shadow property has following format :

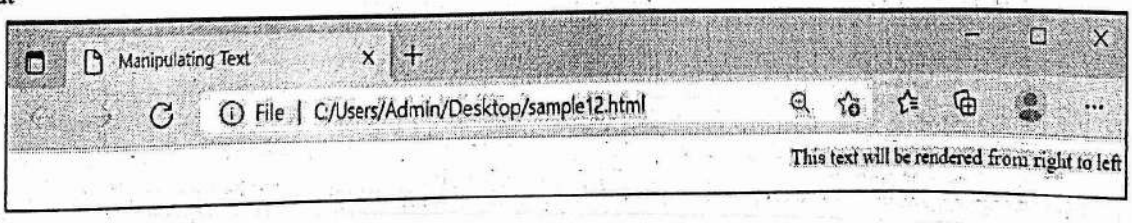
```
text-shadow:4px 4px 8px blue
```

- Where first two values indicate the position of shadow from the text, and the third value indicate the opacity of shadow. 1px indicates dark shadow. And the fourth value indicates the shadow color.

Program 1.23.1 : Write a code to demonstrate use of direction property.

```
<html>
  <head>
    <title>Manipulating Text</title>
    <style>
        p { direction: rtl; }
    </style>
  </head>
  <body>
    <p>
    This text will be rendered from right to left
    </p>
  </body>
</html>
```

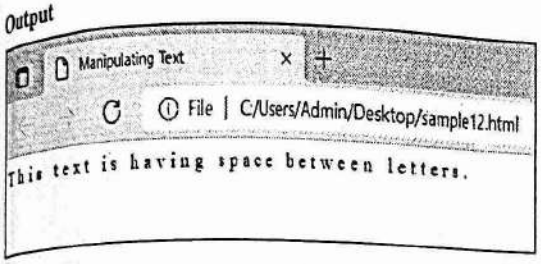
Output



Program 1.23.2 : Write a code to demonstrate the use of letter-spacing property.

```
<html>
<head>
  <title>Manipulating Text</title>
  <style>
     p { letter-spacing:5px; }
  </style>
</head>
<body>
  <p>
  This text is having space between letters.
  </p>
</body>
</html>
```

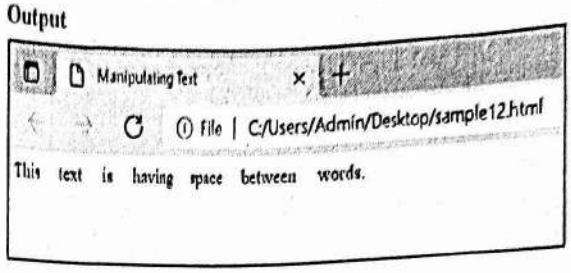
Output



Program 1.23.3 : Write a code to demonstrate the use of word-spacing property.

```
<html>
<head>
  <title>Manipulating Text</title>
  <style>
     p { word-spacing:15px; }
  </style>
</head>
<body>
  <p>
  This text is having space between words.
  </p>
</body>
</html>
```

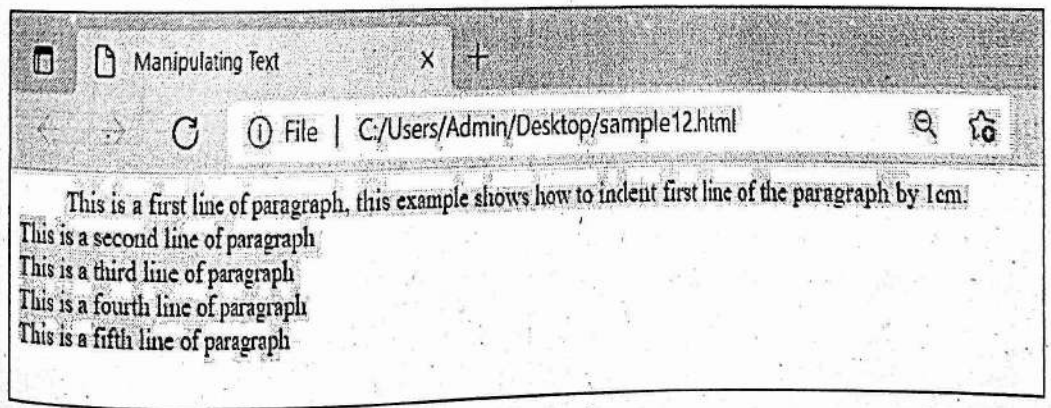
Output



Program 1.23.4 : Write a code to demonstrate the use of text-indent property.

```
<html>
<title>Manipulating Text</title>
   <head>
      <style>
      p { text-indent:1cm; }
      </style>
  </head>
  <body>
   <p >
   This is a first line of paragraph, this example shows how to
indent first line of the paragraph by 1cm. <br>
   This is a second line of paragraph <br/>
   This is a third line of paragraph <br/>
   This is a fourth line of paragraph <br/>
   This is a fifth line of paragraph <br/>
   </p>
  </body>
</html>
```

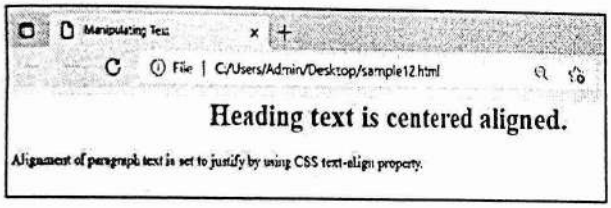
Output



Program 1.23.5 : Write a code to demonstrate the use of text-align property.

```
<html>
<title>Manipulating Text</title>
    <head>
        <style>
        h1 { text-align: center; }
        p { text-align: justify; }
        </style>
  </head>
  <body>
      <h1>
        Heading text is centered aligned.
      </h1>
    <p >
        Alignment of paragraph text is set to justify by using
CSS text-align property.
    </p>
  </body>
</html>
```

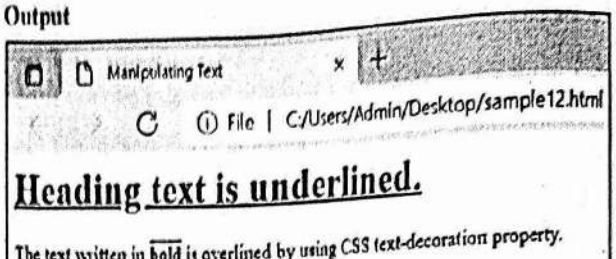
Output



Program 1.23.6 : Write a code to demonstrate the use of text-decoration property.

```
<html>
<title>Manipulating Text</title>
    <head>
        <style>
        h1 { text-decoration: underline; }
        b { text-decoration: overline; }
        </style>
  </head>
  <body>
      <h1>
        Heading text is underlined.
      </h1>
    <p >
        The text written in <b> bold</b> is overlined by
using CSS text-decoration property.
    </p>
  </body>
</html>
```

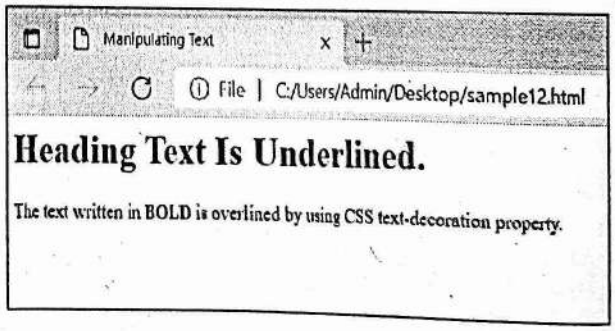
Output



Program 1.23.7 : Write a code to demonstrate the use of text-transform property.

```
<html>
<title>Manipulating Text</title>
    <head>
        <style>
        h1 { text-transform:capitalize; }
        p { text-transform:lowecase;}
        b { text-transform:uppercase; }
        </style>
  </head>
  <body>
      <h1>
        Heading text is underlined.
      </h1>
    <p >
        The text written in <b> bold</b> is overlined by
using CSS text-decoration property.
    </p>
  </body>
</html>
```

Output

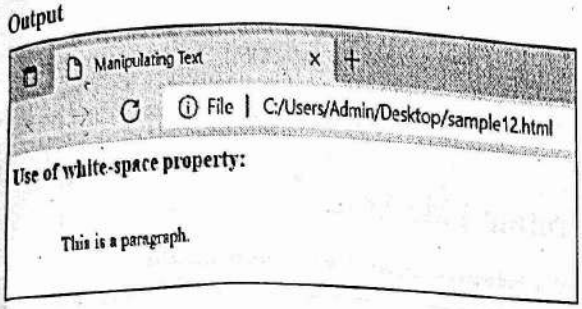


Program 1.23.8 : Write a code to demonstrate the use of white-space property.

```
<html>
<title>Manipulating Text</title>
    <head>
        <style>
        h6 { white-space:nowrap; }
        p { white-space:pre;}
        b { text-transform:uppercase; }
```

```
    </style>
  </head>
  <body>
    <h3>
      Use of white-space property:
    </h3>
  <p >
      This is a paragraph.
  </p>
  </body>
</html>
```

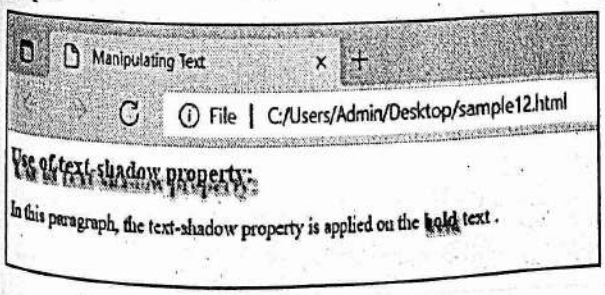
Output



Program 1.23.9 : Write a code to demonstrate the use of text-shadow property.

```
<html>
<title>Manipulating Text</title>
   <head>
      <style>

      h3 { text-shadow:4px 8px 3px red;}
      b { text-shadow:2px 4px 3px maroon;}
      </style>
  </head>
  <body>
    <h3>
      Use of text-shadow property:
    </h3>
   <p >
      In this paragraph, the text-shadow property is applied on the <b>bold</b> text .
   </p>
  </body>
</html>
```

Output



1.24 CSS3 TRANSFORMATIONS

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

1. The translate() Method

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

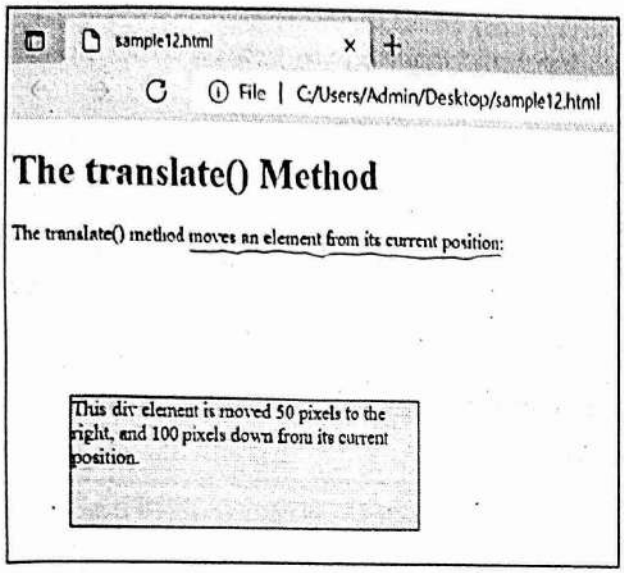
The following example moves the element 50 pixels to the right, and 100 pixels down from its current position :

Program 1.24.1

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  -ms-transform: translate(50px,100px); /* IE 9 */
  transform: translate(50px,100px); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The translate() Method</h1>
```

```
<p>The translate() method moves an element from its current position:</p>
<div>
This div element is moved 50 pixels to the right, and 100 pixels down from its current position.
</div>
</body>
</html>
```

Output



2. The rotate() Method

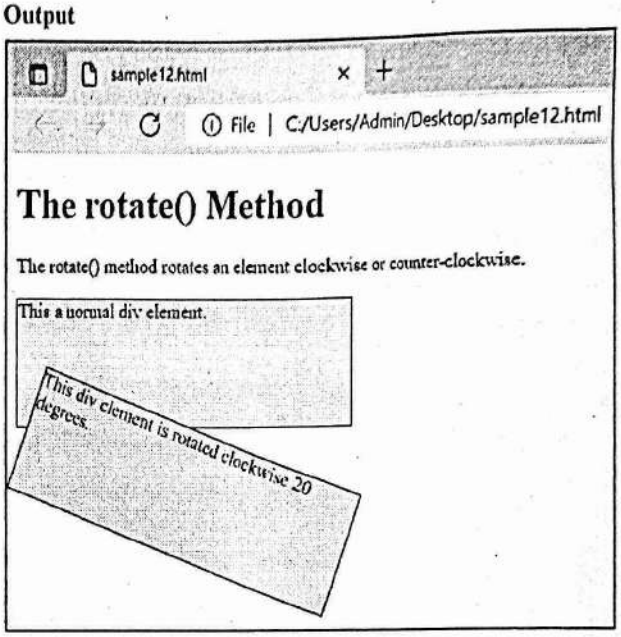
The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the element clockwise with 20 degrees :

Program 1.24.2

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}

div#myDiv {
 -ms-transform: rotate(20deg); /* IE 9 */
 transform: rotate(20deg); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The rotate() Method</h1>
<p>The rotate() method rotates an element clockwise or counter-clockwise.</p>
<div>
This a normal div element.
</div>
<div id="myDiv">
This div element is rotated clockwise 20 degrees.
</div>
</body>
</html>
```

Output



Using negative values will rotate the element counter-clockwise.

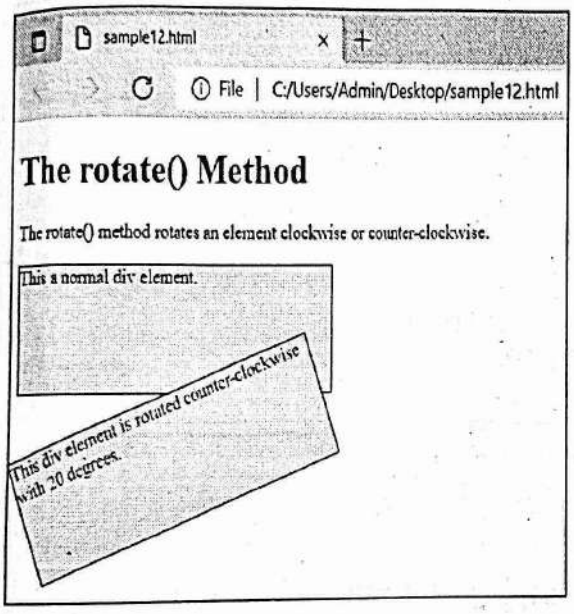
The following example rotates the element counter-clockwise with 20 degrees:

Program 1.24.3

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}
div#myDiv {
 -ms-transform: rotate(-20deg); /* IE 9 */
 transform: rotate(-20deg); /* Standard syntax */
```

```
}
</style>
</head>
<body>
<h1>The rotate() Method</h1>
<p>The rotate() method rotates an element clockwise or
counter-clockwise.</p>
<div>
This a normal div element.
</div>
<div id="myDiv">
This div element is rotated counter-clockwise with 20 degrees.
</div>
</body>
</html>
```

Output



3. The scale() Method

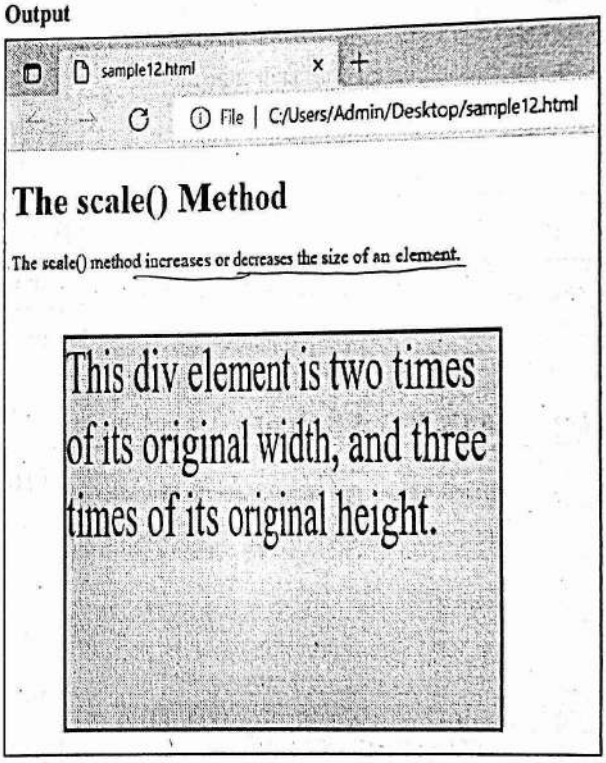
The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the element to be two times of its original width, and three times of its original height :

Program 1.24.4

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 margin: 150px;
 width: 200px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
 -ms-transform: scale(2,3); /* IE 9 */
 transform: scale(2,3); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The scale() Method</h1>
<p>The scale() method increases or decreases the size of an
element.</p>
<div>
This div element is two times of its original width, and three
times of its original height.
</div>
</body>
</html>
```

Output



4. The scaleX() Method

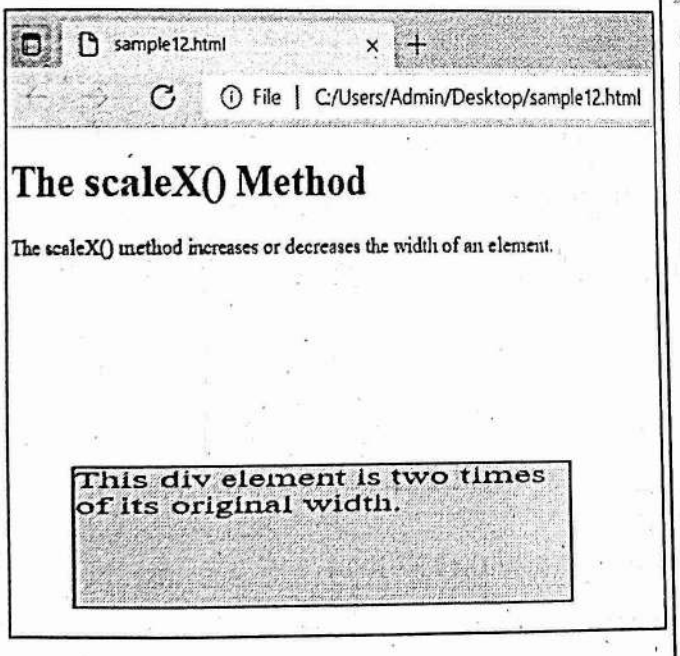
The scaleX() method increases or decreases the width of an element.

The following example increases the element to be two times of its original width :

Program 1.24.5

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 margin: 150px;
 width: 200px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
 -ms-transform: scaleX(2); /* IE 9 */
 transform: scaleX(2); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The scaleX() Method</h1>
<p>The scaleX() method increases or decreases the width of
an element.</p>
<div>
This div element is two times of its original width.
</div>
</body>
</html>
```

Output



5. The scaleY() Method

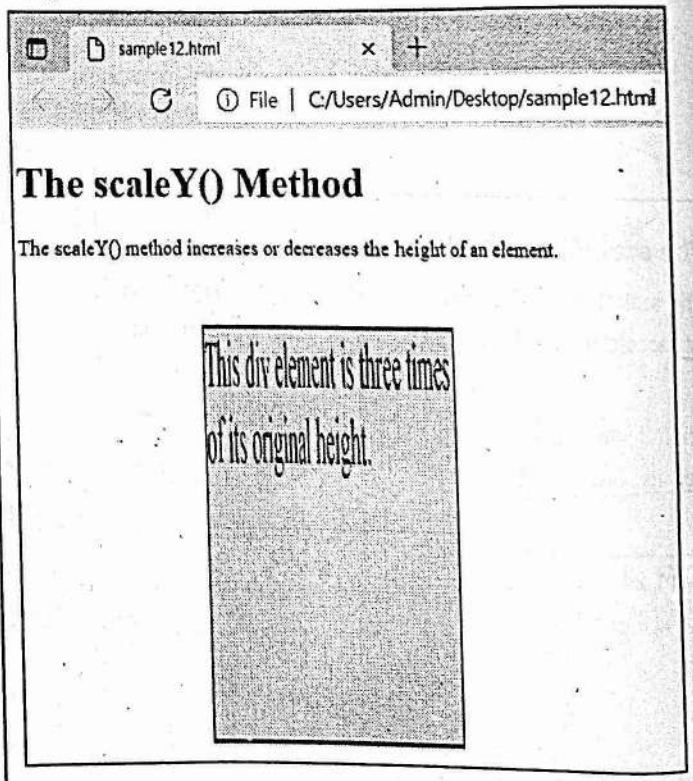
The scaleY() method increases or decreases the height of an element.

The following example increases the element to be three times of its original height.

Program 1.24.6

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 margin: 150px;
 width: 200px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
 -ms-transform: scaleY(3); /* IE 9 */
 transform: scaleY(3); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The scaleY() Method</h1>
<p>The scaleY() method increases or decreases the height of
an element.</p>
<div>
This div element is three times of its original height.
</div>
</body>
</html>
```

Output



The skew() Method

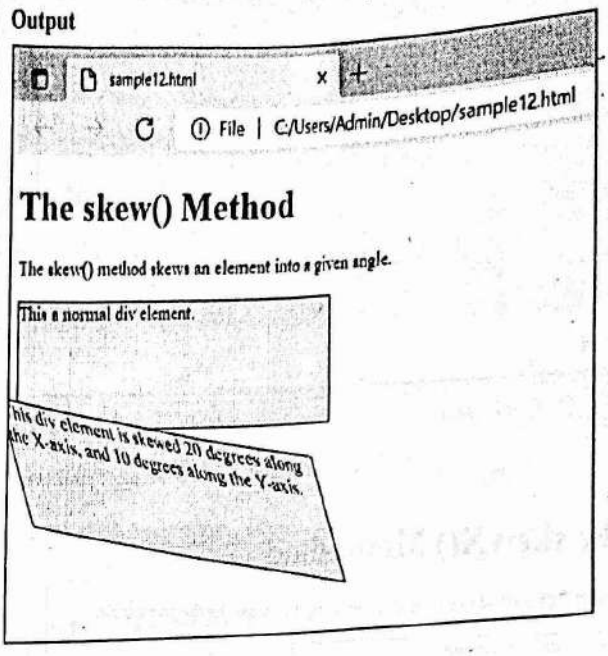
The skew() method skews an element along the X and Y-axis by the given angles.

The following example skews the element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

Program 1.24.7

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}

div#myDiv {
 -ms-transform: skew(20deg,10deg); /* IE 9 */
 transform: skew(20deg,10deg); /* Standard syntax */
}
</style>
</head>
<body>

<h1>The skew() Method</h1>
<p>The skew() method skews an element into a given angle.</p>

<div>
This a normal div element.
</div>

<div id="myDiv">
This div element is skewed 20 degrees along the X-axis, and 10 degrees along the Y-axis.
</div>

</body>
</html>
```

Output



The skewX() Method

The skewX() method skews an element along the X-axis by the given angle.

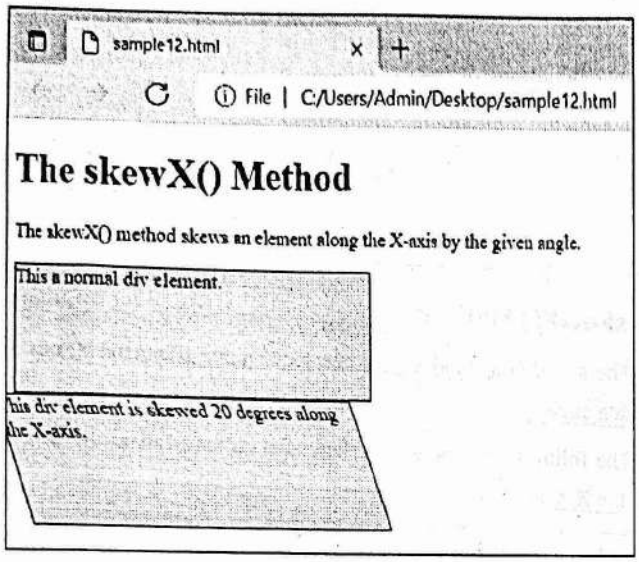
The following example skews the element 20 degrees along the X-axis:

Program 1.24.8

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}

div#myDiv {
 -ms-transform: skewX(20deg); /* IE 9 */
 transform: skewX(20deg); /* Standard syntax */
}
</style>
</head>
<body>

<h1>The skewX() Method</h1>
<p>The skewX() method skews an element along the X-axis by the given angle.</p>

<div>
```

```
This a normal div element.
</div>

<div id="myDiv">
This div element is skewed 20 degrees along the X-axis.
</div>

</body>
</html>
```

Output



The skewY() Method

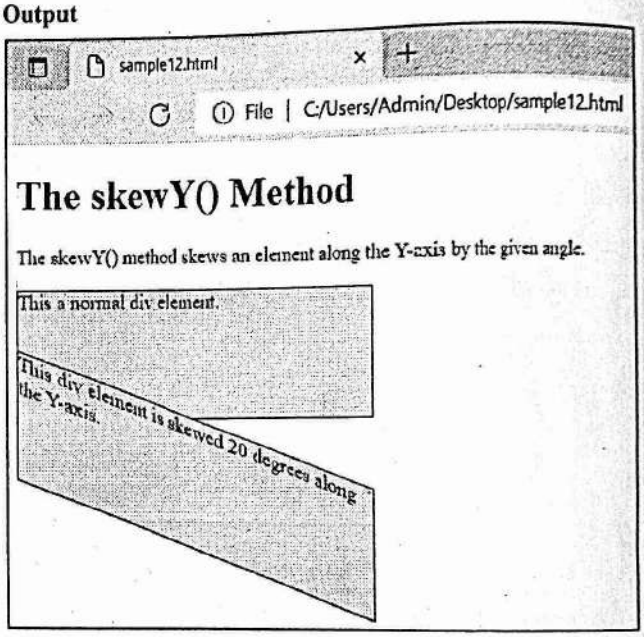
The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the element 20 degrees along the Y-axis.

Program 1.24.9

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}

div#myDiv {
 -ms-transform: skewY(20deg); /* IE 9 */
 transform: skewY(20deg); /* Standard syntax */
}
</style>
</head>
<body>

<h1>The skewY() Method</h1>
<p>The skewY() method skews an element along the Y-axis
by the given angle.</p>

<div>
This a normal div element.
</div>

<div id="myDiv">
This div element is skewed 20 degrees along the Y-axis.
</div>

</body>
</html>
```

Output



The matrix() Method

The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

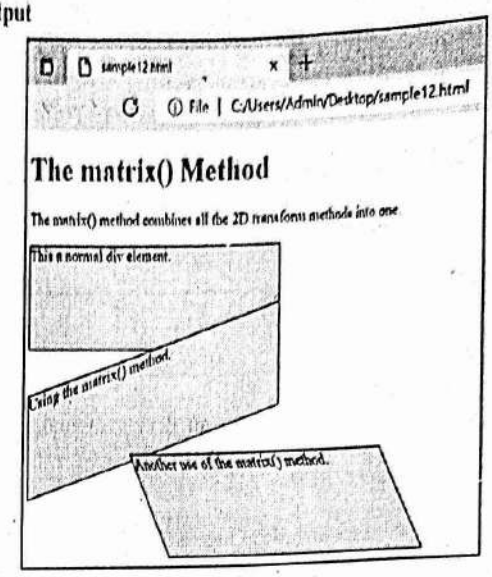
The parameters are as follow :

```
matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),
translateY())
```

Program 1.22.10

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}

div#myDiv1 {
 -ms-transform: matrix(1, -0.3, 0, 1, 0, 0); /* IE 9 */
 transform: matrix(1, -0.3, 0, 1, 0, 0); /* Standard syntax */
}

div#myDiv2 {
 -ms-transform: matrix(1, 0, 0.5, 1, 150, 0); /* IE 9 */
 transform: matrix(1, 0, 0.5, 1, 150, 0); /* Standard syntax */
}
</style>
</head>
<body>

<h1>The matrix() Method</h1>
<p>The matrix() method combines all the 2D transform
methods into one.</p>

<div>
This a normal div element.
</div>

<div id="myDiv1">
Using the matrix() method.
</div>

<div id="myDiv2">
Another use of the matrix() method.
</div>

</body>
</html>
```

Output



Transform Properties

The following table lists all the 2D transform properties.

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements

1.24.1 CSS 3D Transforms

CSS 3D Transforms Methods

With the CSS transform property you can use the following 3D transformation methods :

- rotateX()
- rotateY()
- rotateZ()

1. The rotateX() Method

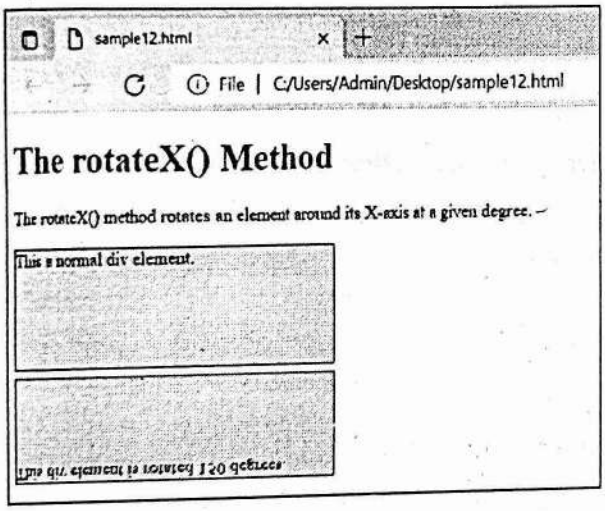
The rotateX() method rotates an element around its X-axis at a given degree:

Program 1.24.11

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
```

```
 border: 1px solid black;
}

#myDiv {
 transform: rotateX(150deg);
}
</style>
</head>
<body>

<h1>The rotateX() Method</h1>
<p>The rotateX() method rotates an element around its X-axis at a given degree.</p>

<div>
This a normal div element.
</div>

<div id="myDiv">
This div element is rotated 150 degrees.
</div>
</body>
</html>
```

Output



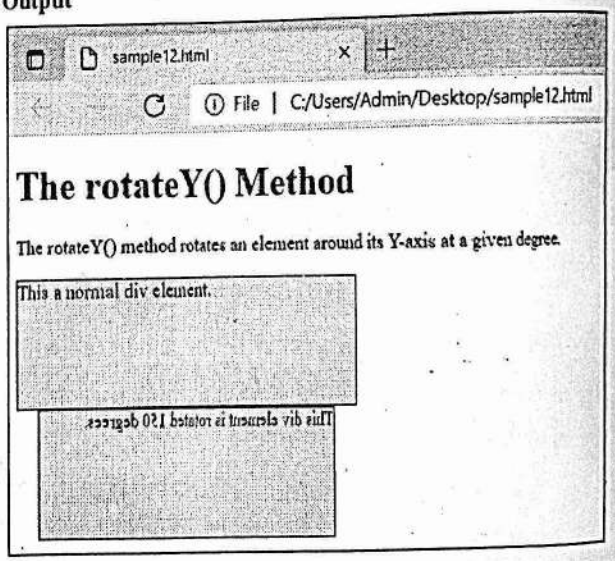
2. The rotateY() Method

The rotateY() method rotates an element around its Y-axis at a given degree.

Program 1.24.12

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}

#myDiv {
 transform: rotateY(150deg);
}
</style>
</head>
<body>

<h1>The rotateY() Method</h1>
<p>The rotateY() method rotates an element around its Y-axis at a given degree.</p>

<div>
This a normal div element.
</div>

<div id="myDiv">
This div element is rotated 150 degrees.
</div>

</body>
</html>
```

Output



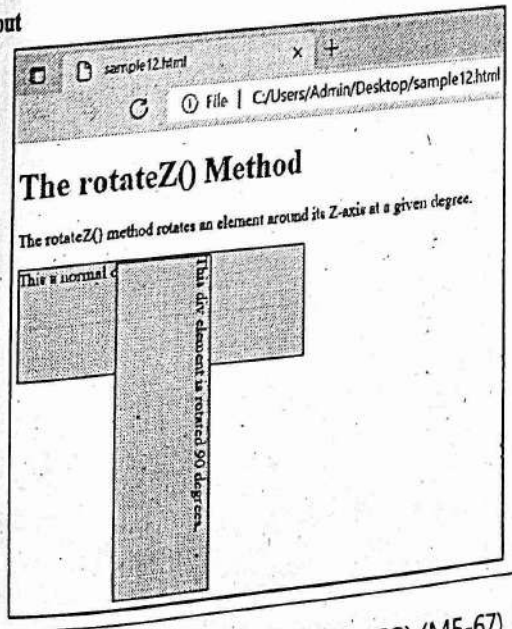
3. The rotateZ() Method

The rotateZ() method rotates an element around its Z-axis at a given degree.

Program 1.24.13

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
}

#myDiv {
 transform: rotateZ(90deg);
}
</style>
</head>
<body>
<h1>The rotateZ() Method</h1>
<p>The rotateZ() method rotates an element around its Z-axis at a given degree.</p>
<div>
This a normal div element.
</div>
<div id="myDiv">
This div element is rotated 90 degrees.
</div>
</body>
</html>
```

Output



CSS Transform Properties

The following table lists all the 3D transform properties.

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

1.25 CSS TRANSITIONS

CSS transitions allows you to change property values smoothly, over a given duration.

To create a transition effect, you must specify two things:

- The CSS property you want to add an effect to
- The duration of the effect

The following example shows a 100px * 100px red element. The element has also specified a transition effect for the width property, with a duration of 2 seconds.

Program 1.25.1

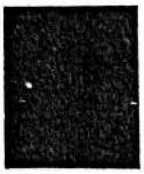
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 100px;
 background: red;
 transition: width 2s;
}
div:hover {
 width: 300px;
}
</style>
</head>
<body>
<h1>The transition Property</h1>
```

```
<p>Hover over the div element below, to see the transition
effect:</p>
<div></div>
</body>
</html>
```

Output

The transition Property

Hover over the div element below, to see the transition effect:



In the above example, when the cursor mouses out of the element, it will gradually change back to its original style.

Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

Program 1.25.2

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s, height 4s;
}
div:hover {
  width: 300px;
  height: 300px;
}
</style>
</head>
<body>
<h1>The transition Property</h1>
<p>Hover over the div element below, to see the transition
effect:</p>
<div></div>
</body>
</html>
```

Output

The transition Property

Hover over the div element below, to see the transition effect:

Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect.

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

- 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 you define your own values in a cubic-bezier function

The following example shows some of the different speed curves that can be used.

Program 1.25.3

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}
div:hover {
```

```
width: 300px;
}
</style>
</head>
<body>
<h1>The transition-timing-function Property</h1>
<p>Hover over the div elements below, to see the different speed curves:</p>
<div id="div1">linear</div><br>
<div id="div2">ease</div><br>
<div id="div3">ease-in</div><br>
<div id="div4">ease-out</div><br>
<div id="div5">ease-in-out</div><br>
</body>
</html>
```

Output

The transition-timing-function Property

Hover over the div elements below, to see the different speed curves:

Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition effect.

The following example has a 1 second delay before starting :

Program 1.25.4

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 3s;
  transition-delay: 1s;
}
div:hover {
  width: 300px;
}
</style>
</head>
<body>
<h1>The transition-delay Property</h1>
<p>Hover over the div element below, to see the transition effect:</p>
<div></div>
<p>The transition effect has a 1 second delay before starting.</p>
</body>
</html>
```

Output

The transition-delay Property

Hover over the div element below, to see the transition effect:

The transition effect has a 1 second delay before starting.

CSS Transition Properties

The following table lists all the CSS transition properties:

Property	Description
transition	A shorthand property for setting the four transition properties into a single property
transition-delay	Specifies a delay (in seconds) for the transition effect
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-property	Specifies the name of the CSS property the transition effect is for
transition-timing-function	Specifies the speed curve of the transition effect

Program 1.25.5

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 100px;
 background: red;
 transition-property: width;
 transition-duration: 2s;
 transition-timing-function: linear;
 transition-delay: 1s;
}

div:hover {
 width: 300px;
}
</style>
</head>
<body>
<h1>The transition Properties Specified One by One</h1>

<p>Hover over the div element below, to see the transition effect:</p>
<div></div>
<p> The transition effect has a 1 second delay before starting.</p>
</body>
</html>
```

Output

The transition Properties Specified One by One

Hover over the div element below, to see the transition effect:

The transition effect has a 1 second delay before starting.

1.26 CSS ANIMATIONS

- CSS allows animation of HTML elements without using JavaScript or Flash.
- An animation lets an element gradually change from one style to another.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties :

Property	Description
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties
animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long time an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies the name of the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of the animation

The @keyframes Rule

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

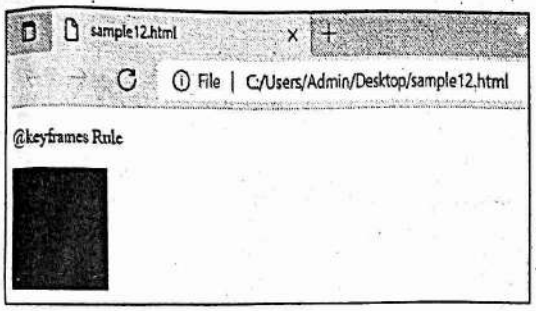
The following example binds the "example" animation to the element. The animation will last for 4 seconds, and it will gradually change the background-color of the element from "red" to "yellow".

Program 1.26.1

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

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

}
@keyframes example {
 from {background-color: red;}
 to {background-color: yellow;}
}
</style>
</head>
<body>
<p>@keyframes Rule</p>
<div></div>
</body>
</html>
```

Output



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

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

Delay an Animation

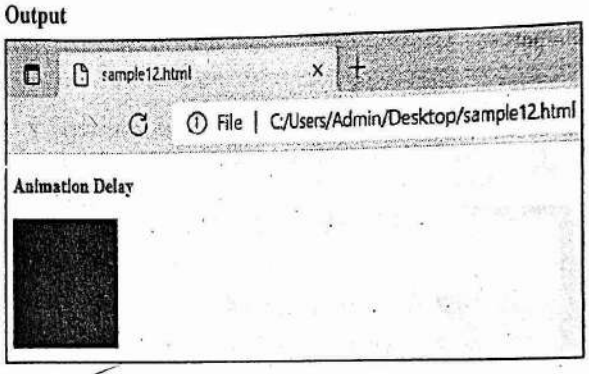
The animation-delay property specifies a delay for the start of an animation.

The following example has a 2 seconds delay before starting the animation :

Program 1.26.2

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 100px;
 background-color: red;
 position: relative;
 animation-name: example;
 animation-duration: 4s;
 animation-delay: 2s;
}
@keyframes example {
 0%  {background-color:red; left:0px; top:0px;}
 25% {background-color:yellow; left:200px; top:0px;}
 50% {background-color:blue; left:200px; top:200px;}
 75% {background-color:green; left:0px; top:200px;}
 100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
<p><b>Animation Delay</b></p>
<div></div>
</body>
</html>
```

Output



The animation-iteration-count

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

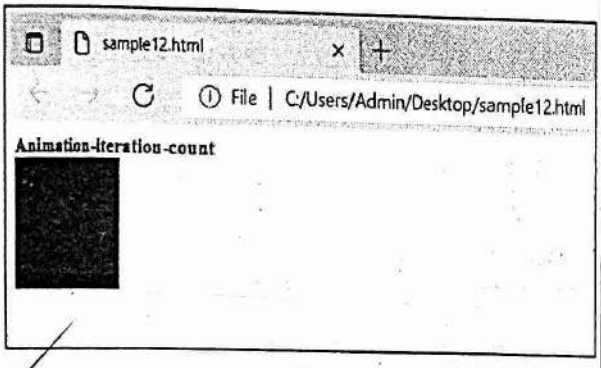
The following example will run the animation 3 times before it stops :

Program 1.26.3

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 100px;
 background-color: red;
```

```
 position: relative;
 animation-name: example;
 animation-duration: 4s;
 animation-iteration-count: 3;
}
@keyframes example {
 0%  {background-color:red; left:0px; top:0px;}
 25% {background-color:yellow; left:200px; top:0px;}
 50% {background-color:blue; left:200px; top:200px;}
 75% {background-color:green; left:0px; top:200px;}
 100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
<b>Animation-iteration-count</b>
<div></div>
</body>
</html>
```

Output



Animation direction property

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

The animation-direction property can have the following values :

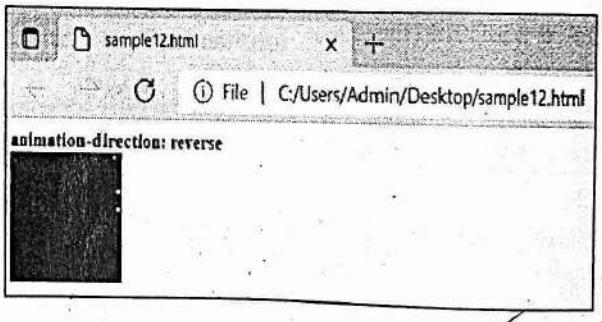
- **normal** - The animation is played as normal (forwards). This is default.
- **reverse** - The animation is played in reverse direction (backwards).
- **alternate** - The animation is played forwards first, then backwards.
- **alternate-reverse** - The animation is played backwards first, then forwards.

The following example will run the animation in reverse direction (backwards) :

Program 1.26.4

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 100px;
 background-color: red;
 position: relative;
 animation-name: example;
 animation-duration: 4s;
 animation-direction: reverse;
}
@keyframes example {
 0%  {background-color:red; left:0px; top:0px;}
 25% {background-color:yellow; left:200px; top:0px;}
 50% {background-color:blue; left:200px; top:200px;}
 75% {background-color:green; left:0px; top:200px;}
 100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
<b>animation-direction: reverse</b>
<div></div>
</body>
</html>
```

Output



Animation-timing-function property

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

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

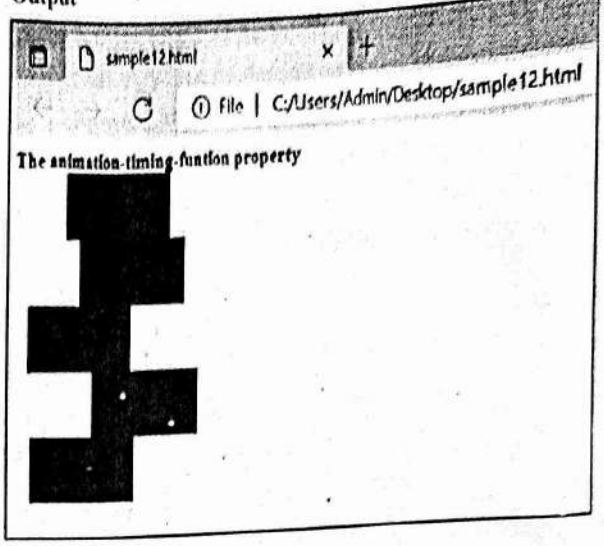
- **ease** - Specifies an animation with a slow start, then fast, then end slowly (this is default).
- **linear** - Specifies an animation with the same speed from start to end.
- **ease-in** - Specifies an animation with a slow start.
- **ease-out** - Specifies an animation with a slow end.
- **ease-in-out** - Specifies an animation with a slow start and end
- **cubic-bezier(n,n,n,n)** - Lets you define your own values in a cubic-bezier function.

The following example shows some of the different speed curves that can be used :

Program 1.26.5

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 50px;
 background-color: red;
 font-weight: bold;
 position: relative;
 animation: mymove 5s infinite;
}
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
@keyframes mymove {
 from {left: 0px;}
 to {left: 300px;}
}
</style>
</head>
<body>
<b>The animation-timing-funtion property </b>
<div id="div1">linear</div>
<div id="div2">ease</div>
<div id="div3">ease-in</div>
<div id="div4">ease-out</div>
<div id="div5">ease-in-out</div>
</body>
</html>
```

Output



Animation-fill-mode property

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

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

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

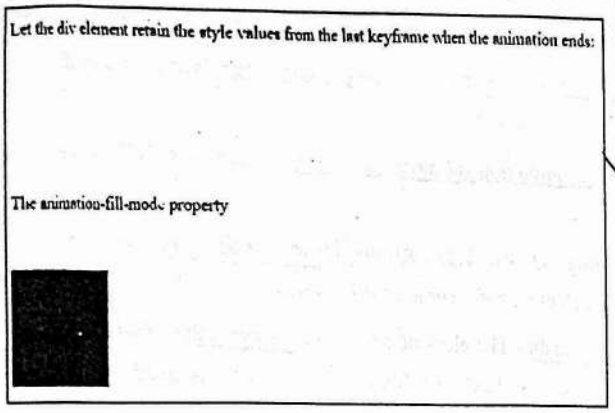
The following example lets the element retain the style values from the last keyframe when the animation ends :

Program 1.26.6

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 100px;
 background: red;
 position: relative;
 animation-name: example;
```

```
 animation-duration: 3s;
 animation-fill-mode: forwards;
}
@keyframes example {
 from {top: 0px;}
 to {top: 200px; background-color: blue;}
}
</style>
</head>
<body>
<p>Let the div element retain the style values from the last
keyframe when the animation ends:</p>
<div></div>
<p>The animation-fill-mode property</p>
</body>
</html>
```

Output

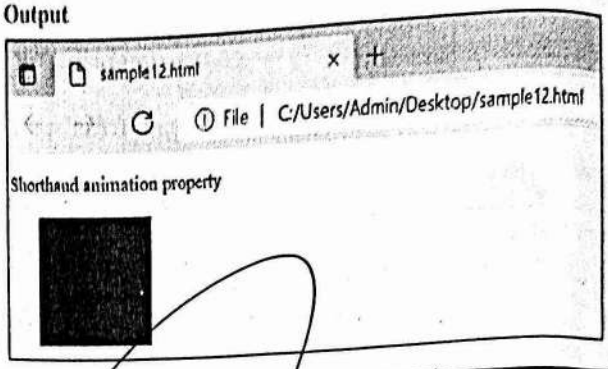


Animation Shorthand Property

Program 1.26.7

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
 height: 100px;
 background-color: red;
 position: relative;
 animation: myfirst 5s linear 2s infinite alternate;
}

@keyframes myfirst {
 0%  {background-color:red; left:0px; top:0px;}
 25% {background-color:yellow; left:200px; top:0px;}
 50% {background-color:blue; left:200px; top:200px;}
 75% {background-color:green; left:0px; top:200px;}
 100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
<p>Shorthand animation property</p>
<div></div>
</body>
</html>
```

Output



1.27 BASICS OF BOOTSTRAP

- Bootstrap is a powerful front-end framework for faster and easier web development. It includes HTML and CSS based design templates for creating common user interface components like forms, buttons, navigations, dropdowns, alerts, modals, tabs, accordions, carousels, tooltips, and so on.
- Bootstrap gives you ability to create flexible and responsive web layouts with much less efforts.
- Bootstrap was originally created by a designer and a developer at Twitter in mid-2010. Before being an open-sourced framework, Bootstrap was known as Twitter Blueprint.

What You Can Do with Bootstrap?

There are lot more things you can do with Bootstrap.

- You can easily create responsive websites.
- You can quickly create multi-column layout with pre-defined classes.
- You can quickly create different types of form layouts.
- You can quickly create different variation of navigation bar.
- You can easily create components like accordions, modals, etc. without writing any JS code.
- You can easily create dynamic tabs to manage large amount of content.
- You can easily create tooltips and popovers to show hint text.

- You can easily create carousel or image slider to showcase your content.
- You can quickly create different types of alert boxes.

1.27.1 Advantages of Bootstrap

1. **Save lots of time :** You can save lots of time and efforts using the Bootstrap predefined design templates and classes and concentrate on other development work.
2. **Responsive features :** Using Bootstrap you can easily create responsive websites that appear more appropriately on different devices and screen resolutions without any change in markup.
3. **Consistent design :** All Bootstrap components share the same design templates and styles through a central library, so the design and layout of your web pages will be consistent.
4. **Easy to use :** Bootstrap is very easy to use. Anybody with the basic working knowledge of HTML, CSS and JavaScript can start development with Bootstrap.
5. **Compatible with browsers :** Bootstrap is created with modern web browsers in mind and it is compatible with all modern browsers such as Chrome, Firefox, Safari, Internet Explorer, etc.
6. **Open Source :** It is completely free to download and use.

1.27.2 Bootstrap Containers

- Containers are the most basic layout element in Bootstrap and are required when using the grid system. Containers are basically used to wrap content with some padding. They are also used to align the content horizontally center on the page in case of fixed width layout.
- Bootstrap provides three different types containers :
  1. .container, which has a max-width at each responsive breakpoint.
  2. container-fluid, which has 100% width at all breakpoints.
  3. container-{breakpoint}, which has 100% width until the specified breakpoint.
- The following table illustrates how each container's max-width changes across each breakpoint.
- For example, when using the .container class the actual width of the container will be 100% if the viewport width is less than 576px, 540px if the viewport width is greater than or equal to 576px, 720px if the viewport width is greater than or equal to 768px, 960px if the viewport width is greater than or equal to 992px, 1140px if viewport width is greater than or equal to 1200px.
- Similarly, when you use the .container-lg class the actual width of the container will be 100% until the viewport width is less than 992px, 960px if the viewport width is greater than or equal to 992px, and 1140px if the viewport width is greater than or equal to 1200px.

Class	Extra small	Class	Extra small	Class	Extra small
.container	100%	540px	720px	960px	1140px
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px
.container-xl	100%	100%	100%	100%	1140px
.container-fluid	100%	100%	100%	100%	100%

Creating Responsive Fixed-width Containers

You can simply use the .container class to create a responsive, fixed-width container. The width of the container will change at different breakpoints or screen sizes, as shown above.

Program 1.27.1

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,
initial-scale=1">
<title>Bootstrap 4 Responsive Fixed-width
Containers</title>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/b
ootstrap.min.css">
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/boo
tstrap.bundle.min.js"></script>
<style>
  .bs-example{
    margin: 20px 0;
```

```
  }
</style>
</head>
<body>
<div class="bs-example">
  <div class="container">
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
  </div>
  <p class="m-3"> <strong>Tip:</strong> Open the output in a new blank tab (Click the arrow next to "Show Output" button) and resize the browser window to understand how the Bootstrap responsive grid system works.</p>
</div>
</body>
</html>
```

Creating Fluid Containers

You can use the .container-fluid class to create a full width container. The width of the fluid container will always be 100% irrespective of the devices or screen sizes.

Example

```
<div class="container-fluid">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>
```

Adding Background and Borders to Containers

By default, container doesn't have any background-color or border. But if you need you can apply your own styles, or simply use the Bootstrap background-color and border utility classes to add background-color or border on them, as shown in the following example.

Example

```
<!-- Container with dark background and white text color -->
<div class="container bg-dark text-white">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>
<!-- Container with light background -->
<div class="container bg-light">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>

<!-- Container with border -->
<div class="container border">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>
```

Applying Paddings and Margins to Containers

By default, containers have padding of 15px on the left and right sides, and no padding on the top and bottom sides.

Example

```
<!-- Container with border, extra paddings and margins -->
<div class="container border py-3 my-3">
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</div>
```

1.27.3 Bootstrap Grid System

GQ. What is Bootstrap Grid System?

Bootstrap grid system provides the quick and convenient way to create responsive website layouts.

Creating Two Column Layouts

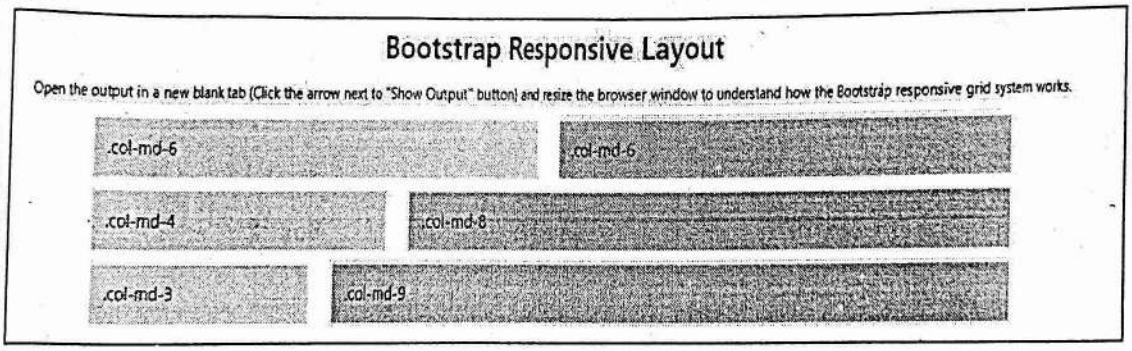
The following example will show you how to create two column layouts for medium, large and extra-large devices like tables, laptops and desktops etc. However, on mobile phones (screen width less than 768px), the columns will automatically become horizontal (2 rows, 1 column).

Program 1.27.2

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Bootstrap 4 Two Column Grid Layouts for Tablets and Desktops</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css">
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"></script>
<style>
  /* Some custom styles to beautify this example */
  .demo-content{
    padding: 15px;
    font-size: 18px;
    background: #dbdfe5;
    margin-bottom: 15px;
```

```
  }
  .demo-content.bg-alt{
    background: #abb1b8;
  }
</style>
</head>
<body>
  <h2 class="text-center my-3">Bootstrap Responsive Layout</h2>
    <div class="text-center">Open the output in a new blank tab (Click the arrow next to "Show Output" button) and resize the browser window to understand how the Bootstrap responsive grid system works.</div>
  <div class="container mt-3">
    <!--Row with two equal columns-->
    <div class="row">
      <div class="col-md-6">
        <div class="demo-content">.col-md-6</div>
      </div>
      <div class="col-md-6">
        <div class="demo-content bg-alt">.col-md-6</div>
      </div>
    </div>
    <!--Row with two columns divided in 1:2 ratio-->
    <div class="row">
      <div class="col-md-4">
        <div class="demo-content">.col-md-4</div>
      </div>
      <div class="col-md-8">
        <div class="demo-content bg-alt">.col-md-8</div>
      </div>
    </div>

    <!--Row with two columns divided in 1:3 ratio-->
    <div class="row">
      <div class="col-md-3">
        <div class="demo-content">.col-md-3</div>
      </div>
      <div class="col-md-9">
        <div class="demo-content bg-alt">.col-md-9</div>
      </div>
    </div>
  </div>
</body>
</html>
```

Output



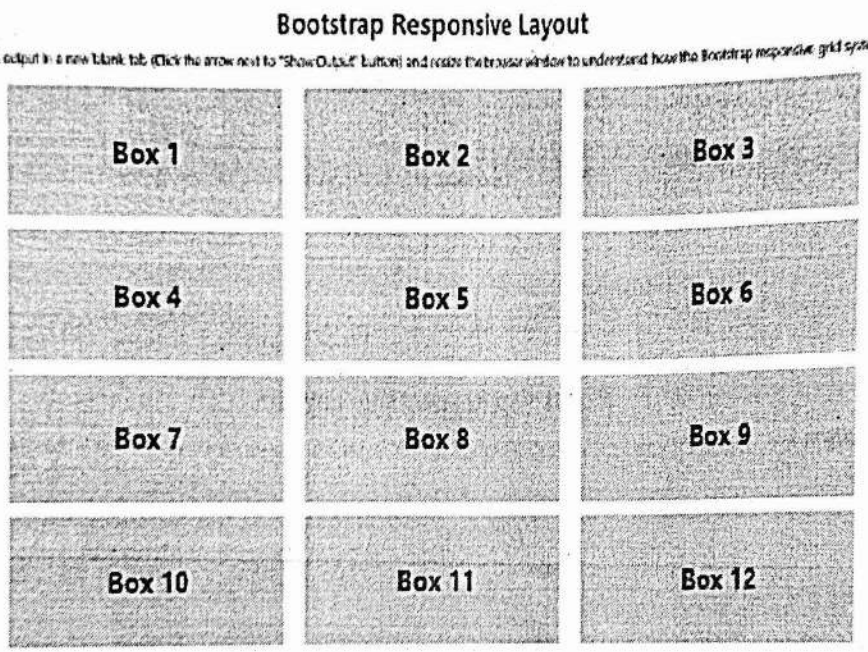
Creating Multi-Column Layouts with Bootstrap

Program 1.27.3

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Bootstrap 4 Grid Layouts for Large Devices</title>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css">
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"></script>
<style>
  /* Some custom styles to beautify this example */
  p{
    padding: 50px;
    font-size: 32px;
    font-weight: bold;
    text-align: center;
    background: #dbdfe5;
```

```
  }
</style>
</head>
<body>
  <h2 class="text-center my-3">Bootstrap Responsive Layout</h2>
    <div class="text-center">Open the output in a new blank tab (Click the arrow next to "Show Output" button) and resize the browser window to understand how the Bootstrap responsive grid system works.</div>
  <div class="container mt-3">
    <div class="row">
      <div class="col-lg-4"><p>Box 1</p></div>
      <div class="col-lg-4"><p>Box 2</p></div>
      <div class="col-lg-4"><p>Box 3</p></div>
      <div class="col-lg-4"><p>Box 4</p></div>
      <div class="col-lg-4"><p>Box 5</p></div>
      <div class="col-lg-4"><p>Box 6</p></div>
      <div class="col-lg-4"><p>Box 7</p></div>
      <div class="col-lg-4"><p>Box 8</p></div>
      <div class="col-lg-4"><p>Box 9</p></div>
      <div class="col-lg-4"><p>Box 10</p></div>
      <div class="col-lg-4"><p>Box 11</p></div>
      <div class="col-lg-4"><p>Box 12</p></div>
    </div>
  </div>
</body>
</html>
```

Output



1.28 MULTIPLE CHOICE QUESTIONS

Q. 1.1 HTTP is a _____.

(a) State Oriented (b) Stateless

(c) (a) and (b) both (d) None of Above ✓**Ans. : (b)**

Q. 1.2 HTTP is _____.

(a) Request-Response Protocol

(b) Stateful

(c) Secured

(d) All of the above ✓**Ans. : (a)**

Q. 1.3 A _____ is a globally unique name for every document that can be accessed on the Web

(a) ID (b) UCL

(c) URL (d) UML ✓**Ans. : (c)**

Q. 1.4 What is the full form of HTTP?

(a) Hyper Text Trial Program

(b) Hyper Text Transfer Protocol

(c) Hyper Text Technology Protocol

(d) Hyper Text Trial Protocol ✓**Ans. : (b)**

Q. 1.5 The first line of HTTP request message is called _____

(a) Request line (b) Header line

(c) Status line (d) Entity line ✓**Ans. : (a)**

Q. 1.6 Which of the following is present in both an HTTP request line and a status line?

(a) HTTP version number (b) URL
(c) Method (d) None of the mentioned

✓Ans. : (a)

Q. 1.7 URL is _____.

(a) web address (b) source code
(c) user's address (d) an attribute ✓Ans. : (a)

Q. 1.8 What is the correct syntax of web address?

(a) path://prefix.port:domain/filename/scheme
(b) scheme://prefix.domain:port/path/filename
(c) prefix://scheme.port:domain/filename/path
(d) port://domain.filenmae:path/scheme/prefix

✓Ans. : (b)

Q. 1.9 Which scheme is used for secure HyperText Transfer Protocol?

(a) ftp (b) http (c) https (d) file

✓Ans. : (c)

Q. 1.10 In a URL, the _____ is the client-server program used to retrieve the document.

(a) path (b) protocol
(c) host (d) None of the above ✓Ans. : (b)

Q. 1.11 The ____ translates internet domain and host names to IP address.

(a) domain name system
(b) routing information protocol
(c) network time protocol
(d) internet relay chat ✓Ans. : (a)

Q. 1.12 A program in HTML can be rendered and read by -

(a) Web browser (b) Server
(c) Interpreter (d) None of the above ✓Ans. : (a)

Q. 1.13 The correct sequence of HTML tags for starting a webpage is -

(a) Head, Title, HTML, body
(b) HTML, Body, Title, Head
(c) HTML, Head, Title, Body
(d) HTML, Head, Title, Body ✓Ans. : (d)

Q. 1.14 Which is the correct way to comment out something in HTML?

(a) Using ## and # (b) Using <!-- and -->
(c) Using </-- and -/-> (d) Using <!-- and -!>

✓Ans. : (b)

Q. 1.15 Which HTML tag is used to display the power in expression, i.e., (x2 - y2)?

(a) <sup> (b) <sub>
(c) <p> (d) None of the above ✓Ans. : (a)

Q. 1.16 Which of the following element is responsible for making the text bold in HTML?

(a) <pre> (b) <a>
(c) <b> (d)
 ✓Ans. : (c)

Q. 1.17 How to create an unordered list (a list with the list items in bullets) in HTML?

(a) <ul> (b) <ol> (c) <li> (d) <i> ✓Ans. : (a)

Q. 1.18 Which character is used to represent the closing of a tag in HTML?

(a) \ (b) ! (c) / (d) . ✓Ans. : (c)

Q. 1.19 How to insert an image in HTML?

(a) <img href = "jtp.png" />
(b) <img url = "jtp.png" />
(c) <img link = "jtp.png" />
(d) <img src = "jtp.png" />

✓Ans. : (d)

Q. 1.20 <input> is -

(a) a format tag. (b) an empty tag.
(c) All of the above (d) None of the above ✓Ans. : (b)

Q. 1.21 Which of the following tag is used to add rows in the table?

(a) <td> and </td> (b) <th> and </th>
(c) <tr> and </tr> (d) None of the above ✓Ans. : (c)

Q. 1.22 What are the types of unordered or bulleted list in HTML?

(a) disc, square, triangle (b) polygon, triangle, circle
(c) disc, circle, square (d) All of the above

✓Ans. : (c)

Q. 1.23 The following elements <header>, <footer>, <article>, <section> are the new elements in HTML5. These elements are called,

(a) Control attributes (b) Semantic elements
(c) Graphic elements (d) Multimedia elements

✓Ans. : (b)

Q. 1.24 What is the new form elements introduced in HTML5?

1) <button> 2) <optgroup>
3) <datalist> 4) <keygen> 5) <output>

(a) 1, 2 and 3 (b) 2, 3 and 4
(c) 3, 4 and 5 (d) 4, 5 and 1 ✓Ans. : (c)

Q. 1.25 Which among the following tags define a caption for a <fieldset> element?

(a) <select> (b) <label>
(c) <legend> (d) <textarea> ✓Ans. : (c)

Q. 1.26 Which feature was already introduced before HTML5?

(a) Canvas/SVG (b) Video
(c) Geolocation (d) Frames ✓Ans. : (d)

Q. 1.27 What is the correct syntax of doctype in HTML5?

(a) <!doctype html> (b) <doctype html!>
(c) <doctype html> (d) </doctype html>
✓Ans. : (a)

Q. 1.28 Which of the following is not semantic element for text in HTML5?

(a) <mark> (b) <time>
(c) <wbr> (d) <article> ✓Ans. : (d)

Q. 1.29 Which of the following defines the title of a work?

(a) <cite> (b) <hr>
(c) <a> (d) <address> ✓Ans. : (a)

Q. 1.30 CSS stands for _____.

(a) Cascade style sheets (b) Color and style sheets
(c) Cascading style sheets (d) None of the above
✓Ans. : (c)

Q. 1.31 Which of the following is the correct syntax for referring the external style sheet?

(a) <style src = example.css>
(b) <style src = "example.css" >
(c) <stylesheet> example.css </stylesheet>
(d) <link rel="stylesheet" type="text/css" href="example.css">
✓Ans. : (d)

Q. 1.32 The property in CSS used to change the background color of an element is _____

(a) bgcolor (b) color
(c) background-color (d) All of the above
✓Ans. : (c)

Q. 1.33 The CSS property used to control the element's font-size is _____.

(a) text-style (b) text-size
(c) font-size (d) None of the above ✓Ans. : (c)

Q. 1.34 The HTML attribute used to define the inline styles is -

(a) style (b) styles
(c) class (d) None of the above ✓Ans. : (a)

Q. 1.35 The HTML attribute used to define the internal stylesheet is -

(a) <style> (b) style
(c) <link> (d) <script> ✓Ans. : (a)

Q. 1.36 Which of the following CSS property is used to set the background image of an element?

(a) background-attachment
(b) background-image
(c) background-color
(d) None of the above ✓Ans. : (b)

Q. 1.37 Which of the following is the correct syntax to make the background-color of all paragraph elements to yellow?

(a) p {background-color : yellow;}
(b) p {background-color : #yellow;}
(c) all {background-color : yellow;}
(d) all p {background-color : #yellow;} ✓Ans. : (a)

Q. 1.38 Which of the following is the correct syntax to display the hyperlinks without any underline?

(a) a {text-decoration : underline;}
(b) a {decoration : no-underline;}
(c) a {text-decoration : none;}
(d) None of the above ✓Ans. : (c)

Q. 1.39 Which of the following CSS property is used to add shadows to the text?

(a) text-shadow (b) text-stroke
(c) text-overflow (d) text-decoration ✓Ans. : (a)

Q. 1.40 Which of the following CSS property is used to specify whether the table cells share the common or separate border?

(a) border-collapse (b) border-radius
(c) border-spacing (d) None of the above ✓Ans. : (a)

Q. 1.41 Which of the following CSS property specifies the origin of the background-image?

(a) background-origin (b) background-attachment
(c) background-size (d) None of the above
✓Ans. : (a)

Chapter Ends...

□□□

IP

MODULE 2

CHAPTER 2

Front End Development

Syllabus

JavaScript : 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.

2.1 Introductions to JavaScript 2-3

2.1.1 Characteristics or Advantages of JavaScript 2-3

2.1.2 Advantages and Disadvantages of Client Side Scripting 2-3

2.1.3 Differences between Client Side and Server Side Scripting Language 2-4

2.1.4 Using JavaScript in an HTML Document 2-4

2.1.5 JavaScript Example 2-5

2.2 Programming Fundamentals of JavaScript : Variables, Operators 2-6

2.2.1 Variables 2-6

2.2.2 Operators 2-6

2.3 Control Flow Statements 2-9

2.3.1 Conditional Statements 2-9

2.3.2 Loop Statements 2-14

2.4 Popup Boxes 2-17

2.5 Functions 2-17

2.5.1 Function Definition 2-17

2.5.2 Defining Function Arguments 2-18

2.5.3 Defining a Return Statement 2-19

2.6 JavaScript Objects 2-19

2.6.1 Creating Objects in JavaScript 2-19

2.6.2 String Object 2-20

2.6.3 RegExp Object 2-22

2.6.4 Math Object 2-22

2.6.5 Date Object 2-23
2.6.6 Number Object 2-23
2.6.7 Array Object 2-23
2.7 Browser Objects - Window, Navigator, History, Location, Document 2-24
2.7.1 Window Object 2-24
2.7.2 Navigator Object 2-24
2.7.3 History Object 2-25
2.7.4 Location Object 2-26
2.8 Document Object Model 2-27
2.8.1 DOM Levels 2-27
2.8.2 Properties and Methods of Document Object 2-28
2.8.3 Manipulation using DOM 2-31
2.9 Events Handling 2-33
2.10 Form Validation using JavaScript 2-38
2.11 Exception Handling in JavaScript 2-42
2.12 DHTML JavaScript 2-44
2.13 JSON Introduction 2-46
2.13.1 Syntax Rules 2-48
2.13.2 JSON Data Types 2-48
2.13.3 JSON Object 2-48
2.13.4 JSON Function Files 2-50
2.13.5 JSON Http Request 2-51
2.13.6 JSON SQL 2-52
2.14 Multiple Choice Questions 2-53
- **Chapter Ends** 2-58

2.1 INTRODUCTIONS TO JAVASCRIPT

- JavaScript is a light-weight object-oriented programming language which is used by several websites for scripting the webpages.
- JavaScript was developed by Mr. Brendan Eich in 1995 who was working in Netscape.
- JavaScript is not a compiled language, but it is a translated language.
- The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.
- It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.
- JavaScript helps to make our webpage more lively and interactive. JavaScript is widely used in mobile application development as well as in game development.
- JavaScript was initially called as LiveScript and later on the name is changed to JavaScript.
- With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Features of JavaScript

The features of JavaScript are as follows:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

Applications of JavaScript

JavaScript is used to create interactive websites. It is mainly used for :

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks, etc.

Module 2

2.1.1 Characteristics or Advantages of JavaScript

GQ. Explain characteristics of JavaScript.

- JavaScript is a lightweight, interpreted client-side scripting language.
- Designed for developing network-based applications.
- JavaScript is complementary to Java.
- JavaScript is complementary to and integrated with HTML.
- It is open source and cross-platform.
- The user input is validated before sending the page to the server. This minimizes the server traffic, which tends to fewer loads on the server.
- There is no need for the user to wait to see if something has been forgotten to enter.
- Interactive interfaces can be created, which can give responses to end user actions like mouse or keyboard activities.
- JavaScript can include elements like drag-drop components and sliders to provide a feel of rich interface to the users.

2.1.2 Advantages and Disadvantages of Client- Side Scripting

GQ. What are the advantages and disadvantages of client side scripting?

Advantages

1. Immediate response to a user's actions, which enables more interactivity.
2. No need to go to the server; hence, execution is fast.
3. Improve the usability of web sites for users whose browsers support scripts.

4. Developers get more control over the look and behaviour of their web widgets.
5. Possible to substitute by HTML if a user's browsers do not support scripts.
6. Are reusable and obtainable from various types of free resources.

Disadvantages

1. Scripts are not supported by all of the browsers; hence, errors might occur if no alternatives have been provided.
2. There is need of more quality assurance testing as different browsers and browser versions support scripts differently.
3. May need more time and effort for development if the scripts are not already available through other resources.
4. Sometimes, the web widget looks like a standard control, but their behaviour may be different or vice-versa, which may lead to usability problems.

2.1.3 Differences between Client Side and Server Side Scripting Language

GQ. What are differences between client side and server side scripting language?

Parameters	Client Side Scripting Language	Server Side Scripting Language
Execution	The client side script is executed by web browser which is located at the user's computer.	The server side script is executed by the Web Server that outputs the page which is to be sent to the browser.
Database	Client side scripting language cannot connect to the databases which is located on the web server.	Server side scripting language can connect and access to the databases which is located on the web server.
File System	Client side scripting language cannot have access to the file system which is located on the web server.	Server side scripting language has access to the file system which is located on the web server.
Access to Setting	Client side scripting language can access the files and settings that are local at the user's computer.	Server side scripting language cannot access the settings that belong to web server.
Block	User can block the Client side scripting language	User cannot block the Server side scripting language
Response	Response from a client-side script is quick since the scripts are processed on the local computer.	Response from a server-side script is slow since the scripts are processed on the remote computer.
Examples	JavaScript, VBScript, etc.	PHP, JSP, ASP, ASP.Net, Ruby, Perl, etc

2.1.4 Using JavaScript in an HTML Document

- The code (script) of JavaScript is written in the script opening <script> and closing </script> HTML tags in a web page. Usually, the <script> tag is allowed anywhere in the html page, but the <head> section is normally recommended.
- The <script> tag basically notifies the browser that the code written is a script of JavaScript.

Syntax

Syntax of JavaScript will appear as follows :

```
<script ...>
  JavaScript statements;
</script>
```

- The <script> tag has two important attributes which basically serves the same purpose.
- **Language and Type :** These attributes specify the name of scripting language used. Here its value will be "JavaScript" for attribute *Language* and "text/JavaScript" for attribute *Type*.
- JavaScript segment will look like :

```
<script language="JavaScript">
  JavaScript statements
</script>
```

Or

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

2.1.5 JavaScript Example

1. Embedded JavaScript

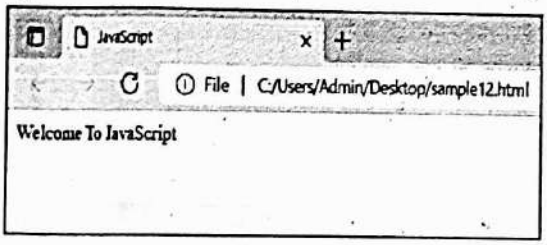
The JavaScript code can be embed into a HTML file using the <script> tag.

```
<!DOCTYPE html>
<html>
<head>
<title>
JavaScript
</title>
<script language="JavaScript">
document.write("Welcome To JavaScript");
</script>
</head>
<body>
</body>
</html>
```

document.write :

This method writes a string into HTML document.

Output

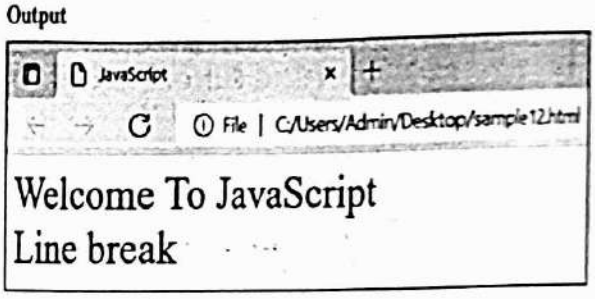


In the above program, outside the <script> element, the HTML tags can be used in their normal way, but if we want to use the HTML tags in <script> section, then document.write() method is used.

➢ **Program 2.1.1 :** Write a program to demonstrate line break example.

```
<html>
<head>
<title>
JavaScript
</title>
</head>
<body>
<font size=6 color="blue">
<script type="text/JavaScript">
document.write("Welcome To JavaScript");
document.write("<br>Line break");
</script>
</font>
</body>
</html>
```

Output



2. External JavaScript

GQ. Write note on External JavaScript with an example.

- An external JavaScript file can be created to embed it in many html pages.
- It supports the concept of code reusability as single JavaScript file can be embed into several html pages. An external JavaScript file is saved by the extension ".js".

Example : External JavaScript code demonstrating welcome message

MyFile.js

```
function msg()
{
    alert("Welcome To The World of Web");
}
```

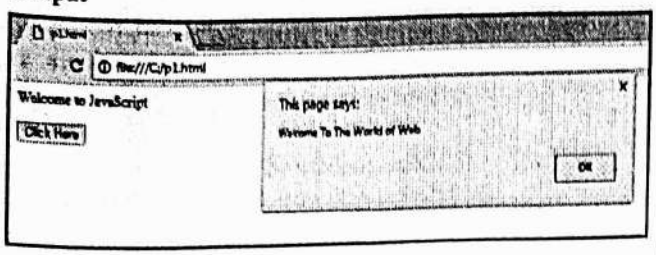
HTML page

```
<html>
<head>
<script language="JavaScript" src="MyFile.js">
</script>
</head>

<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="Click Here"
onclick="msg()"/>
</form>
</body>
```

Module 2

```
</html>
```

Output



2.2 PROGRAMMING FUNDAMENTALS OF JAVASCRIPT : VARIABLES, OPERATORS

2.2.1 Variables

- Variable is a name given to memory location where we can store some value. The value depends upon the data type of the variable.
- In JavaScript there are number of data types used to store different types of values. These data types are primarily categorized as :

1. JavaScript primitive data types

In JavaScript, there are five types of primitive data types as follows :

Sr. No.	Data Type	Description
1.	String	Represents sequence of characters e.g. "Snehal".
2.	Number	Represents numeric values e.g. 13.
3.	Boolean	Represents Boolean value either true or false.
4.	Undefined	Represents undefined value.
5.	Null	Represents null means no value at all.

2. JavaScript non-primitive data types

Sr. No.	Data Type	Description
1.	Object	Represents instance which helps to access members.
2.	Array	Represents set of same values.
3.	RegExp	Represents regular expression.

- JavaScript is considered as a **dynamic type language** that is there is no need to specify type of the variable. This type is dynamically decided by the JavaScript engine. While declaring a variable, "var" keyword is used on place of data type. Var means variant, that is the variable can store any type of value like numbers, strings, dates etc.

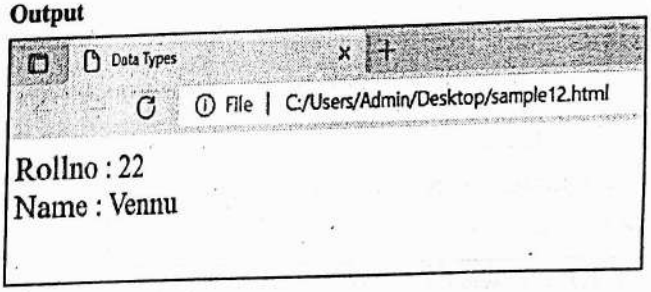
Examples

1. var rno = 101; //holding number
2. var sname="Shraddha"; //holding string

Program 2.2.1 : To display roll no. and name of student using "var" keyword.

```
<html>
<head>
<title>
Data Types
</title>
</head>
<body>
<script language="JavaScript">
document.write("<font size=5 color=blue>");
var rno = 22;
var sname = "Vennu";
document.write("Rollno : "+rno);
document.write("<br>Name : "+sname);
document.write("</font>");
</script>
</body>
</html>
```

Output



2.2.2 Operators

An operator is a symbol which operates on one or more values or variables. For example: + is an operator which performs addition of more than one values or variables.

Types of Operators

JavaScript has collections of operators to perform various operations.

A. Arithmetic Operators

GQ. State four arithmetic operators.

- Arithmetic operators are binary operator means that it takes two operands to perform operation on it.
- Addition, subtraction, division, multiplication are the basic operations performed by arithmetic operators.
- Following table shows all the arithmetic operators supported by the JavaScript language.
- Here we consider two variables M = 5 and N = 10

Operator	Description	Example
+	Performs addition of two operands	M + N = 15
–	Performs subtraction of two operands. Subtract second operand from first.	N - M = 5
*	Performs multiplication of two operands	M* N = 50
/	Performs division of two operands and produces quotient as the result	N / M = 2
%	Performs division of two operands and produces reminder as the result.	N % M = 0

B. Increment and Decrement Operator

GQ. State the use of increment and decrement operators.

- Increment and Decrement operators are unary operators means that they take one operand to perform operation on it.
- Increment operator adds 1 to the variable.
- Decrement operator subtracts 1 from the variable.
- Pre-increment and pre-decrement add and subtract one to and from the variable first and then perform assignment operation.

Example 1

```
Pre-increment:  int a=5;
                int b=++a;
Here a and b both have 6.
Pre-decrement:  int a=5;
                int b=--a;
Here a and b both have 4.
```

- Post-increment and Post-decrement do the assignment operation first and then perform add and subtract one to and from the variable.

Example 2

```
Post-increment:  int a=5;
                 int b=a++;
Here value a will be 6 and b will be 5
Post-decrement:  int a=5;
                 int b=a--;
Here value a will be 4 and b will be 5
```

- Here we consider a variable M =5

Operator	Description	Example
++	Increment operator increases the integer value by 1.	Pre-increment: ++M = 6
		Post-increment: M++ = 6
--	Decrement operator decreases the integer value by 1.	Pre-decrement: --M = 4
		Post-decrement: M-- = 4

☞ Difference between ++I and I++

GQ. State the difference between I++ and ++I.

Parameter	++I	I++
Known as	Known as pre-increment operator.	Known as post-increment operator
Processing	After incrementing the value of i by one its updated value is used in expression.	First the expression gets evaluated and then the value of i will be incremented by one.
Effect of result	If we use this operator in an expression then it will affect the result of that expression.	If we use this operator in an expression then it will not affect the result of that expression.
Example	Consider a=7, b=0 then, b= b + 2 + (++a); After execution of above statement the value of a and b will be: a = 8, b = 10.	Consider a=7, b=0 then, b= b + 2 + (a++); After execution of above statement the value of a and b will be: a = 8, b = 9.

Module 2

☞ Difference between --i and i--

GQ. Compare --i and i--.

	--i	i--
Known as	Known as pre-decrement operator.	Known as post-decrement operator
Processing	After decrementing the value of i by one its updated value is used in expression.	First the expression gets evaluated and then the value of i will be decremented by one.
Effect of result	If we use this operator in an expression then it will affect the result of that expression.	If we use this operator in an expression then it will not affect the result of that expression.
Example	Consider a=7, b=0 then, b= b + 2 + (--a); After execution of above statement the value of a and b will be: a = 6, b = 8.	Consider a=7, b=0 then, b= b + 2 + (a--); After execution of above statement the value of a and b will be: a = 6, b = 9.

C. Relational Operators

GQ. Describe any two relational operators in JavaScript with simple example.

- Relational operators check the relation between two operands. Following table shows all the relational operators supported by JavaScript.
- Here we consider two variables M=10 and N=20

Operator	Description	Example
==	Determines whether the two operands have same values or not. If equal then condition is true.	(M == N) is not true.
!=	Determines whether the two operands have same values or not. If not equal then condition is true.	(M != N) is true.
>	Determines whether the left side operand have greater value than the right side operand. If it is, then the condition is true.	(M > N) is not true.
<	Determines whether the left side operand have less value than the right side operand. If it is, then the condition is true.	(M < N) is true.
>=	Determines whether the left side operand have value greater than or equal to the right side operand. If it is, then the condition is true.	(M >= N) is not true.
<=	Determines whether the left side operand has value less than or equal to the right side operand. If it is, then the condition is true.	(M <= N) is true.

D. Logical Operators

GQ. Describe any two logical operators in JavaScript with simple example.

- Logical operators are used to check the conditions. These operators are used in decision makings statements.
- It returns true or false according to the condition satisfaction.
- Following table shows all the logical operators supported by JavaScript language.
- Here we consider two variables M=10 and N=20

Operator	Description	Example
&&	Logical AND operator. Used to check multiple conditions. If all the conditions get satisfy then condition becomes true.	(M == 10) && (N != M) both conditions are true so result of && operation is true.
‖	Logical OR operator. Used to check multiple conditions. If either of the condition get satisfy then condition becomes true.	(M > 20) ‖ (N > M) here first condition is false and second condition is true so result of ‖ operation is true.
!	Logical NOT Operator. This operator reverses the logical state of the operand. If a condition is true, then false or vice versa.	!(M < N) : here M<N produces true but because of NOT operator it is converted to false.

E. Bitwise Operators

GQ. Explain any two bit-wise operators with example.

Bitwise operators work on bits and perform bit-by-bit operation.

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if bits of both the operands are set (1); otherwise it copies zero to the result.	(M & N) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit to the result if a bit of either the operand is set (1); otherwise it copies zero to the result.	(M | N) = 28, i.e., 0001 1100
^	Binary XOR Operator copies the bit if it is set in either operand, if bits of both the operands are set then it will copy zero into the result; otherwise one will be copied in the result set.	(M ^ N) = 16, i.e., 0010 0000
~	Binary Ones Complement Operator also called as NOT operator. It is unary operator. If the bit of the operand is set (1) it will copy zero in the result and if the bit is set (0) then it will copy one in the result.	(~M) = 28, i.e, 1110 0011 in 1's complement form.
<<	Binary Left Shift Operator. The bits of the left operand are moved left by the number of bits specified by the right operand.	M << 2 = 112 i.e., 0111 0000
>>	Binary Right Shift Operator. The bits of the left operand are moved right by the number of bits specified by the right operand.	M >> 1 = 14 i.e., 0000 1110
>>>	Unsigned Right Shift	

F Conditional Operators / ? : Operator

GQ. What is conditional operator ?

- There are also some other operators which are important and supported by JavaScript language. One of them is Ternary operator (? :) which performs on three operators.
- It is also known as conditional operator.
- Ternary operator performed on three operands :

  A ? B : C
- Means that if A is true then perform B otherwise perform C. Here A, B, C can be conditions of expressions.

Example

Let us consider two variables X=10 and Y=20.

MAX = (X>Y) ? X : Y

- Value of variable having greater value will be assigned to the MAX. In this case the value of Y will be assigned to the MAX.

2.3 CONTROL FLOW STATEMENTS

Module 2

GQ. Explain control structures in JavaScript.

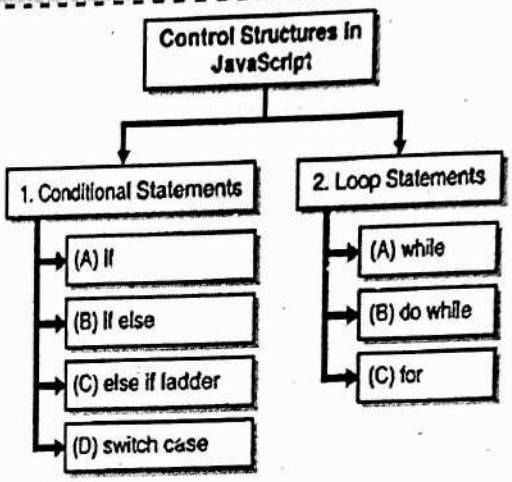


Fig. 2.3.1 : Control Structures in JavaScript

2.3.1 Conditional Statements

JavaScript provides following types of conditional statements:

- **if** – used to specify a block of statements to be executed if the given condition is true.
- **else** – used to specify a block of statements to be executed if the given condition is false.
- **else if** – used to specify another condition to test if the previous condition is false.
- **switch** – used to specify many alternative blocks of code to be executed.

1. The if Statement

The if statement used to specify a block of statements to be executed, if the given condition is true.

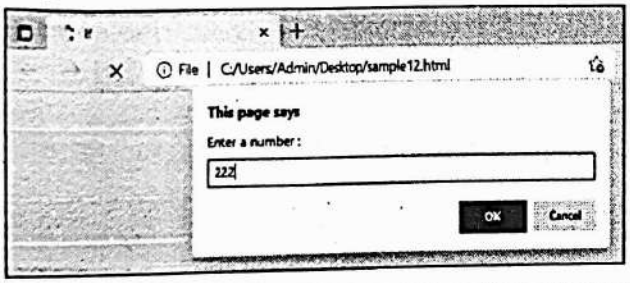
Syntax

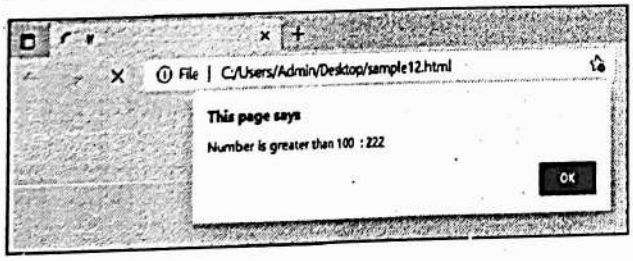
```
if (condition)
{
  statements;
}
```

GQ. Write a program to accept a number and check whether it is greater than 100 or not.

```
<html>
<head>
<title>
if
</title>
</head>
<body>
<script language="JavaScript">
var n;
n = parseInt(prompt("Enter a number : "));
if(n>100)
{
    alert("Number is greater than 100 : "+n);
}
</script>
</body>
</html>
```

Output





2. The if else Statement

The if else statement is used to specify a block of statements to be executed, if the given condition is false.

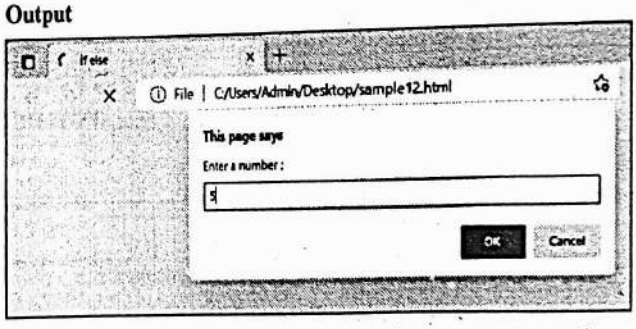
☞ Syntax

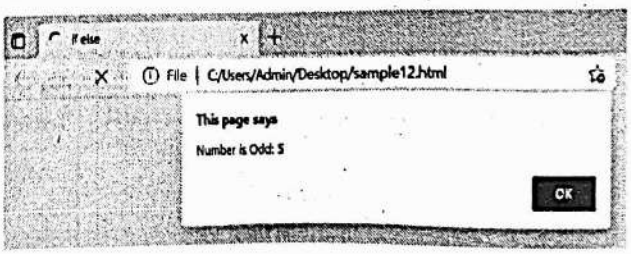
```
if (condition)
{
  statements;
}
else
{
  statements;
}
```

GQ. Write a program to accept a number from user and check whether it is even or odd.

```
<html>
<head>
<title>
if else
</title>
</head>
<body>
<script language="JavaScript">
var n;
n = parseInt(prompt("Enter a number : "));
if(n%2==0)
{
    alert("Number is Even : "+n);
}
else
{
    alert("Number is Odd: "+n);
}
</script>
</body>
</html>
```

Output





3. else if Ladder

It is used to specify another condition to test if the previous condition is false.

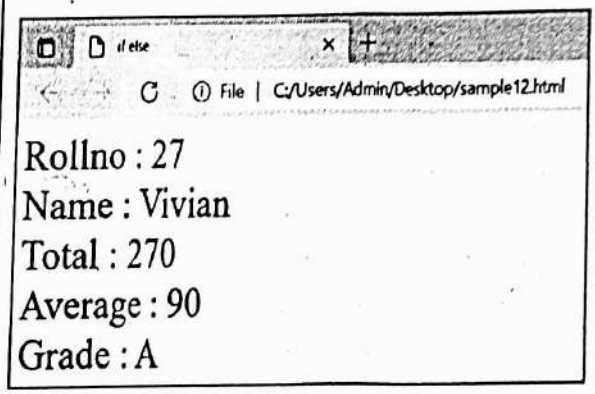
☞ **Syntax**

```
if (condition)
{
    statements;
}
else if(condition)
{
    statements;
}
Else
{
    statements;
}
```

GQ Write a program which accepts roll no., name and marks of 3 subjects from student. Calculate the total and average of marks and print the grade (avg >=80; then grade – A, >=60 then grade – B, >=40 then grade – C else Fail...). (Note : The student should get the grade only if he/she is pass in all the subjects.)

```
<html>
<head>
<title>
if else
</title>
</head>
<body>
<font size=6 color="blue">
<script language="JavaScript">
var rno,sname,Web,DAA,SYS,total,avg;
rno = prompt("Enter roll number : ");
sname = prompt("Enter name : ");
Web = parseInt(prompt("Enter marks of Python : "));
DAA = parseInt(prompt("Enter marks of CN : "));
SYS = parseInt(prompt("Enter marks of ML : "));
total = Web + DAA + SYS;
avg = total / 3;
document.write("Rollno : "+rno);
document.write("<br>Name : "+sname);
document.write("<br>Total : "+total);
document.write("<br>Average : "+avg);
if(Web>=40 && DAA>=40 && SYS>=40)
{
    if(avg>=80)
        document.write("<br>Grade : A");
    else if(avg>=60)
        document.write("<br>Grade : B");
    else if(avg>=40)
        document.write("<br>Grade : C");
}
else
{
    document.write("<br>Fail...");
}
</script>
</font>
</body>
</html>
```

Output



4. Switch Case Statement

Switch statement is used to select one of many blocks of code to be executed.

☞ **Syntax**

```
switch(expression)
{
  case constant_expression:
   statements;
    break;
  case constant_expression:
   statements;
    break;
  ----------
  default:
   statements;
}
```

Module 2

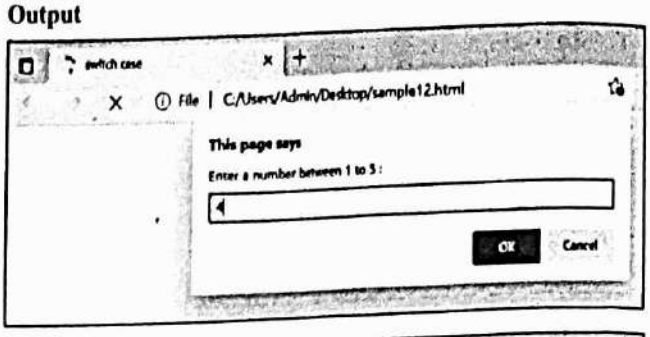
Working

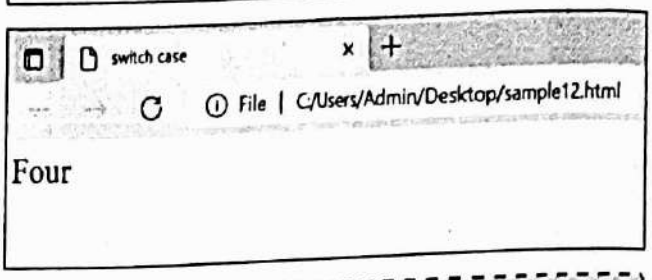
- The switch expression is evaluated only once.
- The value of expression is compared with the values of each constant expression.
- If there is a match, then the related statements are executed.

GQ. Write a program to print the numbers between 1 to 5.

```
<html>
<head>
<title>
switch case
</title>
</head>
<body>
<font size=5 color="blue">
<script language="JavaScript">
var n;
n = parseInt(prompt("Enter a number between 1 to 5 : "));
switch(n)
{
    case 1:
        document.write("One");
    break;
    case 2:
        document.write("Two");
    break;
    case 3:
        document.write("Three");
    break;
    case 4:
        document.write("Four");
    break;
    case 5:
        document.write("Five");
    break;
    default:
        document.write("Not in the range");
}
</script>
</font>
</body>
</html>
```

Output



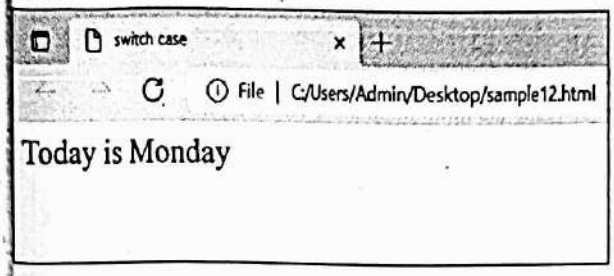


GQ. Write a program to display the name of weekday using switch case.

```
<html>
<head>
<title>
switch case
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
switch (new Date().getDay())
{
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
```

```
 case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
document.write("Today is "+day);
</script>
</font>

</body>
</html>
```

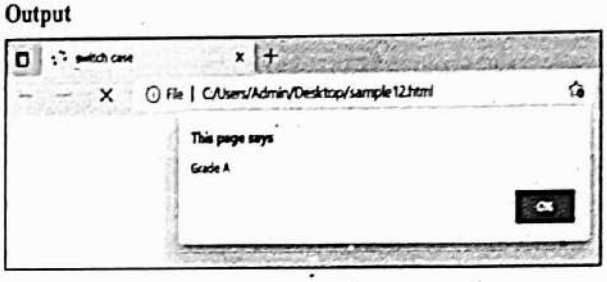
Output

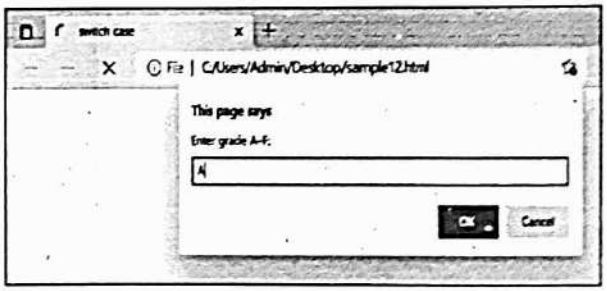


GQ. Write a JavaScript code to enter grade from user between 'A' TO 'F', display appropriate message.

```
<html>
<head>
<title>
switch case
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var grade;
grade = prompt("Enter grade A-F:");

switch (grade)
{
  case 'A':
alert("Grade A");
    break;

  case 'B':
alert("Grade B");
    break;
  case 'C':
alert("Grade C");
    break;
  case 'D':
alert("Grade D");
    break;
  case 'E':
alert("Grade E");
    break;
  case 'F':
alert("Grade F");
    break;
default:
alert("Invalid grade...");

}

</script>
</font>

</body>
</html>
```

Output





Module 2

2.3.2 Loop Statements

GQ. Explain Loop statements in JavaScript.

1. **while Loop**

Syntax

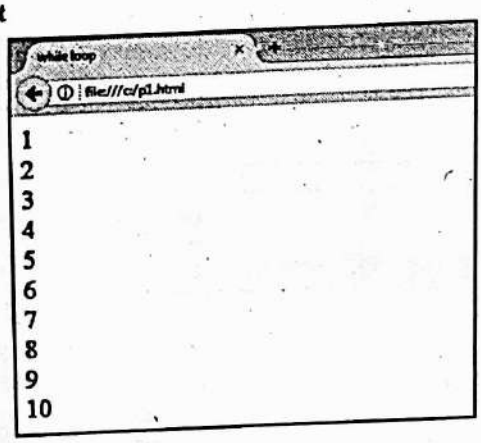
```
While (condition)
{
    Statements;
}
```

The while is an entry controlled loop. That means if the given condition is not satisfied then the loop statements will never get execute.

GQ. Write a program to print 1 to 10 numbers using while loop.

```
<html>
<head>
<title>
while loop
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var i = 1;
while(i<=10)
{
    document.write(i + "<br>");
    i = i + 1;
}
</script>
</font>
</body>
</html>
```

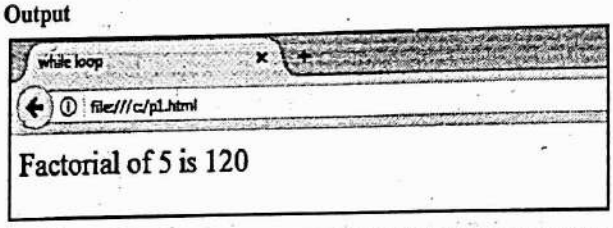
Output



GQ. Write a program to accept a number from user and print factorial of it.

```
<html>
<head>
<title>
while loop
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var n,n1,f;
f = 1;
n = parseInt(prompt("Enter a number :"))
n1 = n;
while(n>0)
{
    f = f * n;
    n--;
}
document.write("Factorial of " + n1 + " is " + f);
</script>
</font>
</body>
</html>
```

Output



GQ. Write a JavaScript to find first 10 prime numbers.

JavaScript to find first 10 prime numbers

```
<html>
<head>
<title>
JavaScript
</title>
</head>

<body>
```

```
<foht size=5>
<script language="JavaScript">
  document.write("First Ten Prime Numbers Are<br>");

var n,flag,i,cnt;
nt = 1;
= 1;

hile(1)

    i = 2;
    flag = 0;

hile(i<n)

    if(n%i==0)

    flag = 1;
    break;

++;

f(flag==0)
{
    document.write(n + "<br>")
    cnt++;
}
f(cnt>10)
reak;
++;
}
</script>
</font>
</body>
</html>
```

Output

2. do while Loop

Syntax

```
do
{
    Statements;
} while (condition);
```

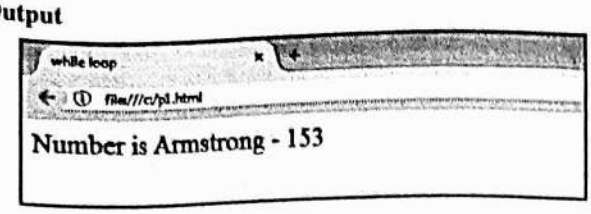
The do while statement is an exit-controlled loop. That is, the loop statement is executed without first checking any conditions. The condition is checked starting with the second execution. As a result, even if the condition is not met, the loop statements are executed once.

Module 2

GQ. Write a program to accept a number from user and check whether it is Armstrong number or not.

```
<html>
<head>
<title>
while loop
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var n,n1,r,sum;
sum = 0;
n = parseInt(prompt("Enter a number :" ))
n1 = n;
do
{
    r = n%10;
    sum = sum + (r*r*r);
    n = Math.floor(n / 10);
}while(n>0);
if(n1==sum)
    document.write("Number is Armstrong - "+n1);
else
    document.write("Number is not Armstrong - "+n1);
</script>
</font>
</body>
</html>
```

Output



3. **for Loop**

☞ Syntax

```
for (initialization; test condition; iteration statement)
{
    Statements;
}
```

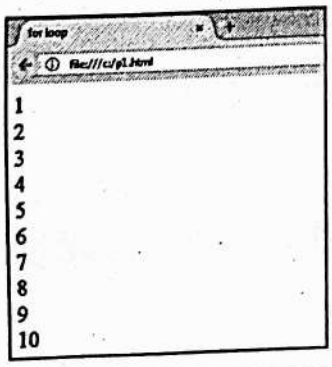
☞ Explanation

- In for loop the initialization, condition and increment or decrement of loop variable is done in a single statement.
- For loop helps to minimize the code.

GQ. Write a program to print 1 to 10 numbers using for loop.

```
<html>
<head>
<title>
for loop
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var i = 1;
for(i=1;i<=10;i++)
{
    document.write(i + "<br>");
}
</script>
</font>
</body>
</html>
```

Output



☞ Nested for loop

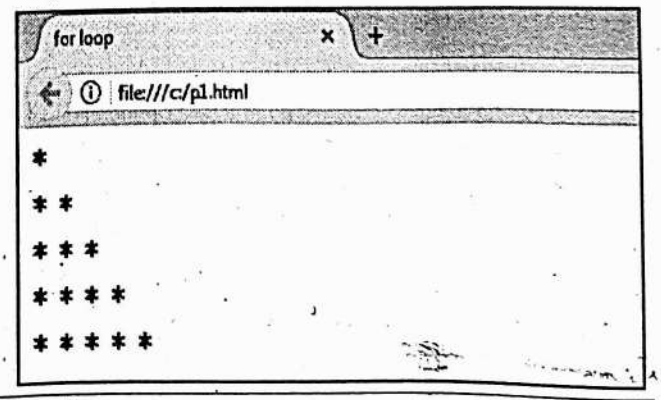
GQ. Write a program to print the following pattern.

```
*
* *
* * *
* * * *
* * * * *
```

☞ Program to display the given ascending * pattern

```
<html>
<head>
<title>
for loop
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var i,j;
for(i=1;i<=5;i++)
{
for(j=1;j<=i;j++)
{
    document.write("* ");
}
 document.write("<br>");
}
</script>
</font>
</body>
</html>
```

Output



2.4 POPUP BOXES

GQ. Which are three types of popup boxes in JavaScript? Explain with syntax and examples.

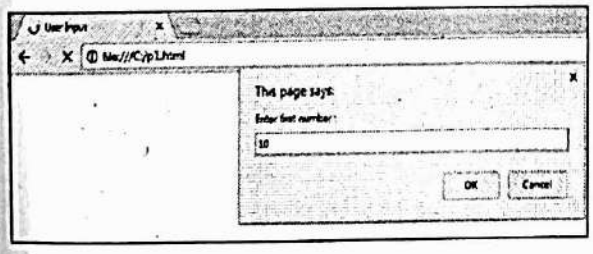
- JavaScript provides following types of popup boxes :
- **prompt() :** This method is used to display an input box which can accept value from user.
- **alert() :** This method is used to display a message box.
- **Confirm()** - displays the confirm dialog box containing message with ok and cancel button.
- Prompt() and alert() are methods of Windows Object.

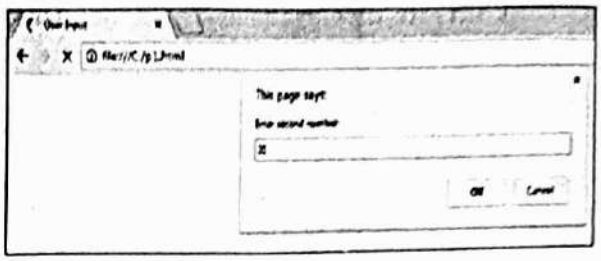
GQ. Write a program to accept two numbers from user and display their summation.

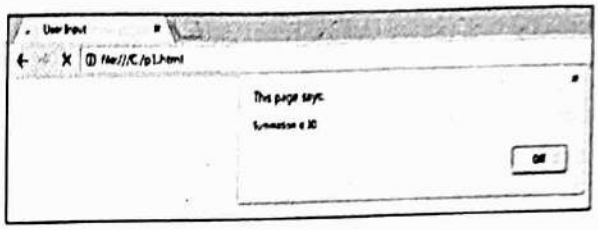
```
<html>
<head>
<title>
User Input
</title>
</head>
<body>
<script language="JavaScript">
var n1,n2,sum;
n1 = parseInt(prompt("Enter first number : "));
n2 = parseInt(prompt("Enter second number : "));
sum = n1 + n2;
alert("Summation is "+sum);
</script>
</body>
</html>
```

The parseInt() method converts the string type of data in integer format.

Output







Module 2

2.5 FUNCTIONS

GQ. Explain user defined function in JavaScript with suitable example.

- Function helps to avoid repetition of code and write modular codes. A program can be divided into small and manageable modules called as functions.
- Just like the other programming languages such as C, C++ and Java, the function concept with all its features is supported by JavaScript.
- We can create our own functions known as user defined functions.

2.5.1 Function Definition

In JavaScript, a function is defined by using the **function** keyword, followed by name of function, a parameters list [optional], and a statement block inside curly braces.

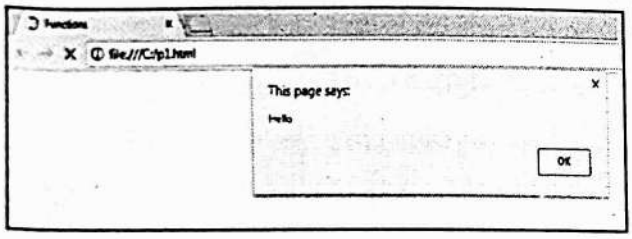
Syntax

```
<script type="text/javascript">
    function function_name([parameter-list])
    {
      Statements;
    }
</script>
```

Program to display hello message using function call

```
<html>
<head>
<title>
Functions
</title>
</head>
<body>
<script language="JavaScript">
function callMe()
{
    alert("Hello");
}
callMe();
</script>
</body>
</html>
```

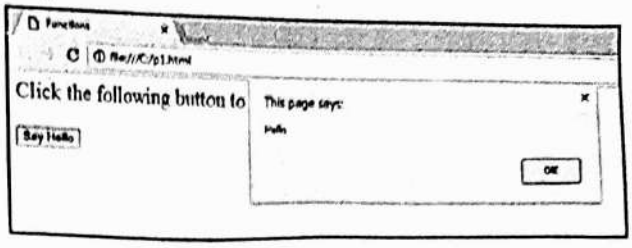
Output



Calling a function on click event of a button

```
<html>
<head>
<title>
Functions
</title>
<script language="JavaScript">
function callMe()
{
    alert("Hello");
}
</script>
</head>
<body>
<font size=5>
 <p>Click the following button to call the function</p>
<form>
<input type="button" onClick="callMe();" value="Say
Hello">
</form>
</font>
</body>
</html>
```

Output



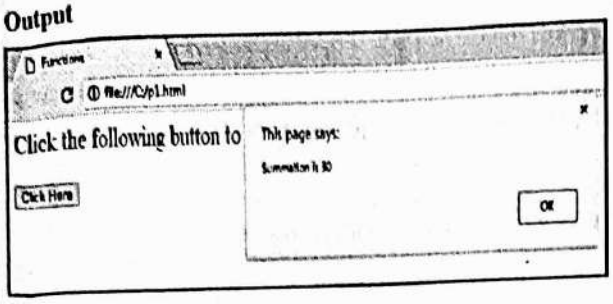
2.5.2 Defining Function Arguments

While defining a function, we can declare variables in the header statement of the function. These variables are known as parameters or formal arguments. When this function gets called, we can pass values for these variables. These values are known as arguments or actual arguments.

GQ. Write a program to display the summation of two values using parameterized function.

```
<html>
<head>
<title>
Functions
</title>
<script language="JavaScript">
function add(a,b)
{
    var sum = a + b;
    alert("Summation is "+sum);
}
</script>
</head>
<body>
<font size=5>
 <p>Click the following button to call the function</p>

    <form>
      <input type="button" onClick="add(10,20);"
    value="Click Here">
    </form>
</font>
</body>
</html>
```

Output



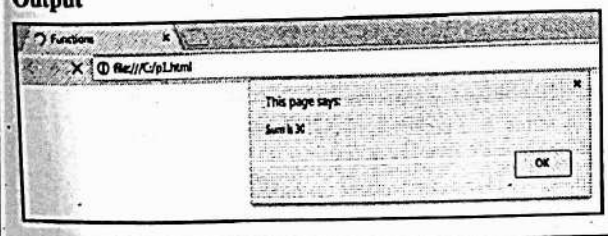
2.5.3 Defining a Return Statement

- In JavaScript function we can have return statement which is optional. This helps to return a value from a function. This statement is written at the last in a function.
- The returned value goes to the location where the function is called.

GQ. Write a program which will accept two numbers as arguments and returns their summation.

```
<html>
<head>
<title>
Functions
</title>
<script language="JavaScript">
function add(a,b)
{
    var sum = a + b;
    return(sum);
}
</script>
</head>
<body>
<script language="JavaScript">
var r = add(10,20);
alert("Sum is "+r);
</script>
</body>
</html>
```

Output



2.6 JAVASCRIPT OBJECTS

GQ. What are JavaScript objects? List the important built-in objects. How can you write your own object?

An object is nothing but an entity having its own state and behaviour (properties and methods).

- **For example :** A flower is an object having properties like color, fragrance etc. Other examples of objects are car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-oriented language. Everything in JavaScript is considered as an object.

Module 2

Examples of object

Following are some of the examples of objects in JavaScript.

(i) Booleans (ii) Numbers
(iii) Strings (iv) Dates
(v) Regular expressions (vi) Arrays
(vii) Functions

- We can create our own user defined objects in JavaScript.
- JavaScript is primarily a template-based scripting language rather than a class-based language. Hence, we directly create the object without class.

2.6.1 Creating Objects in JavaScript

1. JavaScript Object by Object Literal

Syntax

Following is the syntax of creating object using object literal :

```
object={property1:value1,property2:value2.....propertyN:valu
eN}
```

As we can observe, the property and value is separated by the separator : (colon).

GQ. Write a sample program of creating object using object literal.

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
```

```
<script language="JavaScript">
employee = {id:101,name:" Prajakta Lonkar",salary:40000}
document.write(employee.id+"-"+employee.name+"-"
+employee .salary);
</script>
</font>
</body>
</html>
```

Output

```
29-Prajakta Lonkar-40000
```

2. **By creating instance of object**

☞ Syntax

Following is the syntax of creating instance of object :

```
var objectname=new Object();
```

Here, **new keyword** is used to create object.

☞ Program of creating the instance of object

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var employee=new Object();
employee.id=18;
employee.name="Chetana Murudkar";
employee.salary=800000;
document.write(employee.id+"-"+employee.name+"-"
+employee.salary);
</script>
</font>
</body>
</html>
```

Output

```
18-Chetana Murudkar-800000
```

3. **By using an object constructor**

- Here, we have to create parameterized function. **"this"** keyword is used to assign each argument value in the current object.
- The "this" keyword refers to the current object.

☞ Program using "this" keyword

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
function student(id,sname,marks)
{
    this.id=id;
    this.sname=sname;
    this.marks=marks;
}
s = new student(20,"Gauri",90);
document.write(s.id+" – "+s.sname+" – "+s.marks);
</script>
</font>
</body>
</html>
```

Output

```
20 - Gauri - 90
```

2.6.2 String Object

The JavaScript String object is a global object that is used to store strings. A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all.

☞ Syntax

```
var val = new String(string);
```

The String parameter is a series of characters that has been properly encoded.

☞ String Methods

Here is a list of the methods available in String object along with their description.

Sr. No.	Method	Description
1.	charAt()	Returns the character at the specified index.
2.	charCodeAt()	Returns the Unicode of the character at the specified index.
3.	concat()	Joins two or more strings, and returns a new string.
4.	endsWith()	Checks whether a string ends with a specified substring.
5.	fromCharCode()	Converts Unicode values to characters.
6.	includes()	Checks whether a string contains the specified substring.
7.	indexOf()	Returns the index of the first occurrence of the specified value in a string.
8.	lastIndexOf()	Returns the index of the last occurrence of the specified value in a string.
9.	localeCompare()	Compares two strings in the current locale.
10.	match()	Matches a string against a regular expression, and returns an array of all matches.
11.	repeat()	Returns a new string which contains the specified number of copies of the original string.
12.	replace()	Replaces the occurrences of a string or pattern inside a string with another string, and return a new string without modifying the original string.
13.	search()	Searches a string against a regular expression, and returns the index of the first match.
14.	slice()	Extracts a portion of a string and returns it as a new string.
15.	split()	Splits a string into an array of substrings.
16.	startsWith()	Checks whether a string begins with a specified substring.
17.	substr()	Extracts the part of a string between the start index and a number of characters after it.
18.	substring()	Extracts the part of a string between the start and end indexes.
19.	toLocaleLowerCase()	Converts a string to lowercase letters, according to host machine's current locale.
20.	toLocaleUpperCase()	Converts a string to uppercase letters, according to host machine's current locale.
21.	toLowerCase()	Converts a string to lowercase letters.
22.	toString()	Returns a string representing the specified object.
23.	toUpperCase()	Converts a string to uppercase letters.
24.	trim()	Removes whitespace from both ends of a string.
25.	valueOf()	Returns the primitive value of a String object.

2.6.3 RegExp Object

- RegExp object used to validate the pattern of characters.
- RegExp define methods that use regular expressions to perform powerful pattern-matching and search and replace functions on text.
- Regular expressions can be defined by using following ways
  - Using RegExp() Constructor: var RegularExpression = new RegExp("pattern","flag");
  - Using Literal: var RegularExpression = /pattern/flag;
  - Pattern – A String that specifies the pattern of the regular expression or another regular expression.
  - Flag – An optional string containing any of the "g", "I" and "m" attributes that specify global, case-insensitive and multiple matches respectively.

Properties of RegExp object

1. **Constructor** - Returns the function that created the RegExp object.
2. **Global** - Checks whether the "g" modifier is set.
3. **ignoreCase** - Checks whether the "i" modifier is set.
4. **lastIndex** - Specifies the index at which to start the next match.
5. **multiline** - Checks whether the "m" modifier is set.
6. **source** - Returns the text of the RegExp pattern.

Methods of RegExp object

1. **exec()** - Tests for a match in a string. Returns the first match.
2. **test()** - Tests for a match in a string. Returns true or false.
3. **toString()** - Returns the string value of the regular expression.
4. **toSource()** - Returns an object literal representing the specified object.

Following examples explain more about matching characters.

Expression	Description
[^a-zA-Z]	It matches any string not containing any of the characters ranging from a through z and A through Z.
p.p	It matches any string containing p, followed by any character, in turn followed by another p.
^.{2}$	It matches any string containing exactly two characters.
(.*)	It matches any string enclosed within and .
p(hp)*	It matches any string containing a p followed by zero or more instances of the sequence hp.

2.6.4 Math Object

- The Math object is used to perform simple and complex arithmetic operations.
- The Math object provides a number of properties and methods to work with number values
- The Math object does not have any constructors. All of its methods and properties are static; that is, they are member functions of the Math object itself. There is no way to create an instance of the Math object.

Syntax

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

Math Methods

Following are some important methods of Math Object :

Method	Description
abs()	Returns the absolute value of a number.
ceil()	Returns the smallest integer greater than or equal to a number.
exp()	Returns E^N, 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.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is, base exponent.
random()	Returns a pseudo-random number between 0 and 1.
round()	Returns the value of a number rounded to the nearest integer.
sqrt()	Returns the square root of a number.

2.6.5 Date Object

- At times when user need to access the current date and time and also past and future date and times. JavaScript provides support for working with dates and time through the Date object.
- The Date object provides a system-independent abstraction of dates and times.
- Date object can be created as : **var today = new Date( );**
- Dates may be constructed from a year, month, day of the month, hour, minute, and second, and those six components, as well as the day of the week, may be extracted from a date.
- Dates may also be compared and converted to a readable string form. A Date is represented to a precision of one millisecond.

Properties of Date object

1. **Constructor** - Returns the function that created the Date object.
2. **Prototype** - Add properties and methods to an object.

Methods of Date object

1. **Date()** - Returns today's date and time
2. **getDate()** - Returns the day of the month for the specified date
3. **getDay()** - Returns the day of the week for the specified date
4. **getFullYear()** - Returns the year of the specified date
5. **getHours()** - Returns the hour in the specified date according to local time.
5. **getMilliseconds()** - Returns the milliseconds in the specified date according to local time.
6. **getMinute(), getMonth(), getTime(), getTimezoneOffset(), setDate(), setFullYear(), setHours(), setMilliseconds(), setMinutes(), setMonth(), setSeconds(), setTime()** are some of the methods used in Date object.

2.6.6 Number Object

- The Number object represents numerical date, either integers or floating-point numbers.
- A Number objects are created using the number() constructor **var num = new number(value);**

Properties of Number object

1. **Constructor** - Returns the function that created the Number object.
2. **MAX VALUE** - Returns maximum numerical value possible in JavaScript.
3. **MIN VALUE** - Returns minimum numerical value possible in JavaScript.
4. **NEGATIVE INFINITY** - Represent the value of negative infinity.
5. **POSITIVE INFINITY** - Represent the value of infinity.
6. **Prototype** - Add properties and methods to an object.

Methods of Number object

1. **toExponential()** - Converts a number into exponential notation.
2. **toFixed()** - Formats a number with a specific number of digits to the right of the decimal.
3. **toLocaleString()** - Returns a string value version of the current number in a format that may vary according to a browser's locale settings.
4. **toPrecision()** - Defines how many total digits to display of a number.
5. **toString()** - Returns the string representation of the number's value.
6. **valueOf()** - Returns the number's value.

2.6.7 Array Object

- Multiple values are stored in a single variable using the Array object.
- In JavaScript, an array can hold different types of data types, which implies that an array can have a string, a number or an object in a single slot.
- An Array object can be created by using following ways:

Using the Array Constructor

To create empty array when don't know the exact number of elements to be inserted in an array

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

To create an array of given size

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

Module 2

To create an array with given elements

```
var arrayname = new Array("element 1","element 2",.....,"element n");
```

Using the Array Literal Notation:

To create empty array

```
var arrayname =[];
```

To create an array when elements are given

```
var arrayname =["element 1","element 2",.....,"element n"];
```

☞ Properties of the Array object

1. **Length** - Returns the number of elements in the array.
2. **Constructor** - Returns the function that created the Array object.
3. **Prototype** - Add properties and methods to an object.

☞ Methods of the Array object

1. **reverse()** - Reverses the array elements
2. **concat()** - Joins two or more arrays
3. **sort()** - Sort the elements of an array
4. **push()** - Appends one or more elements at the end of an array
5. **pop()** - Removes and returns the last element
6. **shift()** - Removes and returns the first element
7. **unshift(), join(), indexOf(), lastIndexOf(), slice(startindex, endindex)** are some of the methods used in Array object.

⏭ 2.7 BROWSER OBJECTS - WINDOW, NAVIGATOR, HISTORY, LOCATION, DOCUMENT

Following are some important browser objects provided by JavaScript :

2.7.1 Window Object

GQ. What is the purpose of Window object in JavaScript?

- The **window object** represents a window in browser. An object of window is created automatically by the browser.
- JavaScript Window Object Properties

Property	Description
closed	returns a boolean value that specifies whether a window has been closed or not
Default Status	specifies the default message that has to be appeared in the status bar of a window
document	specifies a document object in the window
frames	specifies an array of all the frames in the current window
history	specifies a history object for the window
Inner Height	specifies the inner height of a window's content area
Inner Width	specifies the inner width of a window's content area
length	specifies the number of frames contained in a window
location	specifies a location object for the window

☞ Methods of window object

The important methods of window object are as follows:

Methods	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

2.7.2 Navigator Object

- The **JavaScript navigator object** is used for browser detection. It can be used to get browser information such as appName, appCodeName, userAgent etc.
- The navigator object is the window property, so it can be accessed by:

```
window.navigator
OR
navigator
```

JavaScript Navigator Object Properties

Property	Description
appcodename	specifies the code name of the browser
appname	specifies the name of the browser
appversion	specifies the version of the browser being used
cookieenabled	specifies whether the cookies are enabled or not in the browser
platform	contains a string indicating the machine type for which the browser was compiled
useragent	contains a string representing the value of the user-agent header sent by the client for the server in the http protocol

Methods of JavaScript navigator object

- Following are important methods of navigator object :

Method	Description
javaEnabled()	checks if java is enabled.
taintEnabled()	checks if taint is enabled. It is deprecated since JavaScript 1.2.

Example

```
<html>
<body>
<h2>JavaScript Navigator Object</h2>
<script>
document.writeln("<br/>navigator.appCodeName:
"+navigator.appCodeName);
document.writeln("<br/>navigator.appName:
"+navigator.appName);
document.writeln("<br/>navigator.appVersion:
"+navigator.appVersion);
document.writeln("<br/>navigator.cookieEnabled:
"+navigator.cookieEnabled);
document.writeln("<br/>navigator.language:
"+navigator.language);
document.writeln("<br/>navigator.userAgent:
"+navigator.userAgent);
document.writeln("<br/>navigator.platform:
"+navigator.platform);
document.writeln("<br/>navigator.onLine:
"+navigator.onLine);
</script>
</body>
</html>
```

Output

JavaScript Navigator Object

navigator.appCodeName: Mozilla
navigator.appName: Netscape
navigator.appVersion: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
navigator.cookieEnabled: true
navigator.language: en-GB
navigator.userAgent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
navigator.platform: Win32
navigator.onLine: true

Module 2

2.7.3 History Object

GQ. What is the purpose of History browser object in JavaScript ? Explain its any two methods.

- The **JavaScript history object** represents an array of URLs visited by the user. By using this object, you can load previous, forward or any particular page.
- The history object is the window property, so it can be accessed by:

```
window.history
OR
history
```

Properties of JavaScript history object

Property	Description
Length	specifies the number of elements contained in the object
Current	specifies the URL of the current entry in the object
Next	specifies the URL of the next element in the History list
previous	specifies the URL of the previous element in the History list

Methods of JavaScript history object

There are only 3 methods of history object.

Method	Description
forward()	loads the next page.
back()	loads the previous page.
go()	loads the given page number.

Example

```
history.back();//for previous page
history.forward();//for next page
history.go(2);//for next 2nd page
history.go(-2);//for previous 2nd page
```

2.7.4 Location Object

GQ. What is the purpose of Location browser objects in JavaScript? Explain its methods.

- The location object in JavaScript helps in storing the information of current URL of the window object. It is a child object of the window object.
- The table given below describes the properties of the Location object in JavaScript.

Property	Description	Example
href	represents a string specifying the entire URL	http://PhoenixGlobe.com:80?test.asp?id=1#start
protocol	represents a string at the beginning of a URL up to the first colon (:), which specifies the method of access to the URL ,	http: or https:
host	represents a string consisting of the hostname and port strings	PhoenixGlobe.com:80
hostname	represents the server name, subdomain, and domain name of a URL	PhoenixGlobe.com
port	represents a string specifying the communications port that the server uses	80
pathname	represents a string portion of a URL specifying how a particular resource can be accessed	test.asp
search	represents a string beginning with a question mark that specifies any query information in an HTTP URL	id=1
hash	represents a string beginning with a # specifies an anchor name in an HTTP URL	start

Location Object Methods

The following table lists the methods of the Location object in JavaScript.

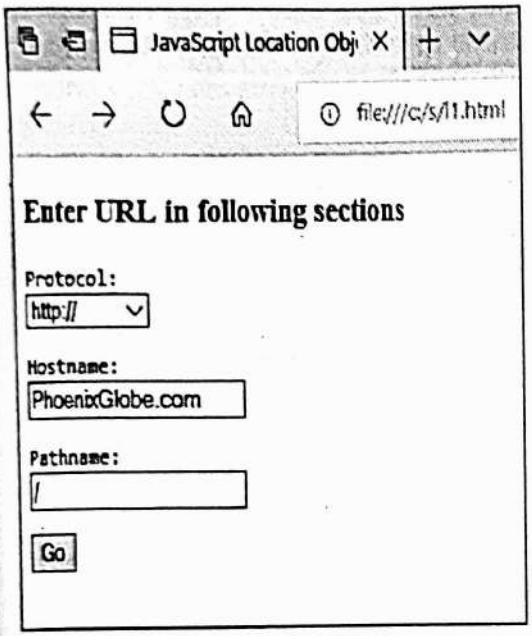
Method	Description
assign()	Loads a new document in the browser
reload()	reloads the current document that is contained in the location.href property
replace()	replaces the current document with the specified new one, you can not navigate back to the previous document using the browser's Back button

Example

```
<!DOCTYPE HTML><html>
<head>
    <title>JavaScript Location Object</title>
    <script type="text/javascript">
        function gotoUrl()
        {
window.location.href = window.document.loctn.ProtocolFld.
options[window.document.loctn.ProtocolFld.selectedIndex].
text + document.loctn.HostnameFld.value + document.loctn.
PathnameFld.value
        }
    </script>
</head>
<body>
<h3>Enter URL in following sections</h3>
<form name="loctn" method="post">
<pre>Protocol:
<select name="ProtocolFld" size="1">
<option>http://</option>
<option>file://</option>
<option>javascript:</option>
<option>ftp://</option>
<option>mailto:</option>
</select>
</pre>
<pre>
Hostname:
<input type="text" size="20" maxlength="256"
name="HostnameFld" value="PhoenixGlobe.com">
</pre>
<pre>
```

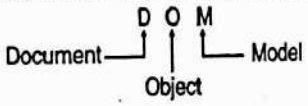
```
Pathname:
<input type="text" size="20" maxlength="100"
name="PathnameFld" value="/">
</pre>
<pre>
<input type="button" name="Go" value="Go"
onclick="gotoUrl()">
</pre>
</form>
</body>
</html>
```

Output



2.8 DOCUMENT OBJECT MODEL

GQ. Discuss Document Object Model.



- **DOM stands for Document Object Model.**
- The entire html document is represented by the document object.
- The html document becomes document object when it is loaded in the browser. The root element represents the html document. The document object has properties and methods. The document object helps to add content dynamically in the web page.
- Any element of HTML page can be accessed by using the document object.

According to W3C(World Wide Web Consortium) "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Hierarchy of objects in web document

Module 2

Document Object Model (DOM) is the method by which the content of document is accessed and modified. In a web document, the organization of objects is implemented in a hierarchical structure.

2.8.1 DOM Levels

The DOM provides JavaScript with all of the features it needs to generate dynamic HTML:

- Changes can be made in all HTML elements.
- Changes can be made in attributes of HTML elements.
- Changes can be made in all CSS styles in the page.
- Existing HTML elements and attributes can be deleted.
- New HTML elements and attributes can be added.
- Response can be given to HTML events.
- New HTML events can be created in HTML page.

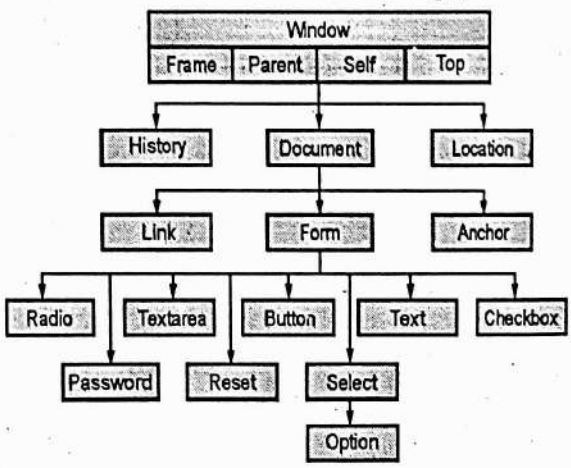


Fig. 2.8.1 : DOM levels

2.8.2 Properties and Methods of Document Object

The contents of document can be accessed and modified with the help of methods.

Sr. No.	Method / Properties	Description
1.	Value	Returns value of specified text field
2.	write("string")	Writes the specified string in the document.
3.	writeln("string")	Writes the specified string in the document with newline character at the end.
4.	getElementById()	Returns the element having the specified id value.
5.	getElementsByName()	Returns all the elements having the specified name value.
6.	getElementsByTagName()	Returns all the elements having the specified tag name.
7.	getElementsByClassName()	Returns all the elements having the specified class name.

1. Accessing field value by document object

➢ **Program 2.8.1 :** Program of document object which accesses value from text field

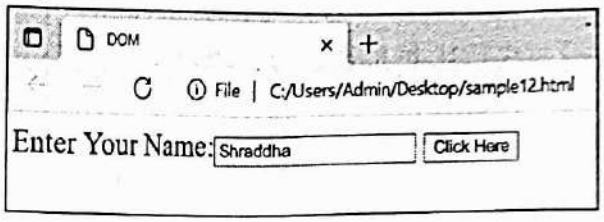
```
<html>
<head>
<title>
DOM
</title>
<script language="JavaScript">
function callMe()
{
var sname=document.form1.txtname.value;
alert("Welcome: "+sname);
}
</script>
</head>
<body>
<font size=5>
<form name="form1">
Enter Your Name:<input type="text" name="txtname"/>
<input type="button" onClick="callMe();" value="Click Here"/>
</form>
</font>
</body>
</html>
```

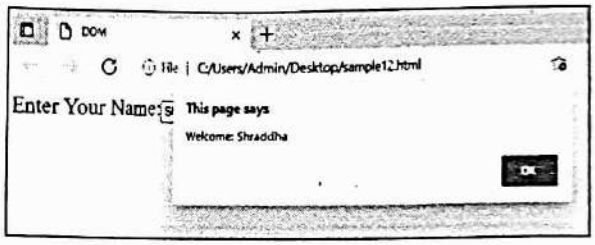
☞ **Explanation**

Here, document represents the html document. It is the root element.

- **form1** - name of the form.
- **txtname** - attribute name of the input text.
- **Value** - property, that returns the value of the input text.

Output





2. write("String") method

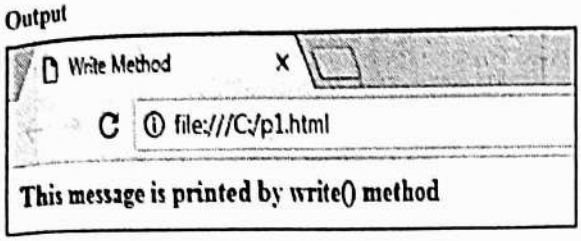
- This method is used to write output stream.
- The write() method is used to writes HTML expressions or JavaScript code to a document.

☞ **Displaying the message using write("string") method**

```
<html>
<head>
<title>
Write Method
</title>
</head>
<body>
<script language="JavaScript">
document.write("<b>This message is printed by write() method</b>");
```

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

Output



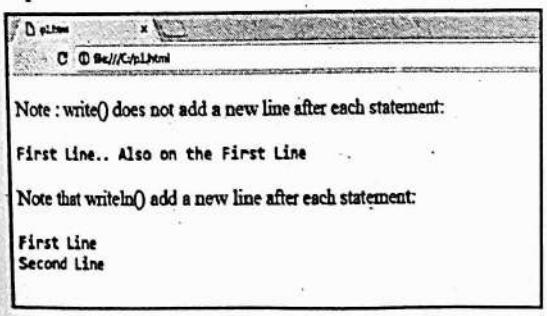
3. writeln("String") method

- This method is used to write output stream.
- writeln() add a new line after each statement.

➤ **Program 2.8.2 :** Program to demonstrate the use of writeln("String") method

```
<!DOCTYPE html>
<html>
<body>
<font size=5>
<p>Note : write() does not add a new line after each
statement:</p>
<pre>
<script>
document.write("First Line");
document.write(".. Also on the First Line");
</script>
</pre>
<p>Note : writeln() add a new line after each
statement:</p>
<pre>
<script>
document.writeln("First Line");
document.writeln("Second Line");
</script>
</pre>
</font>
</body>
</html>
```

Output



4. document.getElementById() method

This method returns the element of specified id.

☞ **Syntax**

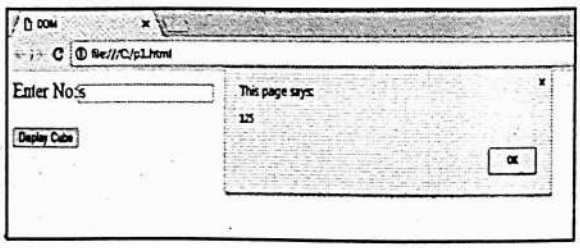
```
document.getElementById("id")
```

In the above script, we have used document.form1.name.value to access the value of text field. For the same purpose now we will use Document.getElementById() method. Here we have to define id for the input text field.

➤ **Program 2.8.3 :** Program to display cube of number using Document.getElementById() method

```
<html>
<head>
<title>
DOM
</title>
<script language="JavaScript">
function cube()
{
var no = document.getElementById("num").value;
alert(no*no*no);
}
</script>
</head>
<body>
<font size=5>
<form>
Enter No:<input type="text" id="num"
name="num"/><br/> <br/>
<input type="button" value="Display Cube"
onClick="cube()"/>
</form>
</font>
</body>
</html>
```

Output



Module 2

5. document.getElementsByName() method

This method returns all the elements of given name.

Syntax

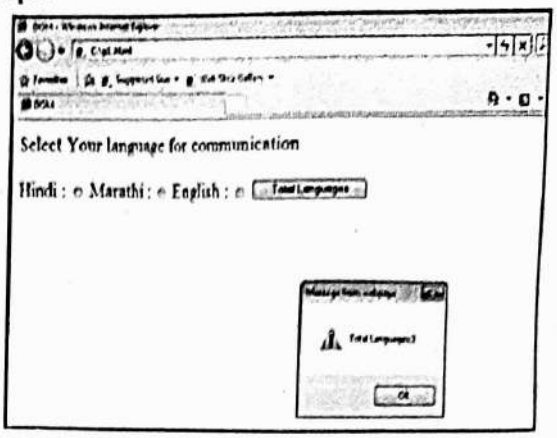
```
document.getElementsByName("name")
```

In this example, we will count total number of languages. Here, we are using getElementsByName() method to get all the languages.

➢ **Program 2.8.4 :** Program to display number of languages and their count on click event of button

```
<html>
<head>
<title>
DOM
</title>
<script type="text/javascript">
function languages()
{
var lan = document.getElementsByName("lang");
alert("Total Languages:"+lan.length);
}
</script>
</head>
<body>
<font size=5>
<form>
Select Your language for communication </br></br>
Hindi : <input type="radio" name="lang" value="Hindi">
Marathi : <input type="radio" name="lang"
value="Marathi">
English : <input type="radio" name="lang"
value="English">
<input type="button" onClick="languages()" value="Total
Languages">
</form>
</font>
</body>
</html>
```

Output



6. document.getElementsByTagName() method

All the elements of specified tag are returned by this method.

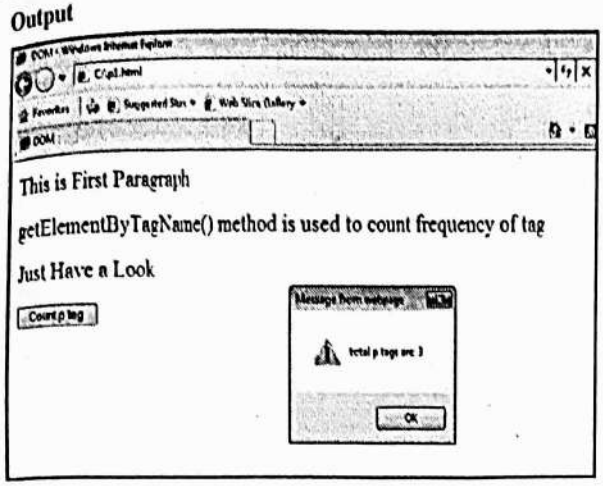
Syntax

```
document.getElementsByTagName("tag_name")
```

➢ **Program 2.8.5 :** Program to count total number of tags used by document.getElementsByTagName() method

```
<html>
<head>
<title>
DOM
</title>
<script type="text/javascript">
function cntp()
{
var cnt =document.getElementsByTagName("p");
alert("total p tags are: "+cnt.length);
}
</script>
</head>
<body>
<font size=5>
<form>
<p>This is First Paragraph</p>
<p>getElementByTagName() method is used to count
frequency of tag </p>
<p>Just Have a Look</p>
<button onclick="cntp();">Count p tag</button>
</form>
</font>
</body>
</html>
```

Output



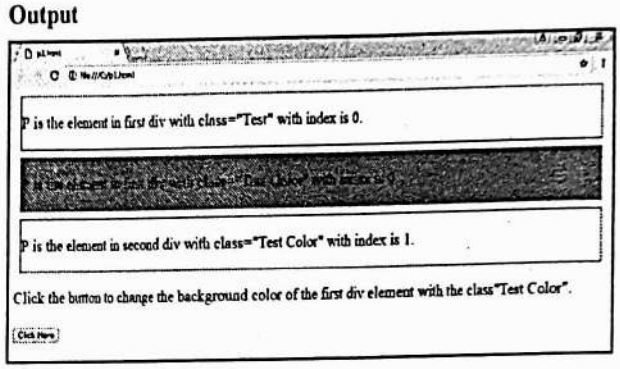
7. document.getElementsByClassName()

- This method returns a set of all the elements of the document with the given class name, as a NodeList object.
- The NodeList object issued to represent a set of nodes. The nodes are accessed using index numbers which starts from 0.

➤ **Program 2.8.6 :** Program to demonstrate the use of getElementsByClassName( ) method.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 2px solid gray;
  margin: 10px;
}
</style>
</head>
<body>
<font size=5>
<div class="Test">
  <p>P is the element in first div with class="Test" with
index is 0.</p>
</div>
<div class="Test Color">
  <p>P is the element in first div with class="Test Color"
with index is 0.</p>
</div>
<div class="Test Color">
  <p>P is the element in second div with class="Test Color"
with index is 1.</p>
</div>
<p>Click the button to change the background color of the
first div element with the class"Test Color".</p>
<button onclick="changeColor()">Click Here</button>
<script language="JavaScript">
function changeColor() {
  var x = document.getElementsByClassName("Test Color");
  x[0].style.backgroundColor = "gray";
}
</script>
</font>
</body>
</html>
```

Output



2.8.3 Manipulation using DOM

GQ. Write a note on DOM manipulation.

Using JavaScript we can add or remove nodes (HTML elements) in the document.

(A) Creating New HTML Elements (Nodes)

Initially we have to create the element (element node) which can be then appended to the existing element.

➤ **Program 2.8.7 :** Program to append the new element to an existing element node

```
<html>
<head>
<title>
DOM Manipulation
</title>
<div id="div1">
<p id="p1">It's not that I'm so smart, it's just that I stay with
problems longer.
</p>
<p id="p2">Albert Einstein : Theoretical Physicist,
Philosopher, Nobel Prize Winner</p>
</div>
```

Module 2

```
<script>
var prg = document.createElement("p");
var nd = document.createTextNode("A new Paragraph.");
prg.appendChild(nd);
var element = document.getElementById("div1");
element.appendChild(prg);
</script>
</body>
</html>
```

Explanation

Let's see the code

```
var prg = document.createElement("p");
```

A new <p> element is created

```
var nd = document.createTextNode("A new paragraph.");
```

A text node is created and text is added to element <p>.

```
prg.appendChild(node);
```

The text node is appended to the element <p>

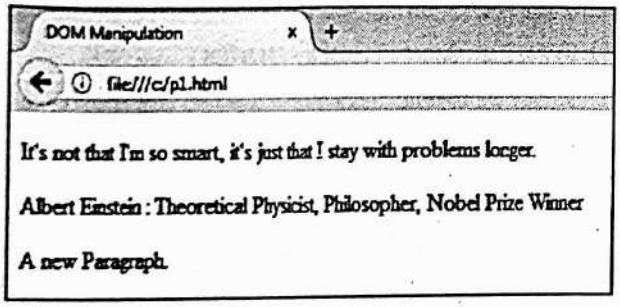
```
var element = document.getElementById("div1");
```

This code finds an existing element:

```
element.appendChild(prg);
```

This code appends the new element to an existing element.

Output



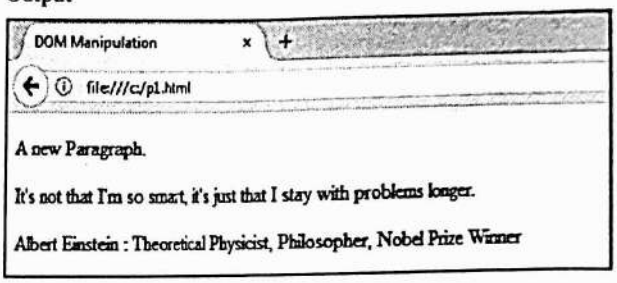
(B) Creating new HTML Elements - insertBefore()

- In the previous program, we have seen the appendChild() method which appends the element at the end in parent.
- The insertBefore() method is used to add the new element at the beginning.

➢ **Program 2.8.8 :** Program to add new element at the beginning using insertBefore() method

```
<html>
<head>
<title>
DOM Manipulation
</title>
<div id="div1">
<p id="p1">It's not that I'm so smart, it's just that I stay with problems longer.
</p>
<p id="p2">Albert Einstein : Theoretical Physicist, Philosopher, Nobel Prize Winner</p>
</div>
<script>
var prg = document.createElement("p");
var nd = document.createTextNode("A new Paragraph.");
prg.appendChild(nd);
var element = document.getElementById("div1");
var child = document.getElementById("p1");
element.insertBefore(prg, child);
</script>
</body>
</html>
```

Output



(C) Removing Existing HTML Elements

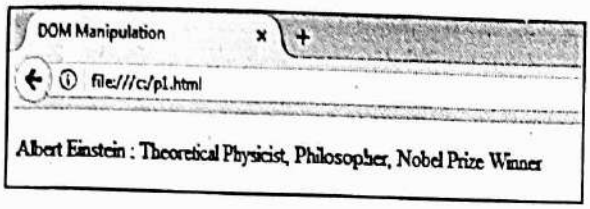
To remove an HTML element, we should have the information about the parent element. The removeChild() method is used to remove the HTML element.

➢ **Program 2.8.9 :** Program to remove an existing element using removeChild() method

```
<html>
<head>
<title>
DOM Manipulation
</title>
<div id="div1">
<p id="p1">It's not that I'm so smart, it's just that I stay with problems longer.
</p>
<p id="p2">Albert Einstein : Theoretical Physicist, Philosopher, Nobel Prize Winner</p>
```

```
</div>
<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
</body>
</html>
```

Output



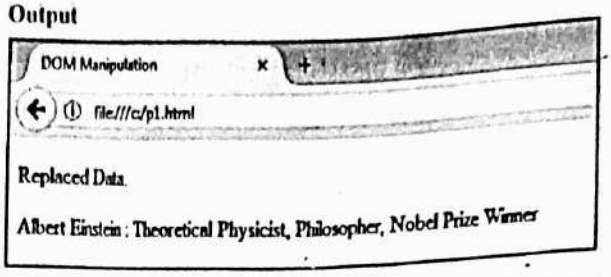
(D) Replacing HTML Elements

It is also possible to replace an existing element by new one. The replaceChild() method is used for this purpose.

➢ **Program 2.8.10 :** Program to replace an existing element by new element using replaceChild() method

```
<html>
<head>
<title>
DOM Manipulation
</title>
<div id="div1">
<p id="p1">It's not that I'm so smart, it's just that I stay with problems longer.
</p>
<p id="p2">Albert Einstein : Theoretical Physicist, Philosopher, Nobel Prize Winner</p>
</div>
<script>
var prg = document.createElement("p");
var nd = document.createTextNode("Replaced Data.");
prg.appendChild(nd);
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.replaceChild(prg, child);
</script>
</body>
</html>
```

Output



2.9 EVENT HANDLING

Module 2

GQ. Explain the event handling in JavaScript with simple example.

- Events are the actions perfonned by the end users while browsing the website. For example mouse move or mouse click on the buttons.
- When an event is fired, objects are triggered which are associated with that specific event.
- The event is caught by the event handlers and in response the related code is executed.
- Events are basically classified in four categories :

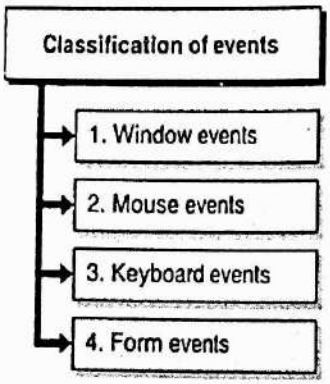


Fig. 2.9.1 : Classification of events

1. Window Events

There are various types of events associated with window :

(i) **onLoad** - triggered when a new page is starting up.

(ii) **onUnload** - triggers when a page is shutting down.

(iii) **onResize** - triggers when a page is resized.

(iv) **onMove** - triggers when a page is moved.

(v) **onAbort** - triggers when a page is cancelled.

(vi) **onError** - triggers when an error occurs.

(vii) **onFocus** - triggers when the window moves to foreground.

(viii) **onBlur** - triggers when window changes to background.

2. **Mouse Events**

There are various types of events associated with mouse :

(i) **onMousedown** - triggers when mouse button is pressed on an element.

(ii) **onMouseup** - triggers when mouse button is released.

(iii) **onMousemove** - triggers when mouse pointer is moved and the pointer is already over an element.

(iv) **onMouseout** - triggers when mouse pointer is moved out of an element.

(v) **onMouseover** - triggers when the pointer is over an element.

(vi) **onClick** - triggers when mouse button is clicked once.

(vii) **onDblclick** - triggers when mouse button is clicked twice.

3. **Keyboard Events**

There are various types of events associated with keyboard :

(i) **onKeydown** - Triggers when a key is pressed down.

(ii) **onKeyup** - Triggers when a key is released.

(iii) **onKeypress** - Triggers when complete key sequence, down press and up release happens.

4. **Form events**

There are following types of events associated with form :

(i) **onReset** - triggers when the reset button on the form is clicked.

(ii) **onSubmit** - triggers when the submit button is clicked.

(iii) **onSelect** - triggers when a content is selected on a page.

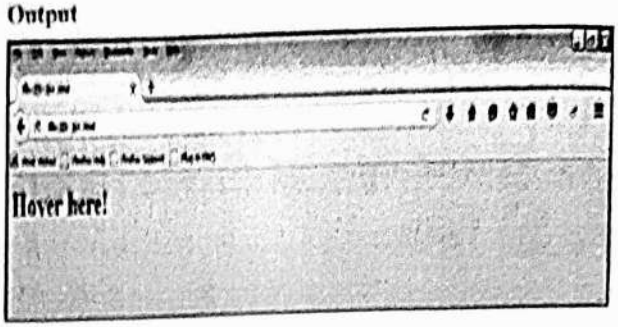
(A) Using Events

All these event can be used to execute some scripts. In the following programs we will use mouseover and mouseout events.

➢ **Program 2.9.1 :** Program to illustrate the use of mouseover and mouseout events

```
<html>
<body bgcolor="lightgray">
<h1 onmouseover="style.color='blue'"
onmouseout="style.color='black'">
Hover here!
</h1>
</body>
</html>
```

Output

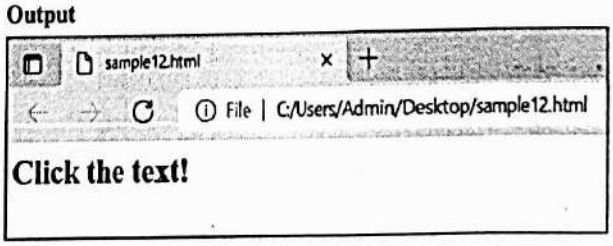


Use of mouse down and mouse up events

➢ **Program 2.9.2 :** Program illustrating mouse up and mouse down events

```
<html>
<body>
<h2 id="myid1" onmousedown="fun1()"
onmouseup="fun2()">Click the text! </h2>
<script>
function fun1()
{
    document.getElementById("myid1").style.color = "blue";
}
function fun2()
{
    document.getElementById("myid1").style.color= "black";
}
</script>
</body>
</html>
```

Output



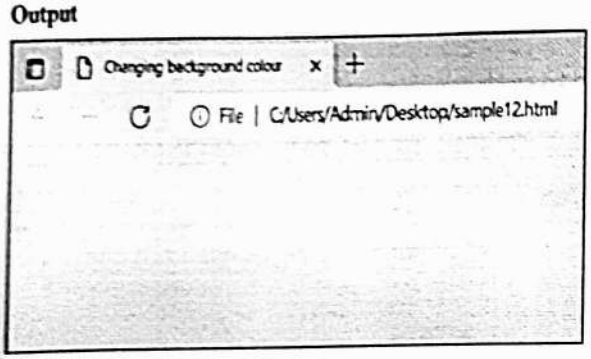
Programs

➢ **Program 2.9.3 :** Write a program to show the use of events to change background color with DHTML. If mouse button is pressed, background color should be red. If mouse button is released up, background color should be yellow.

```
<html>
<head>
<title>Changing background colour </title>
```

```
<script type="text/javascript">
function red()
{
document.body.style.backgroundColor="red"
}
function yellow()
{
    document.body.style.backgroundColor="yellow"
}
</script>
</head>
<body onMouseDown="red();" onMouseUp="yellow();">
</body>
</html>
```

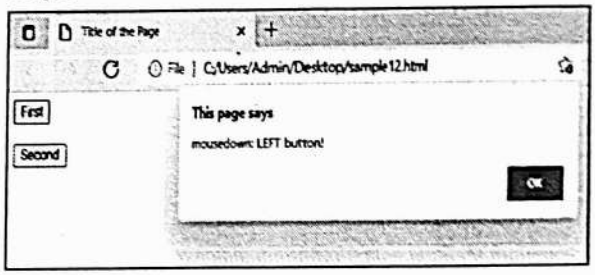
Output

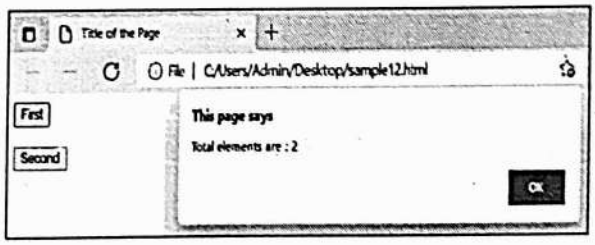


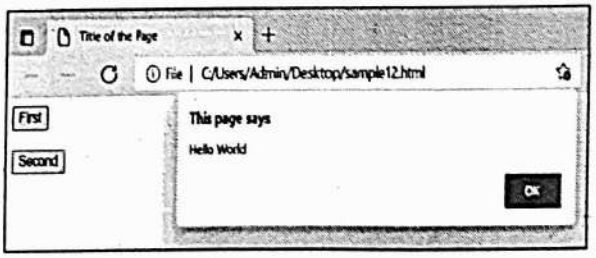
➢ **Program 2.9.4 :** Write using JavaScript how to know which mouse button was clicked, number of elements in form, and write hello world.

```
<html>
<head>
<title>Title of the Page</title>
</head>
<body>
<script language="JavaScript">
var sTestEventType='mousedown';
function handleMouseEvent(e) {
var evt = (e==null ? event:e);
var clickType = 'LEFT';
if (evt.type!=sTestEventType) return true;
if (evt.which)
{
    if (evt.which==3) clickType='RIGHT';
    if (evt.which==2) clickType='MIDDLE';
}
    alert(evt.type+': '+clickType+' button!');
    alert("Total elements are : "+document.frm.length);
    alert("Hello World");
return true;
}
    document.onmousedown = handleMouseEvent;
    document.onmouseup   = handleMouseEvent;
    document.onclick     = handleMouseEvent;
</script>
<form name="frm">
<input type="button" value="First"><br><br>
<input type="button" value="Second">
</form>
</body>
</html>
```

Output







Module 2

➢ **Program 2.9.5 :** Write JavaScript that handles following mouse event.

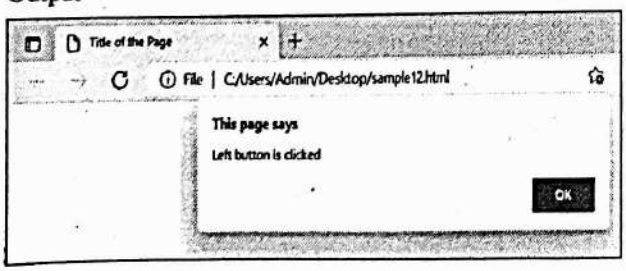
(a) If mouse left button pressed on browser it displayed message "Left Clicked".

(b) If mouse right button pressed on browser it displayed message "Right Clicked".

☞ JavaScript handling mouse events

```
<html>
<head>
<title>Title of the Page</title>
</head>
<body>
<script language="JavaScript">
var sTestEventType='mousedown';
function handleMouseEvent(e)
{
    var evt = (e==null ? event:e);
    var clickType = 'LEFT';
    if (evt.type!=sTestEventType) return true;
 if (evt.which)
 {
    if (evt.which==3) clickType='RIGHT';
 }
if(clickType == 'LEFT')
    alert("Left button is clicked");
    else if(clickType == 'RIGHT')
    alert("Right button is clicked");
    return true;
}
document.onmousedown = handleMouseEvent;
document.onmouseup  = handleMouseEvent;
document.onclick    = handleMouseEvent;
</script>
</body>
</html>
```

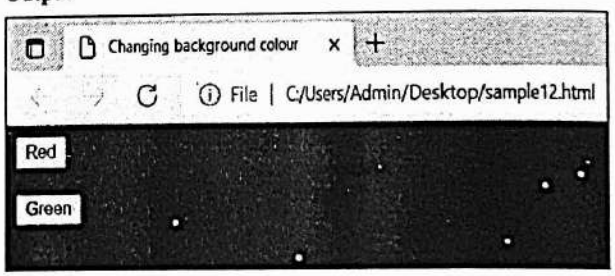
Output



➢ **Program 2.9.6 :** If button named "red" is clicked, background should change to red and If button named "green" is clicked, background should change to green.

```
<html>
<head>
<title>Changing background colour </title>
<script type="text/javascript">
function red()
{
    document.body.style.backgroundColor="red"
}
function green()
{
    document.body.style.backgroundColor="green"
}
</script>
</head>
</head>
<body>
<form name="frm">
<input type="button" value="Red"
onClick="red();"><br><br>
<input type="button" value="Green" onClick="green();">
</form>
</body>
</html>
```

Output



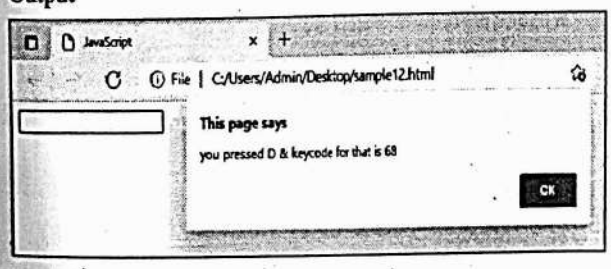
➢ **Program 2.9.7 :** Write a JavaScript that handles following mouse events. Add necessary elements. (i) JavaScript gives the key code for the key pressed. (ii) If the key pressed is "a","e","i","o","u", the script should announce that vowel is pressed. (iii) When the key is released background should change to blue.

☞ JavaScript to handle mouse events

```
<html>
<head>
<title>JavaScript</title>
```

```
<script type="text/javascript">
function myKeyPress(e)
{
var keynum;
if(window.event) // IE
{
    keynum = e.keyCode;
}
else if(e.which){ // Netscape/Firefox/Opera
keynum = e.which;
}
var char1=String.fromCharCode(keynum);
//for vovel
if(char1=='a' || char1=="e" || char1=='i' || char1=='o'
|| char1=='u')
{
    alert("you have pressed vovel");
}
alert("you pressed "+char1 +" & keycode for that is "+
keynum);
}
function changebgcolor()
{
document.getElementById("txt1").style.backgroundColor =
"blue";
}
</script>
</head>
<body>
<form>
<input type="text" id="txt1" onkeyup="changebgcolor()"
onkeypress="myKeyPress(event)"/>
</form>
</body>
</html>
```

Output



➤ **Program 2.9.8 :** Design HTML form which include two fields username and password. Write JavaScript code to show and hide password.

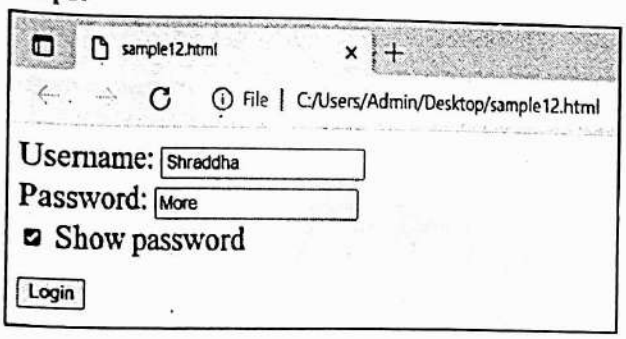
☞ **JavaScript code to show and hide password**

```
<html>
<body>
<Script>
(function() {

    var PasswordToggler = function( element, field ) {
        this.element = element;
        this.field = field;

        this.toggle();
    };
    PasswordToggler.prototype = {
        toggle: function() {
            var self = this;
            self.element.addEventListener( "change",
function() {
                if( self.element.checked ) {
                    self.field.setAttribute( "type", "text" );
                } else {
                self.field.setAttribute( "type", "password" );
                }
        }, false);
        }
    };
    document.addEventListener( "DOMContentLoaded",
function() {
    var checkbox = document.querySelector( "#show-hide" ),
    pwd = document.querySelector( "#pwd" ),
            form = document.querySelector( "#login" );

    form.addEventListener( "submit", function( e ) {
    e.preventDefault();
    }, false);

    var toggler = new PasswordToggler( checkbox, pwd );

    });

})();
</Script>
<form action="" method="post" id="login">
```

Module 2

```
    <font size=5><div>
Username: <input type="text" id="name" name="name"
/><br>
Password: <input type="password" id="pwd" name="pwd"
/>
    </div>
    <div>
        <input type="checkbox" id="show-hide"
        name="show-hide" value="" />
        <label for="show-hide">Show password</label>
    </div>
</font>
    <p><input type="submit" value="Login" /></p>
</form>
</body>
</html>
```

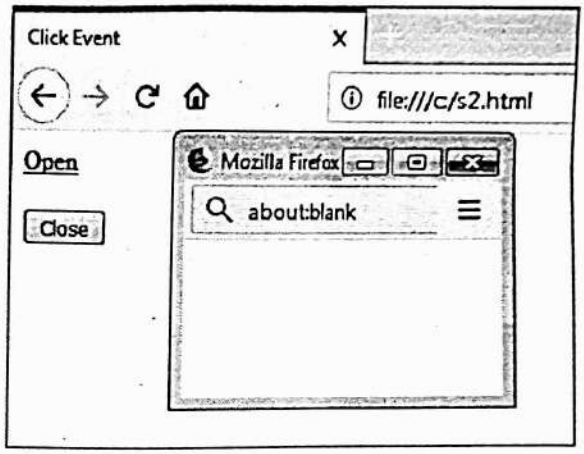
Output



➢ **Program 2.9.9 :** Write a code in JavaScript to open a window when a link on a page is clicked. The new window open is closed by placing a button on the window and writing JavaScript code on the onClick event of the button.

```
<html>
<head>
<title>
Click Event
</title>
</head>
<script language="JavaScript">
function openWin() {
   myWindow = window.open("","myWindow","width=200,
height=100");
}

function closeWin() {
   myWindow.close();
}
</script>

<form name="frm">
<a href="javascript:openWin()">Open</a>
<br><br>
<input type="button" value="Close"
onClick="closeWin();"> </input>
</form>
</html>
```

Output



2.10 FORM VALIDATION USING JAVASCRIPT

- JavaScript forms are the logical container of all the controls (elements).
- Before submitting any form to server, it is very important to check the form for its validity because it may possible that user has missed some data or entered some inappropriate data. It can be done on server side also. But if data is not appropriate then it has to be send to client side for correction which is a time consuming process. Hence JavaScript provides client side data validation with the help of forms. It makes the processing fast.
- Data validation is considered as a process of verifying that input given by user is clean, correct, and useful.

Validation tasks

Typical validation tasks are as follows :

- Whether all required fields are given by user?
- Whether date entered by user is valid?
- Whether data types of all values are correct?

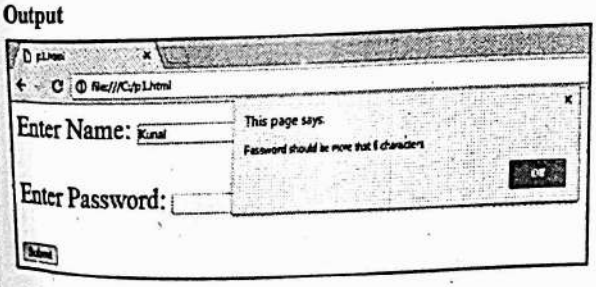
JavaScript form validation programs

GQ. Accept name and password from user and validate the data as follows : The name should not be empty. Password should not be less than 6 characters.

Validation of data

```
<html>
<body>
<script>
function chkData()
{
var nm = document.frm.username.value;
var pw = document.frm.pwd.value;

if (nm=="")
{
 alert("Please enter name");
return false;
}
else if(pw.length<6)
{
 alert("Password should be more that 6 characters");
 return false;
 }
}
</script>
<body>
<form name="frm" method="post" action="#"
onsubmit="return chkData();">
<font size=6>
Enter Name: <input type="text" name="username"><br/>
<br/>
Enter Password: <input type="password"
name="pwd"><br/>  <br/>
<input type="submit" value="Submit">
</font>
</form>
</body>
</html>
```

Output

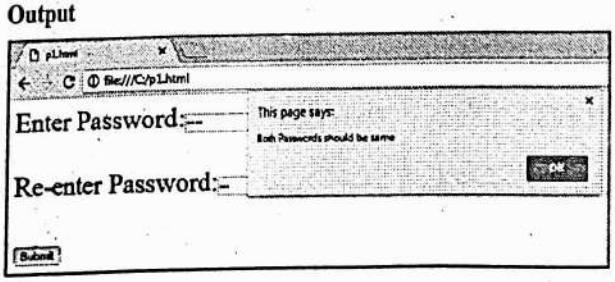


GQ. Write a JavaScript to check password and confirm passwords are same or not.

JavaScript to check the password

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function chkPass()
{
var first = document.frm.pass1.value;
var second = document.frm.pass2.value;

if(first==second)
{
return true;
}
else
{
alert("Both Passwords should be same");
return false;
}
}
</script>
</head>
<body>
<form name="frm" action="#" onsubmit="return
chkPass()">
<font size=6>
Enter Password:<input type="password" name="pass1"
/><br/><br/>
Re-enter Password:<input type="password"
name="pass2"/><br/><br/>
<input type="submit">
</font>
</form>
</body>
</html>
```

Output



Module 2

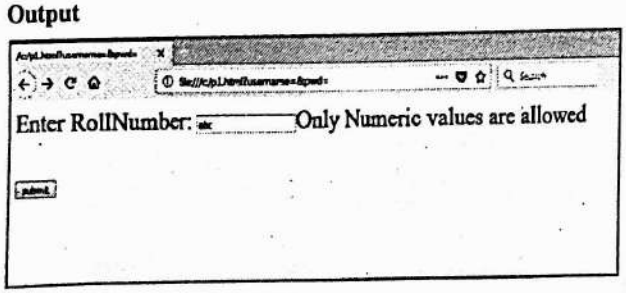
Number Validation

GQ. Write a JavaScript to validate the textfield for numeric value only. Use NaN() function.

JavaScript to validate the textfield using NaN() function

```
<!DOCTYPE html>
<html>
<head>
<script>
function validate()
{
    var n=document.frm.numdata.value;
    if (isNaN(n))
 {
 document.getElementById("numsp").innerHTML="Only
Numeric values are allowed";
 return false;
}
else
{
 return true;
 }
}
</script>
</head>
<body>
<form name="frm" action="#"onsubmit="return validate()"
>
<font size=6>
Enter RollNumber: <input type="text"
name="numdata"><span
id="numsp"></span><br/><br/>
<input type="submit" value="submit">
</font>
</form>
</body>
</html>
```

Output



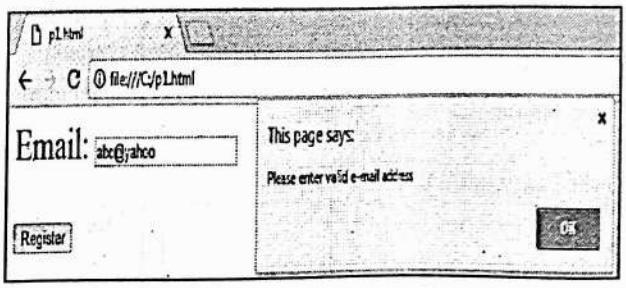
JavaScript email validation

GQ. Accept email-id from user and check with following criteria : Email id must contain .(dot) character and @ symbol. There should be minimum 1 character before and after the @. There should be minimum two characters after. (dot).

JavaScript to check email validation

```
<html>
<body>
<script>
function email_validate()
{
var eid = document.frm.txtemail.value;
var atpos = eid.indexOf("@");
var dotpos = eid.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=eid.length)
{   alert("Please enter valid e-mail address");
 return false;
 }
}
</script>
<body>
<form name="frm" method="post" action="#"
onsubmit="return email_validate();">
<font size=6>
Email: <input type="text" name="txtemail"><br/>  <br/>
 <input type="submit" value="Register">
</font>
</form>
</body>
</html>
```

Output



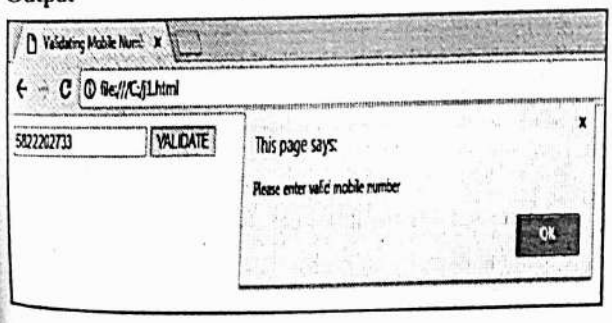
JavaScript mobile and email validation

GQ. Write down Java script code to check mobile number (mobile number should start with 9 or 8) and mail id(check @ and .).

JavaScript to check mobile and email validation

```
<html>
<head>
<title>Validating Mobile Number</title>
<script type="text/javascript">
function validate(mob)
{
  var flag = false;
  var mlength = mob.length;
  if((mob.charAt(0)==='8' || mob.charAt(0)==='9') &&
mlength==10)
  {
     flag=true;
  }

 if(flag==false)
alert("Please enter valid mobile number");
else
alert("Thanks, data submitted");
}
</script>
</head>
<body>
<form name="MyForm">
<input type="text" id="txtmobno" />
<input type="button" value="VALIDATE"
onclick="validate(txtmobno.value)" />
</form>
</body>
</html>
```

Output



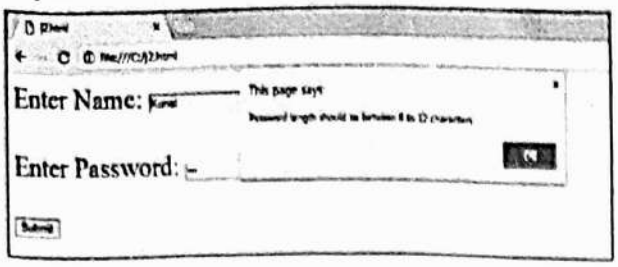
GQ. Write down Java script code to validate user name and password (password length must in between 6 to 12 characters. User name should not start with _, @, or any number, both are not blank).

JavaScript code to validate username password between specific length of characters

```
<html>
<body>
<script>
function chkData(nm,pw)
{
if (nm==""" || pw==""")
{
alert("Please fill all data");
}
else if(nm.charAt(0)=='_' || nm.charAt(0)=='@' ||
(nm.charAt(0)>='0' && nm.charAt(0)<='9'))
{
alert("Please enter valid name");
}
else if(pw.length<6 || pw.length>12)
{
 alert("Password length should be between 6 to 12
characters");
 }
 else
{
alert("Data submitted");
}
}
</script>
<body>
<form name="frm">
<font size=6>
Enter Name: <input type="text" name="username"><br/>
<br/>
Enter Password: <input type="password"
name="pwd"><br/>  <br/>
<input type="button" value="Submit"
onclick="chkData(username.value,pwd.value);">
</font>
</form>
</body>
</html>
```

Module 2

Output

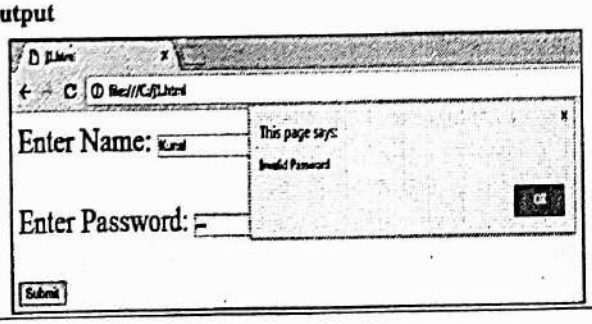


GQ. Design an login form using HTML and JavaScript with following validations on password field: minimum length 8 characters, it should have some special character.

JavaScript code to validate a form with minimum password length 8 characters

```
<html>
<body>
<script>
function chkData(pwd)
{
var paswd =  /^(?=.*[!@#$%^&*]).{8,}$/;
if(paswd.test(pwd))
alert("Correct Password");
else
alert("Invalid Password");

}
</script>
<body>
<form name="frm">
<font size=6>
Enter Name: <input type="text" name="username"><br/>
<br/>
Enter Password: <input type="password"
name="pwd"><br/>  <br/>
<input type="button" value="Submit"
onclick="chkData(pwd.value);">
</font>
</form>
</body>
</html>
```

Output

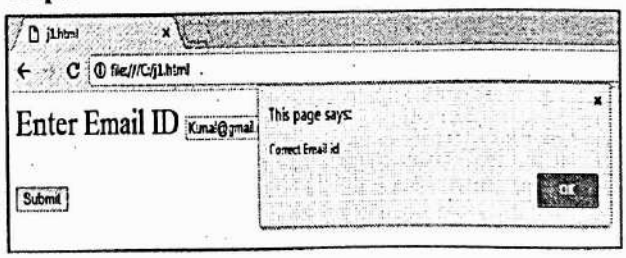


GQ. Show the use of regular expression in JavaScript to validate the email address with example.

JavaScript to validate the email address using regular expression

```
<html>
<body>
<script>
function chkData(eml)
{
var verf = /^\w+([\.-]?\w+)*@\w+([\.-]?\
w+)*(\.\w{2,3})+$/;
if(verf.test(eml))
alert("Correct Email id");
else
alert("Invalid Email id");
 }
</script>
<body>
<form name="frm">
<font size=6>
Enter Email ID <input type="text"
name="txtemail"><br/>  <br/>
<input type="button" value="Submit"
onClick="chkData(txtemail.value);">
</font>
</form>
</body>
</html>
```

Output



2.11 EXCEPTION HANDLING IN JAVASCRIPT

- In programming, exception handling is a process or method used for handling the abnormal statements in the code and executing them. It also enables to handle the flow control of the code/program. For handling the code, various handlers are used that process the exception and execute the code.

- For example, the Division of a non-zero value with zero will result into infinity always, and it is an exception. Thus, with the help of exception handling, it can be executed and handled.
- In exception handling a throw statement is used to raise an exception. It means when an abnormal condition occurs, an exception is thrown using throw keyword.
- The thrown exception is handled by wrapping the code into the try...catch block. If an error is present, the catch block will execute, else only the try block statements will get executed.
- Thus, in a programming language, there can be different types of errors which may disturb the proper execution of the program.

☞ Types of Errors

While coding, there can be three types of errors in the code :

1. **Syntax Error :** When a user makes a mistake in the pre-defined syntax of a programming language, a syntax error may appear.
2. **Runtime Error :** When an error occurs during the execution of the program, such an error is known as Runtime error. The codes which create runtime errors are known as Exceptions. Thus, exception handlers are used for handling runtime errors.
3. **Logical Error :** An error which occurs when there is any logical mistake in the program that may not produce the desired output, and may terminate abnormally. Such an error is known as Logical error.

JavaScript provides error-handling mechanism to catch runtime errors using try-catch-finally block, similar to other languages like Java or C#.

☞ Syntax

```
try
{
  // code that may throw an error
}
catch(ex)
{
  // code to be executed if an error occurs
}
finally{
  // code to be executed regardless of an error occurs or not
}
```

- **try{} statement :** The code that requires error testing is kept within the try block. If an error occurs, it is passed to the catch{} block, which takes appropriate actions and handles the error. Otherwise, it runs the code that was written within.
- **catch{} statement :** This block handles the error of the code by executing the set of statements written within the block. This block contains either the user-defined exception handler or the built-in handler. This block executes only when any error-prone code needs to be handled in the try block. Otherwise, the catch block is skipped.
- **Finally statement :** Code in the finally block will always be executed regardless of the occurrence of an error. The finally block can be used to complete the remaining task or reset variables that might have changed before error occurred in try block.

Module 2

Example : Error Handling in JS

```
<!DOCTYPE html>
<html>
<body>
    <h1>Demo: Error Handling</h1>

    <p id="errorMessage"></p>

    <script>
        try
        {
            var result = Sum(10, 20); // Sum is not defined
yet
        }
        catch(ex)
        {

    document.getElementById("errorMessage").innerHTML
= ex;
        }
    </script>
</body>
</html>
```

Output

Demo: Error Handling

ReferenceError: Sum is not defined

- In the above example, we are calling function Sum, which is not defined yet. So, try block will throw an error which will be handled by catch block. The "ex" includes error message that can be displayed.
- The finally block executes regardless of whatever happens.

Example : finally Block

```
<!DOCTYPE html>
<html>
<body>
    <h1>Demo: Error Handling</h1>

    <p id="errorMessage">Error: </p>
    <p id="message"></p>

    <script>
        try
        {
            var result = Sum(10, 20); // Sum is not
defined yet
        }
        catch(ex)
        {

document.getElementById("errorMessage").innerHTML +=
ex;
        }
        finally{

document.getElementById("message").innerHTML = "finally
block executed";
        }

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

Output

Demo: Error Handling

Error: ReferenceError: Sum is not defined

finally block executed

Throw statement

Use throw keyword to raise a custom error.

```
<!DOCTYPE html>
<html>
<body>
    <h1>Demo: throw</h1>

    <p id="errorMessage"></p>

    <script>
        try
        {
            throw "Error occurred";
        }
        catch(ex)
        {

    document.getElementById("errorMessage").innerHTML
= ex;
        }

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

Output

Demo: throw

Error occurred

2.12 DHTML JAVASCRIPT

- DHTML stands for **Dynamic Hypertext Markup Language** i.e., Dynamic HTML.
- Dynamic HTML is not a markup or programming language but it is a term that combines the features of various web development technologies for creating the web pages dynamic and interactive.

Difference between HTML and DHTML

Sr. No.	HTML (Hypertext Markup language)	DHTML (Dynamic Hypertext Markup language)
1.	HTML is simply a markup language.	DHTML is not a language, but it is a set of technologies of web development.
2.	It is used for developing and creating web pages.	It is used for creating and designing the animated and interactive web sites or pages.
3.	This markup language creates static web pages.	This concept creates dynamic web pages.
4.	It does not contain any server-side scripting code.	It may contain the code of server-side scripting.
5.	The files of HTML are stored with the .html or .htm extension in a system.	The files of DHTML are stored with the .dhtm extension in a system.
6.	A simple page which is created by a user without using the scripts or styles called as an HTML page.	A page which is created by a user using the HTML, CSS, DOM, and JavaScript technologies called a DHTML page.
7.	This markup language does not need database connectivity.	This concept needs database connectivity because it interacts with users.

- DHTML stands for Dynamic HTML. Dynamic means that the content of the web page can be customized or changed according to user inputs i.e. a page that is interactive with the user. In earlier times, HTML was used to create a static page. It only defined the structure of the content that was displayed on the page. With the help of CSS, we can beautify the HTML page by changing various properties like text size, background color etc. The HTML and CSS could manage to navigate between static pages but couldn't do anything else. If 1000 users view a page that had their information for eg. admit card then there was a problem because 1000 static pages for this application build to work. As the number of users increase, the problem also increases and at some point it becomes impossible to handle this problem.
- To overcome this problem, DHTML came into existence. DHTML included JavaScript along with HTML and CSS to make the page dynamic. This combination made the web pages dynamic and eliminated this problem of creating static page for each user. To integrate JavaScript into HTML, a Document Object Model(DOM) is made for the HTML document. In DOM, the document is represented as nodes and objects which are accessed by different languages like JavaScript to manipulate the document.

Module 2

HTML document include JavaScript

- JavaScript can be included in HTML pages, which creates the content of the page as dynamic. We can easily type the JavaScript code within the <head> or <body> tag of a HTML page. If we want to add the external source file of JavaScript, we can easily add using the <src> attribute.
- Following are some of the tasks that can be performed with JavaScript :
  - Performing html tasks
  - Performing CSS tasks
  - Handling events
  - Validating inputs
- Following are the various examples, which describes how to use the JavaScript technology with the DHTML:

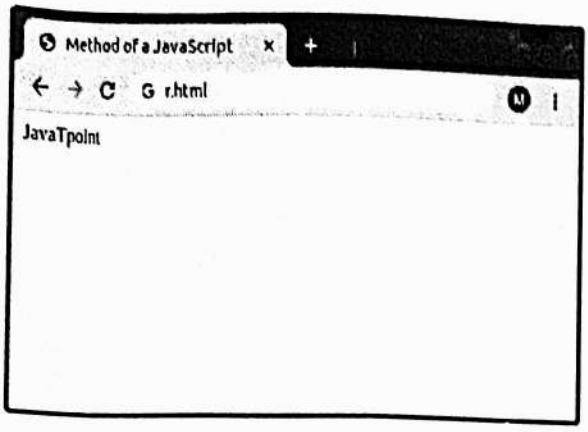
Document.write() Method

The document.write() method of JavaScript, writes the output to a web page.

Example 1 : The following example simply uses the **document.write()** method of JavaScript in the DHTML. In this example, we type the JavaScript code in the <body> tag.

```
<HTML>
<head>
<title>
Method of a JavaScript
</title>
</head>
<body>
<script type="text/javascript">
document.write("JavaTpoint");
</script>
</body>
</html>
```

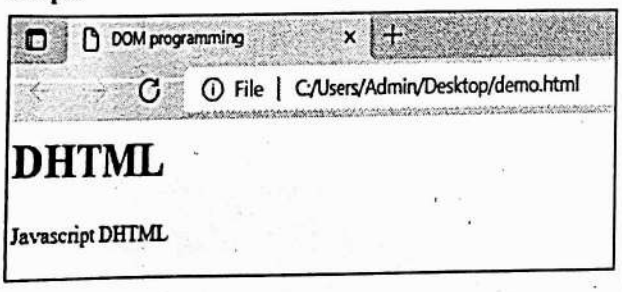
Output



Example 2 : Example to understand how to use JavaScript in DHTML.

```
<html>
  <head>
    <title>DOM programming</title>
  </head>

  <body>
    <h1>DHTML</h1>
    <p id = "demo">Welcome to SJCEM!</p>
    <script style = "text/javascript">
    document.getElementById("demo").innerHTML =
        "Javascript DHTML";
    </script>
  </body>
</html>
```

Output

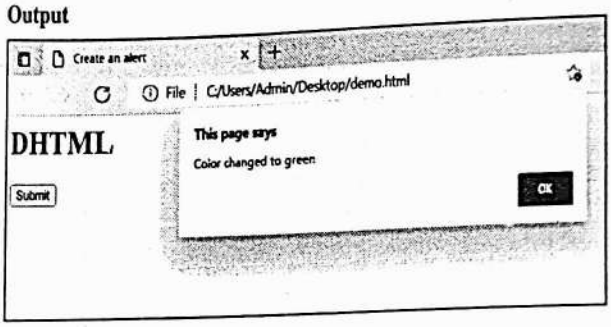


- In the above example, changes the text of paragraph using id. Document is an object of html that is displayed in the current window or object of DOM. The getElementById(id) gives the element id. The innerHTML defines the content within the id element. The id attribute is used to change HTML document and its property. Paragraph content changed by document id.

Example 3 : Creating an alert on click of a button.

```
<html>
  <head>
    <title>Create an alert</title>
  </head>
  <body>
    <h1 id = "para1">DHTML</h1>
    <input type = "Submit" onclick = "Click()"/>
    <script style = "text/javascript">
      function Click() {
        document.getElementById("para1").style.color =
"#009900";
        window.alert("Color changed to green");
      }
    </script>
  </body>
</html>
```

Output



- In this example, creating a function that will be invoked on click of a button and it changes the color of text and display the alert on the screen. Window is an object of current window whose inbuilt method alert() is invoked in Click() function.

2.13 JSON INTRODUCTION

GQ. What is JSON?

- JSON stands for JavaScript Object Notation.
- JSON is a text format which is used for storing and transporting data.
- JSON originated from JavaScript.
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers

- JSON is "self-describing" and easy to understand.
- JSON is easy to read and write than XML.
- JSON is language independent.
- JSON supports array, object, string, number and values.

☞ Why do we use JSON ?

JSON is a lightweight and easy-to-use when compared to other open data interchange options. Following are the advantages of JSON :

1. **Less Verbose :** In contrast to XML, JSON follows a compact style to improve its users' readability. While working with a complex system, JSON tends to make substantial enhancements.
2. **Faster :** The JSON parsing process is faster than that of the XML because the DOM manipulation library in XML requires extra memory for handling large XML files. However, JSON requires less data that ultimately results in reducing the cost and increasing the parsing speed.
3. **Readable :** The JSON structure is easily readable and straightforward. Regardless of the programming language that you are using, you can easily map the domain objects.
4. **Structured Data :** In JSON, a map data structure is used, whereas XML follows a tree structure. The key-value pairs limit the task but facilitate the predictive and easily understandable model.

☞ Features of JSON

1. Simplicity
2. Openness
3. Self-Describing
4. Internationalization
5. Extensibility
6. Interoperability

☞ JSON Vs XML

- Both XML and JSON are widely used today. They are used as data interchange formats and both have been adopted by applications as a way to store structured data.

Module 2

- The following are the differences between the JSON and XML :

Sr. No.	JSON	XML
1.	JSON stands for javascript object notation.	XML stands for an extensible markup language.
2.	The extension of json file is .json.	The extension of xml file is .xml.
3.	The internet media type is application/json.	The internet media type is application/xml or text/xml.
4.	The type of format in JSON is data interchange.	The type of format in XML is a markup language.
5.	It is extended from javascript.	It is extended from SGML.
6.	It is open source means that we do not have to pay anything to use JSON.	It is also open source.
7.	The object created in JSON has some type.	XML data does not have any type.
8.	The data types supported by JSON are strings, numbers, Booleans, null, array.	XML data is in a string format.
9.	It does not have any capacity to display the data.	XML is a markup language, so it has the capacity to display the content.
10.	JSON has no tags.	XML data is represented in tags, i.e., start tag and end tag.
11.	JSON is quicker to read and write.	XML file takes time to read and write because the learning curve is higher.
12.	JSON can use arrays to represent the data.	XML does not contain the concept of arrays.
13.	It can be parsed by a standard javascript function. It has to be parsed before use.	XML data which is used to interchange the data, must be parsed with respective to their programming language to use that.
14.	It can be easily parsed and little bit code is required to parse the data.	It is difficult to parse.
15.	File size is smaller as compared to XML.	File size is larger.
16.	JSON is data-oriented.	XML is document-oriented.
17.	It is less secure than XML.	It is more secure than JSON.

2.13.1 Syntax Rules

- The JSON syntax is a subset of the JavaScript syntax. The syntax is derived from JavaScript object notation syntax and consist of the following rules :
  - Data is written in name or value pairs.
  - Data is separated by commas.
  - Curly braces hold the objects.
  - Square brackets hold the arrays.
- The JSON format is very similar to JavaScript objects. Here, keys must be strings, written with double quotes such as:

{ "name": "abc" }

JSON Example

```
employees.json
{"employees":[
  {"name":"Vivian", "email":"vivianl@gmail.com"},
  {"name":"Shraddha", "email":"shraddham@gmail.com"},
  {"name":"Ronald", "email":"ronaldl@gmail.com"}
]}
```

The XML representation of above JSON example is given below.

```
employees.xml
<employees>
  <employee>
    <name> Vivian </name>
    <email> vivianl@gmail.com </email>
  </employee>
  <employee>
    <name> Shraddha </name>
    <email> shraddham@gmail.com </email>
  </employee>
  <employee>
    <name> Ronald </name>
    <email> ronaldl@gmail.com </email>
  </employee>
</employees>
```

2.13.2 JSON Data Types

In JSON, values must be one of the following data types :

1. String 2. Number 3. Object
4. Arrays 5. Boolean 6. Null

(a) **JSON Strings**

Strings in JSON must be written in double quotes.

Example

```
{"name":"Vidya"}
```

(b) **JSON Numbers**

Numbers in JSON must be an integer or a floating point.

Example

```
{"age":22}
```

(c) **JSON Objects**

Values in JSON can be objects.

Example

```
{
"employee":{"name":"Prajakta", "age":30, "city":"Thane}
}
```

(d) **JSON Arrays**

Values in JSON can be arrays.

Example

```
{
"employees":["Piyu", "Khyati", "Inu", "Praju", "Sanyu",
"More",]
}
```

(e) **JSON Booleans**

Values in JSON can be true/false.

Example

```
{"sale":true}
```

(f) **JSON null**

Values in JSON can be null.

Example

```
{"middlename":null}
```

2.13.3 JSON Object

JSON object holds key/value pair. Each key is represented as a string in JSON and value can be of any type. The keys and values are separated by colon. Each key/value pair is separated by comma.

The curly brace { } represents JSON object.

Let's see an example of JSON object.

```
{
  "employee": {
    "name":     "Sai",
    "salary":   56000,
    "married":  true
  }
}
```

In the above example, employee is an object in which "name", "salary" and "married" are the key. In this example, there are string, number and boolean value for the keys.

☞ JSON Object with Strings

The string value must be enclosed within double quote.

```
{
    "name":     "Vennu",
    "email":    "vennul@gmail.com"
}
```

☞ JSON Object with Numbers

JSON supports numbers in double precision floating-point format. The number can be digits (0-9), fractions (.33, .532 etc) and exponents (e, e+, e-,E, E+, E-).

```
{
"integer": 34,
"fraction": .2145,
"exponent": 6.61789e+0
}
```

☞ JSON Object with Booleans

JSON also supports boolean values true or false.

```
{
"first": true,
"second": false
}
```

☞ JSON Nested Object Example

A JSON object can have another object also. Let's see a simple example of JSON object having another object.

```
{
  "firstName": "Chetana",
  "lastName": "Murudkar",
  "age": 30,
  "address" : {
    "Address": "Sai Nagar",
    "city": "Mumbai",
    "state": "Maharashtra",
    "postalCode": "400140"
  }
}
```

☞ JSON Array

JSON array represents ordered list of values. JSON array can store multiple values. It can store string, number, boolean or object in JSON array.

In JSON array, values must be separated by comma.

The [ ](square bracket) represents JSON array.

☞ JSON Array of Strings

Let's see an example of JSON arrays storing string values.

```
["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"]
```

☞ JSON Array of Numbers

Let's see an example of JSON arrays storing number values.

```
[12, 34, 56, 43, 95]
```

☞ JSON Array of Booleans

Let's see an example of JSON arrays storing boolean values.

```
[true, true, false, false, true]
```

☞ JSON Array of Objects

Let's see a simple JSON array example having 4 objects.

```
{"employees":[
  {"name":"Chetana","email":"gaurim@gmail.com",
"age":31},
  {"name":"Gauri","email":"gaurim@gmail.com","age":31},
  {"name":"Prajakta","email":"prajul@gmail.com",
"age":32},
  {"name":"Vidya","email":"vidyam@gmail.com","age":32}
  {"name":"Shreedevi","email":"shreek@gmail.com",
"age":31},
  {"name":"Yogita","email":"yogin@gmail.com","age":32},
  {"name":"Shraddha","email":"shraddham@gmail.com",
"age":32},
]}
```

Module 2

2.13.4 JSON Function Files

GQ. How to read JSON data using function files.

1: **Create an array of objects.**

Use an array literal to declare an array of objects.

Give each object two properties: display and url.

myArray

```
var myArray = [
{
"display": "JavaScript Tutorial",
"url": http://www.w3schools.com/js/default.asp
},
{
"display": "HTML Tutorial",
"url": http://www.w3schools.com/html/default.asp
},
{
"display": "CSS Tutorial",
"url": http://www.w3schools.com/css/default.asp
}
]
```

2: Create a JavaScript function to display the array.

myFunction()

Create a function myFunction() that loops the array objects, and display the content as HTML links:

```
function myFunction(arr) {
  var out = "";
  var i;
  for(i = 0; i < arr.length; i++) {
    out += '<a href="' + arr[i].url + '">' + arr[i].display
+ '</a><br>';
  }
  document.getElementById("id01").innerHTML = out;
}
Call myFunction() with myArray as argument:
myFunction(myArray);
```

3: **Use an array literal as the argument (instead of the array variable) :**

Call myFunction() with an array literal as argument:

Calling myFunction()

```
myFunction([
{
"display": "JavaScript Tutorial",
"url": http://www.w3schools.com/js/default.asp
},
{
"display": "HTML Tutorial",
"url": http://www.w3schools.com/html/default.asp
},
{
"display": "CSS Tutorial",
"url": http://www.w3schools.com/css/default.asp
}
]);
```

4. Put the function call in an external js file.

Put the function call in a file named **myTutorials.js:**

myTutorials.js

```
myFunction([
{
"display": "JavaScript Tutorial",
"url": http://www.w3schools.com/js/default.asp
},
{
"display": "HTML Tutorial",
"url": http://www.w3schools.com/html/default.asp
},
{
"display": "CSS Tutorial",
"url": http://www.w3schools.com/css/default.asp
}
]);
```

Add the external script to your page (instead of the function call):

```
<script src="myTutorials.js"></script>
```

Complete Example

Following example reads a menu from **myTutorials.js**, and displays the menu in a web page:

```
<!DOCTYPE html>
<html>
<body>
<div id="id01"></div>
```

```
<script>
function myFunction(arr) {
  var out = "";
  var i;
  for(i = 0; i<arr.length; i++) {
    out += '<a href="' + arr[i].url + '">' +
    arr[i].display + '</a><br>';
  }
  document.getElementById("id01").innerHTML = out;
}
</script>
<script src="myTutorials.js"></script>
</body>
</html>
```

Output

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
jQuery Tutorial
JSON Tutorial
AJAX Tutorial
SQL Tutorial
PHP Tutorial
XML Tutorial

2.13.5 JSON Http Request

A common use of JSON is to read data from a web server, and display the data in a web page.

How to read JSON data, using XMLHttp?

Example

Following example reads a menu from **myTutorials.txt**, and displays the menu in a web page:

1. Create an array of objects.

Use an array literal to declare an array of objects.

Give each object two properties : display and url.

myArray

```
var myArray = [
{
"display": "JavaScript Tutorial",
"url": "https://www.w3schools.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "https://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "https://www.w3schools.com/css/default.asp"
}
]
```

2. Create a JavaScript function to display the array.

Create a function myFunction() that loops the array objects, and display the content as HTML links:

myFunction()

```
function myFunction(arr) {
  var out = "";
  var i;
  for(i = 0; i < arr.length; i++) {
    out += '<a href="' + arr[i].url + '">' + arr[i].display
+ '</a><br>';
  }
  document.getElementById("id01").innerHTML = out;
}
```

Call myFunction() with myArray as argument:

myFunction(myArray);

3. Create a text file

Put the array literal in a file named myTutorials.txt:

myTutorials.txt

```
[
{
"display": "HTML Tutorial",
"url": "https://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
```

Module 2

```
"url": "https://www.w3schools.com/css/default.asp"
},
{
"display": "JavaScript Tutorial",
"url": "https://www.w3schools.com/js/default.asp"
},
{
"display": "jQuery Tutorial",
"url": "https://www.w3schools.com/jquery/default.asp"
},
{
"display": "SQL Tutorial",
"url": "https://www.w3schools.com/sql/default.asp"
},
{
"display": "PHP Tutorial",
"url": "https://www.w3schools.com/php/default.asp"
},
{
"display": "XML Tutorial",
"url": "https://www.w3schools.com/xml/default.asp"
}
]
```

4. Read the text file with an XMLHttpRequest

GQ. Write an XMLHttpRequest to read the text file, and use myFunction() to display the array.

XMLHttpRequest

```
var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";

xmlhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
  var myArr = JSON.parse(this.responseText);
  myFunction(myArr);
  }
};

xmlhttp.open("GET", url, true);
xmlhttp.send();
```

Complete Example

```
<!DOCTYPE html>
<html>
<body>
<div id="id01"></div>
<script>
var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";
xmlhttp.onreadystatechange = function() {
 if (this.readyState == 4 && this.status == 200) {
  var myArr = JSON.parse(this.responseText);
  myFunction(myArr);
 }
};
xmlhttp.open("GET", url, true);
xmlhttp.send();
function myFunction(arr) {
 var out = "";
 var i;
 for(i = 0; i < arr.length; i++) {
  out += '<a href="' + arr[i].url + '">' +
  arr[i].display + '</a><br>';
 }
 document.getElementById("id01").innerHTML = out;
}
</script>
</body>
</html>
```

Output

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
jQuery Tutorial
SQL Tutorial
PHP Tutorial
XML Tutorial

2.13.6 JSON SQL

GQ. How to parse JSON in SQL Server ?

Parsing Simple JSON in SQL

Let us take the following JSON as an example:

```
"employeeCode":"S1",
"emailAddress":"shraddham@sjcem.co.in",
"firstName":"Shraddha",
"lastName":"More",
"jobTitleName":"Developer",
"userId":"shraddham"
```

In SQL you must declare JSON as NVARCHAR(MAX) and elect using "OPENJSON" keyword.

```
eclare @JSON NVARCHAR(MAX) = '{
    "employeeCode":"S1",
    "emailAddress":" shraddham@sjcem.co.in ",
    "firstName":" Shraddha ",
    "lastName":" More ",
    "jobTitleName":" Developer ",
    "userId":"shraddham"

ELECT * FROM OPENJSON(@JSON)
```

You can then view the data as a key-value store by passing e JSON as a parameter to the OPENJSON method.

key	value	type
employeeCode	S1	1
emailAddress	shraddham@sjcem.co.in	1
firstName	Shraddha	1
lastName	More	1
jobTitleName	Developer	1
userId	Shraddham	1

If you want to view the same JSON in a column format, you will need to update your query to use WITH clause. Inside the WITH you will need to alias the column name with its type and select the Key attribute from your JSON. Below is the query and its result.

```
SELECT * FROM OPENJSON(@JSON)
WITH(
  EmployeeCode NVARCHAR(MAX) '$.employeeCode',
  EmailAddress NVARCHAR(MAX) '$.emailAddress',
  FirstName NVARCHAR(MAX) '$.firstName',
  LastNamr NVARCHAR(MAX) '$.lastName',
  JobTitleName NVARCHAR(MAX) '$.jobTitleName',
  UserID NVARCHAR(MAX) '$.userId'
)
```

Module 2

EmployeeCode	EmailAddress	FirstName	LastName	JobTitleName	UserID
S1	shraddham@sjcem.co.in	Shraddha	More	Developer	shraddham

2.14 MULTIPLE CHOICE QUESTIONS

Q. 2.1 JavaScript is which type of language?
(a) Object-Oriented (b) Object-Based
(c) Assembly-language (d) High-level ✓Ans. : (b)

Q. 2.2 Inside which HTML element do we put the JavaScript?
(a) < script > (b) < javascript >
(c) < js > (d) < scripting > ✓Ans. : (a)

Q. 2.3 JavaScript is _____ language.
(a) Application (b) Scripting
(c) Programming (d) None of These ✓Ans. : (b)

Q. 2.4 Which company developed JavaScript?
(a) Sun Microsystems (b) IBM
(c) Bell Labs (d) Netscape ✓Ans. : (d)

Q. 2.5 JavaScript is _____ Side Scripting Language.
(a) ISP (b) Server
(c) Browser (d) None of the above ✓Ans. : (c)

Q. 2.6 In JavaScript, what is a block of statement?
(a) Conditional block
(b) Block that combines a number of statements into a single compound statement
(c) Both conditional block and a single statement
(d) Block that contains a single statement ✓Ans. : (b)

Q. 2.7 Where is the correct place to insert JavaScript?
(A) Both the < head > section and the < body > section are correct
(b) The < body > section
(c) The < head > section
(d) None of the above ✓Ans. : (a)

Q. 2.8 Which of the following is not JavaScript Data Types?
(a) Undefined (b) Number
(c) Boolean (d) Float ✓Ans. : (d)

Q. 2.9 Which of the following is the correct statement for comment in JavaScript?

(a) /*This is a comment*/ (b) $This is a comment$
(c) //This is a comment (d) \\This is a comment

✓Ans. : (c)

Q. 2.10 Which of the following code creates an object?

(a) var book = Object();
(b) var book = new Object();
(c) var book = new Book();
(d) var book = new OBJECT();

✓Ans. : (b)

Q. 2.11 JavaScript is designed for the following purpose:

(a) To style HTML pages
(b) To execute Queries related to databases on a server
(c) To add interactivity to html pages
(d) All of the above

✓Ans. : (d)

Q. 2.12 What is the correct JavaScript syntax to write "Hello World"?

(a) System.out.println("Hello World")
(b) document.write("Hello World")
(c) println ("Hello World")
(d) response.write("Hello World")

✓Ans. : (b)

Q. 2.13 How we write "Hello World" in an alert box?

(a) alertBox("Hello World");
(b) msgBox("Hello World");
(c) msg("Hello World");
(d) alert("Hello World");

✓Ans. : (d)

Q. 2.14 To include external JS code inside an HTML document which attribute is used?

(a) src (b) link (c) script (d) ext Ans. : (a)

Q. 2.15 The "function" and "var" are known as:

(a) Keywords (b) Special keywords
(c) Declaration statements (d) Prototypes

✓Ans. : (c)

Q. 2.16 Which of the following function of the String object returns the calling string value converted to upper case?

(a) toLocaleUpperCase() (b) toString()
(c) substring() (d) toUpperCase()

✓Ans. : (d)

Q. 2.17 What are the types of Pop up boxes available in JavaScript?

(a) Confirm (b) Alert
(c) Prompt (d) All of the above

✓Ans. : (d)

Q. 2.18 Which of the following are the errors in JavaScript?

(a) Load time errors (b) Logical Errors
(c) Run time errors (d) All of the above

✓Ans. : (d)

Q. 2.19 Local Browser used for validations on the Web Pages uses

(a) HTML (b) CSS
(c) JS (d) Java

✓Ans. : (c)

Q. 2.20 Comments in JS are ignored by _____.

(a) Browser (b) JVM
(c) Operating System (d) Compiler

✓Ans. : (a)

Q. 2.21 Which of the following is not JavaScript Data Types?

(a) Float (b) Undefined
(c) Boolean (d) Number

✓Ans. : (a)

Q. 2.22 JavaScript is interpreted by

(a) Server (b) Client
(c) Object (d) None of the mentioned

✓Ans. : (b)

Q. 2.23 The statement is an example of: while (3==3) {}

(a) A typographical error
(b) An infinite loop
(c) An illegal JavaScript statement
(d) None of the above

✓Ans. : (b)

Q. 2.24 Which of the following is a syntactically correct for loop?

(a) for (i<=10;i++) (b) for i=1 to 10
(c) for (i=0;i<=10;i++) (d) for (i=0;i<=10)

✓Ans. : (c)

Q. 2.25 Which operator is known as the equality operator, which checks whether its two operators are "equal"?

(a) = (b) ==
(c) === (d) &&

✓Ans. : (b)

Q. 2.26 Which operator is known as the strict equality operator, which checks whether its two operators are identical or not?

(a) = (b) ==
(c) === (d) &&

✓Ans. : (c)

Q. 2.27 Which operator will return false if two value are equal?

(a) ! (b) !=
(c) !== (d) All of the above

✓Ans. : (b)

Q. 2.28 When using comparative operators, if after object to primitive conversion, at least one operand is not a string, both operands are converted to?

(a) String (b) Boolean
(c) Number (d) Object

✓Ans. : (c)

Q. 2.29 Which operator evaluates its operands, then discards the value and returns undefined?

(a) In (b) Comparison
(c) Eval (d) Void

✓Ans. : (d)

Q. 2.30 Variable declared is example of _____ Variable.

```
function message() {
var name = "Bench";
alert(name);
}
```

(a) Local (b) Global
(c) Semi-Global (d) None of the above ✓Ans. : (a)

Q. 2.31 Which of the following is the descendant operator?

(a) .. (b) ... (c) * (d) @ ✓Ans. : (b)

Q. 2.32 What will be the output of the following JavaScript code?

```
< p id="demo"></p>
< script>
var a = 20;
var b = "30";
document.getElementById ("demo").innerHTML=x + y;
</script>
```

(a) 50 (b) 20+30
(c) 2030 (d) error ✓Ans. : (c)

Q. 2.33 What will be the output of the following JavaScript code?

```
< p id="demo">< /p>
< script>
var a = "10";
var b = "10";
var c = a % b;
document.getElementById ("demo").innerHTML = c;
</script>
```

(a) true (b) 1
(c) 0 (d) false ✓Ans. : (c)

Q. 2.34 What among the following is an appropriate event handler for input text among the below options ?

(a) onclick (b) onchange
(c) onkeyup (d) onblur ✓Ans. : (b)

Q. 2.35 What among the following is an appropriate when an event occurs when the user clicks on an element?

(a) onclick (b) onchange
(c) onkeyup (d) onblur ✓Ans. : (a)

Q. 2.36 Consider the below code:

Identify the correct code in order to fetch the value entered in username text field?

```
< body>
< form name="register">
Enter username < input value="John" id="name"
name="username">
</form>
</body>
```

(a) document.register.name.value
(b) document.getElementById ("name").value
(c) document.getElementByName ("name").value
(d) None of the above ✓Ans. : (b)

Q. 2.37 Which of the following statements is/are TRUE about DOM?

(a) W3C standard
(b) Each page can have multiple Document objects
(c) form elements can be accessed using the form object
(d) Both A and C ✓Ans. : (d)

Module 2

Q. 2.38 What is the full form DOM?

(a) Document Object Model
(b) Document Onhalt Model
(c) Document Oriented Model
(d) Document Oriented Management ✓Ans. : (a)

Q. 2.39 Which object is the top of the hierarchy?

(a) Form Control Elements (b) Document Object
(c) Form Object (d) Window Object
✓Ans. : (d)

Q. 2.40 How are the objects organized in the HTML DOM?

(a) list (b) stack (c) queue (d) Hierarchy
✓Ans. : (d)

Q. 2.41 Consider the below code :

Choose the appropriate statement to be inserted at line 2, in order to create a JavaScript object named student?

```
var json= '{ "sId":2001,"sName":"John" }';
//line 2
console.log(student.sId+" "+student.sName);
```

(a) var student=json.parse(json);
(b) var student=JSON.stringify(json);
(c) var student=JSON.parse(json);
(d) var student=json.stringify(json); ✓Ans. : (c)

Q. 2.42 Which among the following regular expression specifies that password should contain at least one capital letter or one digit.

(a) /.*[A-Z].*/.test(password) && /.*[0-9].*/.test(password)
(b) /.*[A-Z].*/.test(password) || /.*[0-9].*/.test(password)
(c) /.*[A-Z][0-9].*/.test(password)
(d) None of the above ✓Ans. : (b)

Q. 2.43 Which of the following string will match the RegEx "((d).(dd))$"?

(a) 223.123 (b) 22.123
(c) 223.12 (d) 2.2 ✓Ans. : (c)

Q. 2.44 Consider the following JavaScript statement containing regular expressions and check if the pattern matches?

var text = "lestfindcourse: 1, 2, 3";
var pattern = /d+/g;

(a) text==pattern (b) text.equals(pattern)
(c) text.test(pattern) (d) pattern.test(text)
✓Ans. : (d)

Q. 2.45 The regular expression to match any one character not between the brackets is _____.

(a) [....] (b) [^]
(c) [^...] (d) [\D] ✓Ans. : (c)

Q. 2.46 What will be the output of the following JavaScript code?

System.out.println(Pattern.matches("[lfc]", "lets"));

(a) true (b) false
(c) undefined (d) l ✓Ans. : (b)

Q. 2.47 Which of the following is correct about features of JavaScript?

(a) It can not Handling dates and time.
(b) JavaScript is a object-based scripting language.
(c) JavaScript is not interpreter based scripting language.
(d) All of the above ✓Ans. : (b)

Q. 2.48 Choose the correct JavaScript syntax to change the content of the following HTML code.

(a) document.getElement ("letsfindcourse").innerHTML = "I am a letsfindcourse";
(b) document.getElementById ("letsfindcourse").innerHTML = "I am a letsfindcourse";
(c) document.getId ("letsfindcourse") = "I am a letsfindcourse";
(d) document.getElementById ("letsfindcourse").innerHTML = I am a letsfindcourse; ✓Ans. : (b)

Q. 2.49 Which of the following is the correct syntax to display "Letsfindcourse" in an alert box using JavaScript?

(a) alert-box("Letsfindcourse");
(b) confirm("Letsfindcourse");
(c) msgbox("Letsfindcourse");
(d) alert("Letsfindcourse"); ✓Ans. : (d)

Q. 2.50 What is the correct syntax for referring to an external script called "LFC.js"?

(a) <script src="LFC.js">
(b) <script source="LFC.js">
(c) <script ref="LFC.js">
(d) <script type="LFC.js">
✓Ans. : (a)

Q. 2.51 What is the original name of JavaScript?

(a) LiveScript (b) EScript
(c) Mocha (d) JavaScript ✓Ans. : (c)

Q. 2.52 JavaScript is designed for following purpose -

(a) to style HTML pages
(b) to execute Queries related to databases on a server
(c) to add interactivity to html pages
(d) All of the above ✓Ans. : (d)

Q. 2.53 What will be the output of the following Javascript code?

var string1 = "Letsfindcourse";
var intvalue = 30;
alert(string1 + intvalue);

(a) Letsfindcourse 30 (b) 30
(c) Letsfindcourse30 (d) Exception ✓Ans. : (c)

Q. 2.54 What should appear at the very end of your JavaScript? The <script language="javascript"> tag

(a) <script> (b) <script>
(c) </script language="javascript">
(d) All of the above ✓Ans. : (b)

Q. 2.55 Which symbol is used for comments in Javascript?

(a) \\ (b) //
(c) * *\ (d) * */ ✓Ans. : (b)

Q. 2.56 Which of them is not the looping structures in JavaScript?

(a) for (b) while
(c) forwhich (d) dowhile ✓Ans. : (b)

Q. 2.57 What does JSON stand for?

(a) JavaScript Object Nomenclature
(b) JavaScript Objective Notation
(c) JavaScript Object Notation
(d) JavaScript Orientated Nomenclature ✓Ans. : (c)

Q. 2.58 The Father of JSON is

(a) Dennis Ritchie (b) Douglas Michel
(c) Rasmus Lerdorf (d) Douglas Crockford
✓Ans. : (d)

2.59 JSON name/value pair is written as

(a) name' : 'value' (b) name = 'value'

(c) name = "value" (d) "name" : "value"

✓Ans. : (d)

2.60 JSON strings have to be in

(a) single quote (b) double quote

(c) single quote or double quote ✓Ans. : (b)

2.61 In the below notation, Employee is of type { "Employee": ["Amy", "Bob", "John"] }

(a) Not a valid JSON string (b) Array

(c) Class (d) Object ✓Ans. : (b)

2.62 Which of the following is not a JSON type?

(a) Object (b) date

(c) Array (d) string ✓Ans.: (b)

2.63 Which of these is correct about the JSON standard?

(a) It is an open standard

(b) It is privately developed

(c) It requires a license to use ✓Ans. : (a)

2.64 What is the file extension of JSON?

(a) .jn (b) .js

(c) .jsn (d) .json ✓Ans. : (d)

2.65 In modern websites what is the common usage for JSON?

(a) To store information remotely.

(b) To send and receive bits of data.

(c) To store information locally. ✓Ans. : (b)

2.66 Which of the following code will return a valid JSON object?

(a) JSON.parse('({"FirstName": "John", "LastName":"Doe"})');

(b) JSON.parse("{'FirstName': 'John', 'LastName':'Doe'}");

(c) JSON.parse("({'FirstName': 'John', 'LastName':'Doe'})");

(d) JSON.parse('{"FirstName": "John", "LastName":"Doe"}'); ✓Ans. : (d)

2.67 Which of these is proper a JSON array?

(a) { "letters" : ["a", "b", "c";] }

(b) { 'letters' : {"a", "b", "c" } }

(c) { "letters" : [a, b, c] }

(d) { "letters" : ["a", "b", "c"] } ✓Ans. : (d)

2.68 In the below notation, Employee is of type { "Employee": { "Name": "Amy", "Age": 25 } }

(a) Object (b) Array

(c) Class (d) Not a valid JSON string

✓Ans. : (a)

Q. 2.69 Which answer represents the following order of TYPES? Object, String, Boolean, Number

(a) "{ }", "a string", "false", "0"

(b) { }, 0, "true", "0"

(c) { }, "0", false, 0

(d) { }, hello, "false", "0" ✓Ans. : (c)

Module 2

Q. 2.70 In this example, what is the TYPE of employee?

{ "employee" : { "type" : "Director", "functions" : ['sales', 'marketing'] } }

(a) String (b) Director

(c) type (d) Object ✓Ans. : (d)

Q. 2.71 Given this JSON example:

{"Employee": { "Name": "Amy", "Age": 25 } }; What is the type of Employee.Age ?

(a) Number (b) Object

(c) Array (d) String ✓Ans. : (a)

Q. 2.72 Which of these is a benefit JSON has over XML?

(a) JSON is more forgiving of poor formatting

(b) JSON has less markup requirements and therefore is lighter than XML

(c) JSON can be written poorly and still be parsed

(d) JSON does not need to be stored in a file to be sent remotely ✓Ans. : (b)

Q. 2.73 JSON elements are separated by

(a) semi-colon (b) line break

(c) comma (d) white space ✓Ans. : (c)

Q. 2.74 Which of the following is a valid JSON string?

(a) { "meals" : { "breakfast" , "lunch" , "dinner" } }

(b) { "meals" : ["breakfast" , "lunch" , "dinner"] }

(c) ["meals" : { "breakfast" , "lunch" , "dinner" }]

(d) [{"meals" : { "breakfast" , "lunch" , "dinner" } }]

✓Ans. : (b)

Q. 2.75 Which of the following code will throw an error?

(a) JSON.parse(null); (b) JSON.parse('{}');

(c) JSON.parse(undefined); (d) JSON.parse('[]');

✓Ans. : (c)

Q. 2.76 Which of these data interchange formats has seen a decline in usage in favor of JSON?

(a) ASCII (b) Plain-text

(c) SQL (d) XML ✓Ans. : (d)

Q. 2.77 What kind of format is JSON, and what does the acronym mean?

(a) A lightweight data-encoding framework. Java Omnipresent Notation.

(b) A lightweight data-interchange format. JavaScript Object Notation.

(c) A lightweight data-interchange format. Java Objective Notion.

(d) A lightweight database framework. JavaScript Object Notation. ✓**Ans. : (b)**

Q. 2.78 In this example, what is the TYPE of employee.functions?

{ "employee" : { "type" : "Director", "functions" : ['sales', 'marketing'] } }

(a) String (b) Object

(c) Array (d) Sales, Marketing ✓**Ans. : (c)**

Q. 2.79 What function will convert a JavaScript object to a JSON string?

(a) JSON.text() (b) JSON.serialize()

(c) JSON.toString() (d) JSON.stringify() ✓**Ans. : (d)**

Q. 2.80 In this example, what would the VALUE of employee.functions[1] be?

{ "employee" : { "type" : "Director", "functions" : ['sales', 'marketing'] } }

(a) null (b) undefined

(c) marketing (d) sales ✓**Ans. : (c)**

Q. 2.81 What is the MIME type of JSON?

(a) application/x-json (b) text/json

(c) application/json (d) application/javascript

✓**Ans. : (c)**

Q. 2.82 Which of these is supported as a JSON Value type?

(a) Infiniti (b) Null

(c) Undefined (d) NaN ✓**Ans. : (b)**

Q. 2.83 When coding a string object in JSON, what must separate the string and the value?

(a) A space (b) A semicolon

(c) A comma (d) A colon ✓**Ans. : (d)**

Q. 2.84 What is the value of obj in the following code?

var obj = JSON.parse('{"fruit": "Apple"}', function(k, v) { if (v == "Apple") return "Orange" else return v; });

(a) { "fruit" : "Apple"} (b) { "fruit" : "Orange"}

(c) {"Orange"} (d) {"Apple"} ✓**Ans. : (b)**

Q. 2.85 What is the value of json in the following code?

var days = {}; days['Monday'] = true; days['Wednesday'] = true; days['Sunday'] = false; var json = JSON.stringify({x: days})

(a){"day":{"Monday":"true","Wednesday":"true","Sunday":"false"}}

(b){"x":{"Monday":true,"Wednesday":true,"Sunday":false}}

(c){"day":{"Monday":true,"Wednesday":true,"Sunday":false}}

(d){"x":["Monday":true,"Wednesday":true,"Sunday":false]}

✓**Ans. : (b)**

Q. 2.86 Which of the following is not a valid way to parse JSON string?

(a) JSON.eval() (b) JSON.parse()

(c) jQuery.parseJSON() (d) eval() ✓**Ans. : (a)**

Q. 2.87 What is the value of json in the following code?

var cars = []; cars[0] = 'Ford'; cars[1] = 'Toyota'; cars[2] = 'BMW'; var json = JSON.stringify({x: cars});

(a) {"x":['Ford','Toyota','BMW']}

(b) {"x":{"Ford","Toyota","BMW"}}

(c) {"x":["Ford","Toyota","BMW"]}

(d) {"cars":["Ford","Toyota","BMW"]} ✓**Ans. : (c)**

Q. 2.88 What types of values can you have in JSON key:value pairs?

(a) Strings, Arrays, and Primitives

(b) Arrays, Primitives, and Objects stored as strings

(c) Strings, Arrays, Primitives and Objects

(d) Strings only ✓**Ans. : (c)**

Q. 2.89 JSON is a programming language?

(a) True (b) False ✓**Ans. : (b)**

Chapter Ends...

□□□

MODULE 3

CHAPTER 3

Back End Development

Syllabus

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.

3.1 Servlet 3-2
- 3.1.1 Introduction to Servlets 3-2
- 3.1.2 CGI vs. Servlets 3-2
- 3.1.3 Advantages of Servlet 3-2
- 3.1.4 Disadvantages of Servlet 3-3

3.2 Java Servlet Architecture 3-3

3.3 Servlet Life Cycle 3-5

3.4 Form GET and POST actions 3-8

3.5 Session Handling 3-10

3.6 Installing and Configuring Apache Tomcat Web Server 3-15

3.7 Database connectivity: JDBC Perspectives 3-21
- 3.7.1 Introduction to JDBC 3-21
- 3.7.2 JDBC Driver 3-21
- 3.7.3 Steps to connect a Java Application to Database 3-22
- 3.7.4 JDBC program example 3-24

3.8 JSP 3-26
- 3.8.1 Introduction to JSP 3-26
- 3.8.2 JSP Life Cycle 3-27
- 3.8.3 Creating a simple JSP Page 3-28
- 3.8.4 Elements of JSP Page 3-28

3.9 Multiple choice questions 3-38

- **Chapter Ends** 3-42

3.1 SERVLET

3.1.1 Introduction to Servlets

GQ. What is a Servlet?

- Servlet is a server side java application which is used to create dynamic web pages.
- A Java Servlet extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on web servers.
- The Servlets executes on a web or application server. It works as an interface between a client request which comes from a web browser or other HTTP client and server side applications or databases.
- Servlets are used to accept input from end users through the form, presents the records from server database or any other storage media and create dynamic web pages.
- Before Servlet, there was a language known as CGI (Common Gateway Interface) which works as a server side language to create dynamic content. Servlets serve the similar purpose as of the Common Gateway Interface but it offers several advantages over CGI.
- Performance of Servlet is significantly better than the CGI.

3.1.2 CGI vs. Servlets

- Before servlets, we had CGI i.e. Common Gateway Interface. It is a standard way for a web server to pass a user's request to an application program and receives the response to forward to the user. When the user requests a web page, the server sends back the requested page.
- However, when a user fills out a form on a web page and sends it in, it is processed by an application program. The web server typically passes the form information to a small application program.
- This program processes the data and sends back a confirmation message. This process of passing data back and forth between the server and the application is called the Common Gateway Interface (CGI).

Parameter	Common Gateway Interface	Servlets
Basic	Programs are written in the native OS.	Programs employed using Java.
Platform dependency	Platform dependent	Does not rely on the platform
Creation of process	Each client request creates its own process.	Processes are created depending on the type of the client request.
Conversion of the script	Present in the form of executables (native to the server OS).	Compiled to Java Byte code.
Runs on	Separate process	JVM
Security	More vulnerable to attacks.	Can resist attacks.
Speed	Slower	Faster
Processing of script	Direct	Before running the scripts it is translated and compiled.
Portability	Cannot be ported	Portable

3.1.3 Advantages of Servlet

GQ. What are the advantages of Servlet ?

1. **Persistent** : Servlets remain in memory until explicitly destroyed. This helps in serving several incoming requests. Servlets establishes connection only once with the database and can handle several requests on the same database. This reduces the time and resources required to establish connection again and again with the same database.
2. **Portable** : Since servlets are written in Java, they are portable. That is, servlets are compatible with almost all operating systems. The programs written on one operating system can be executed on other operating system.
3. **Powerful** : Servlet can handle number of complex tasks which were difficult for CGI. For example Servlet can directly interact with web server, Servlets can share data or information among each other, Servlets can easily make database connection pools which can be implemented easily.

   Servlet support the mechanism of session tracking with the help of which it is possible to maintain and track status and information from request to request.

4. **Efficiency :** The invocation (calling) of Servlet is highly efficient than the CGI scripts. When the Servlet is loaded in the server, it remains persistent in the memory of server as a single object instance. The initialization code of Servlet is executed only the first time when it is loaded by the web server. Afterword for every request only the service() method of Servlet gets called.

5. **Safety :** As servlets are part of java, they inherit the strong type safety of JDK. The automatic garbage collection mechanism and lack of pointers in the Java environment protect the servlets from the problems of memory management. With the exception handling mechanism, it is easy to handle errors in Servlet.

6. **Integration :** Servlets are strongly integrated with the server. Server can be used by the Servlet to perform various operations like translating the file paths, checking authorization, MIME type mapping and perform logging etc.

7. **Extensibility :** The Servlet API is designed purposely in such a manner that it can be easily extensible. Java is robust, well-designed, object oriented language which supports the extended features. As Servlets are created in Java, they can take various such advantages of java and can be extended from existing class to give the ideal solutions as per client requirements. Hence Servlets are extensible.

8. **Inexpensive :** Servlet are executed with the help of web servers which are available without any cost for personal as well as commercial purpose.

9. **Secure :** Servlets are server side components which run through the web servers. Servlets can inherit the security of the web server. Servlets can also have security of Java Security Manager.

10. **Performance :** As compare to CGI, Servlets are relatively very faster because in CGI for every script, a new process is generated which takes lot of time for execution. While in Servlets, only new thread is created for new request. A single Servlet can handle multiple requests simultaneously.

    The initialization process which takes lot times is performed only the first time when Servlet is loaded and remains in memory till times out or server shut downs. Later on handling new requests is only the matter of calling service() method for Servlet. Hence the performance of Servlet to give response to client request is higher.

11. **Server-independent :** Servlets are compatible with any web server available today. Most of the software vendors today support servlets within their web server products. On the other hand, some of the server side technologies like server side JavaSricpt and ASP can run on only selected web servers. The CGI is compatible with the web server that has features to supports it.

12. **Protocol-independent :** Servlets can be created to support any of the protocols like FTP commands, Telnet sessions, NNTP newsgroups, etc. It also provides extended support for the functionality of HTTP protocol.

3.1.4 Disadvantages of Servlet

GQ. What are the disadvantages of servlets?

1. The process of Servlet designing is complicated and slows down the application.
2. The complex business logics written make it difficult to understand the Servlet.
3. To run the Servlets, there is requirement of Java Runtime Environment on the server.
4. It is very difficult for enhancement and bug fixing in Servlet applications.
5. The Servlet technology has requirement of comparatively more steps to develop. Also Servlet need too long time for development.

Module 3

3.2 JAVA SERVLET ARCHITECTURE

Servlets are used to perform client request and response tasks dynamically. Servlets can be used to perform tasks like-

- Control the flow of the application.
- Generate dynamic web content.
- Server-side load balancing.
- Implement business logic.

Components of Servlet Architecture

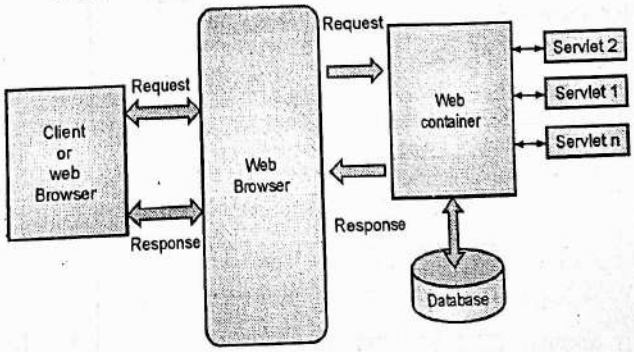


Fig. 3.2.1 : Components of Servlet Architecture

1. **Client :** In this architecture, the web browser acts as a Client. Client or user connected with a web browser. The client is responsible for sending requests or HttpRequest to the web server and processing the Web server's responses.

2. **Web Server** : Web server controls how web user access hosted files, and it's responsible for processing user request and responses. Here server is software it understands URLs and HTTP protocol. Whenever a browser needs to host a file on the web server, process client request using an HTTP request; if it finds the requested file sends it back to the browser through HTTP response. There are two types' web servers Static and Dynamic web servers.in a static web server, it sends the file as it is, but in a dynamic web, the server-hosted file is updated before it is sent to the browser.

3. **Web Container** : A web container is a component in the web server it interacts with Java Servlets. A web container is responsible for managing the lifecycle of servlets, and it also performs the URL mapping task. Web container handles the requests of servlets, JSP and other files at the server-side. The important tasks performed by servlets are loading and unloading servlets, creating and managing requests and response objects, and performing servlet management's overall task.

- A Servlet is a class, which is implemented from the inbuilt interface javax.servlet.Servlet. However in our Servlet application, rather than directly implementing the javax.servlet.Servlet interface, we can inherit our class from such a class which has implemented this interface
- There are two main classes provided by java which are extended from javax.servlet.Servlet.

☞ Classes of Java

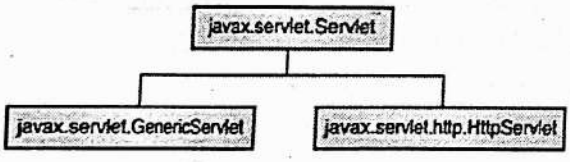


Fig. 3.2.2 : Classes of Java

The javax.servlet package has a rich set of classes and interfaces for writing servlets.

☞ Architecture of package

The architecture of the package is described below :

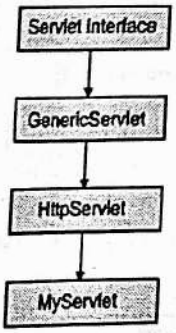


Fig. 3.2.3 : Architecture of package

(a) Servlet Interface

To create a Servlet, we have to implement the Servlet interface. This interface can be implemented by two ways directly or indirectly by extending either of the class GenericServlet or HttpServlet.

(b) GenericServlet

The GenericServlet class is implemented from Servlet interface. To create Servlet, if we extend the GenericServlet class then it is compulsory for us to implement the service method which is declared as an abstract method by java.

(c) HTTPServlet

The HTTPServlet class is extended from GenericServlet class. To create Servlet, if we extend the HTTPServlet class then there is no need to implement the service method as HTTPServlet has already implemented it.

(d) MyServlet

This is our user defined class used to create the Servlet.

1. Client Interaction
2. Servlet Execution

(i) Client Interaction

When a Servlet accepts a call from a client, it receives two objects:

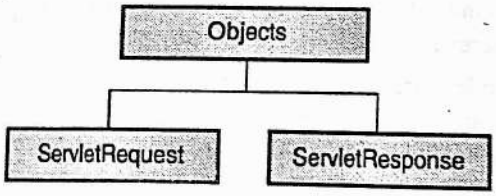


Fig. 3.2.4 : Objects

1. **A ServletRequest** - It encapsulates the communication from the client to the server.
2. **A ServletResponse** - It encapsulates the communication from the server back to the client.

- The ServletRequest interface can access information like the names of the parameters passed in by the client, the protocol (scheme) which is set by the client, and the remote host name which sends the request and the server which received it.
- It also provides the input stream - ServletInputStream which is used by the Servlets to get data from clients. The ServletResponse interface provides the Servlet methods for giving response to the client.

- It permits the Servlet to set the length and MIME type of the content which is to be replied. It also provides an output stream : ServletOutputStream, and a writer which is used by the Servlet to send the reply to client.

(ii) Servlet Execution

Following Fig. 3.2.5 shows the execution of Servlet when client (browser) makes a request to the web server.

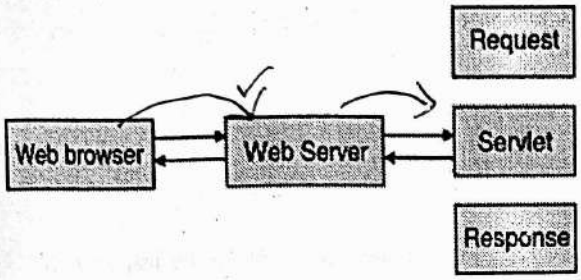


Fig. 3.2.5 : Servlet Execution

Following are the steps how servlet request has been processed

- The web browser (client) sends a request to the web server.
- Web server accepts the request and forwards it to the web container.
- Web container searches web.xml file for request URL pattern and gets the address of the servlet.
- If the servlet is not yet instantiated, it will be instantiated and initialized by calling the init() method.
- The container calls public service() by passing ServletRequest and ServletResponse objects.
- Public service() method typecast ServletRequest and ServletResponse object to HttpServletRequest and HttpServletResponse objects respectively.
- Public service() method calls protected service().
- Protected service() method checks the client request & corresponding do___() method is called.
- The request is handled by sending the result generated by do___() to the client.

3.3 SERVLET LIFE CYCLE

- The lifecycle of a Servlet involves creating an instance of the Servlet, associating the Servlet with initialization information, dispatching request to the servlet instance and finally destroying the servlet instance. The lifecycle of a servlet begins when it is loaded into application server memory and ends when the servlet is terminated or reloaded.

The Servlet life cycle mainly includes the following four stages,

1. Loading and Instantiation
2. Initialization
3. Servicing the Request
4. Destroying the Servlet

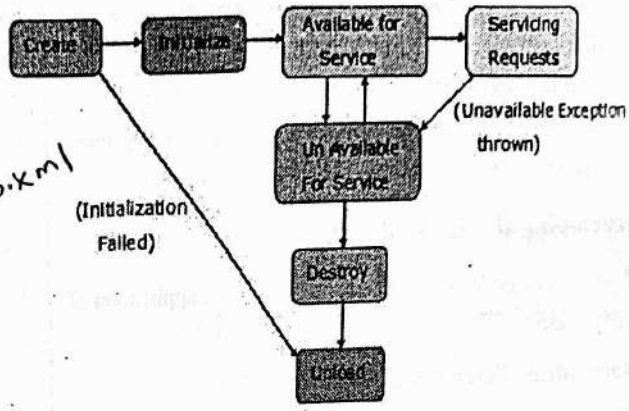


Fig. 3.3.1 : Life Cycle of a Servlet

Module 3

1. **Loading and Instantiation** : The Servlet container (Server) loads the Servlet during startup or when the first request is made. The loading of the Servlet depends on the attribute <load-on-startup> of web.xml file. If the attribute <load-on-startup> has a positive value then the Servlet is load with loading of the Container (Server) otherwise it load when the first request comes for service. After loading of the Servlet, the Container (Server) creates the instances of the Servlet.
2. **Initialization** : After creating the instances, the Servlet Container (Web Server) calls the init() method and passes the Servlet Initialization Parameters to the init() method. The init() must be called by the Servlet Container before the Servlet can service any request. The Initialization Parameters persist until the Servlet is destroyed. The init() method is called only once throughout the life cycle of the Servlet. The Servlet will be available for service if it is loaded successfully otherwise the Servlet container unloads the Servlet.
3. **Servicing the Request** : After successfully completing the initialization process, the Servlet will be available for service. Servlet creates separate threads for each request. The Sevlet container calls the service() method for servicing any request. The service() method determines the kind of request and calls the appropriate method (doGet() or doPost()) for handling the request and sends

response to the client using the methods of the response object.

4. **Destroying the Servlet** : If the Servlet is no longer needed for servicing any request, the Servlet container calls the destroy() method . Like the init() method this method is also called only once throughout the life cycle of the Servlet. Calling the destroy() method indicates to the Servlet container not to send any request for service and the Servlet releases all the resources associated with it. Java Virtual Machine claims for the memory associated with the resources for garbage collection.

Developing the Servlet application

Here we are going to develop a simple Servlet application of servlet life cycle.

- **Home.html :** First we create a html file

```
<html>
<head>
<title>Servlet Example</title>
</head>
<body bgcolor="cyan">
<form action="LifeCycleServlet.java"><br/><br/>
<center>
<input type="submit" value="Invoke Life Cycle Servlet"/>
</center>
</form>
</body>
</html>
```

- **LifeCycleServlet.java :** Then we create a servlet file.

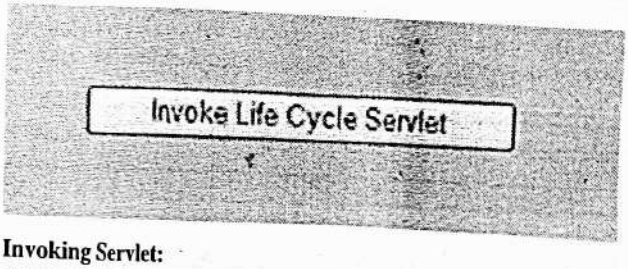
```
package my;
import javax.servlet.*;
import javax.servlet.Servlet;

public class LifeCycleServlet implements Servlet
{
 public void init(ServletConfig sc)
 { config=sc;
 System.out.println("In Init");
 }
 public void service(ServletRequest req,ServletResponse
res)throws ServletException,java.io.IOException
 {
 java.io.PrintWriter out=res.getWriter();
 out.println("Hello from lifecycle servlet");
 System.out.println("in Service");
 }
 public void destroy(){
 System.out.println("In Destroy");
 }
 public String getServletInfo(){return "LyfeCycleServlet";
 }
 public ServletConfig getServletconfig(){
 return config;
 }
 private ServletConfig config;
}
```

- **web.xml :** Then we create a web.xml file for mapping the servlet file with html

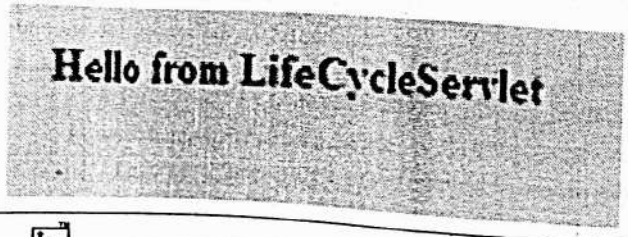
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
xmlns="http://java.sun.com/xml/ns/javaee"xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
<servlet>
<servlet-name>Ls</servlet-name>
<servlet-class>my.LifeCycleServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Ls</servlet-name>
<url-pattern>/lifeCycleServlet</url-pattern>
</servlet-mapping>
</web-app>
```

Output

Home.html



Invoking Servlet:



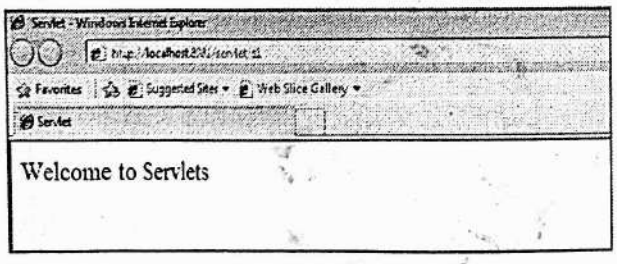
Simple Servlet Example

➢ **Program 3.3.1 :** Write a program of Servlet.

```
import java.io.*;
import javax.servlet.*;
public class s1 extends
GenericServlet
{
  public void service(ServletRequest request,
ServletResponse response)
  throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  out.println("<html>");
  out.println("<head>");
  out.println("<title>Servlet</title>");
  out.println("</head>");
  out.println("<body>");
  out.println("<font color=red size=5>");
  out.println("Welcome to Servlets");
  out.println("</font>");
  out.println("</body>");
  out.println("</html>");
  out.close();
  }
}
```

Here the Servlet class is inherited from GenericServlet, hence it is necessary for us to define the service method.

Output



➢ **Program 3.3.2 :** Create an HTML page that accepts the user's name. It should be sent to Servlet. The Servlet should display a welcome message with the name specified.

HTML Script

```
<html>
<head>
<title>Servlet</title>
</head>
<body>
<font color="blue" size=5>
<form
method="post"ACTION="http://localhost:8081/servlet/s2">
 Enter your name <input type="text"
name="n"><br><br>
<input type="submit" value="Submit">
</form>
</font>
</body>
</html>
```

Explanation of HTML Script (s2.html file)

The form method takes two arguments.

1. First is **method** to which we have to assign the name of method to send the data like get, post etc depending upon the requirement.
2. Second argument is ***action*** to which we provide the URL of Servlet to which we want to send the data.

Module 3

Servlet Code

```
import java.io.*;
import javax.servlet.*;
public class s2 extends GenericServlet
{
  public void service(ServletRequest request,
ServletResponse response)
  throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html><head><title>Servlet</title>
</head>");
  out.println("<body bgcolor=skyblue>");
  out.println("<font color=red size=5>");
  String nm = request.getParameter("n");

  out.println("Hello "+nm + " Welcome to Servlet");
  out.println("</font></body></html>");
    out.close();
  }
}
```

Explanation of Servlet code (s2.class file)

response.setContentType("text/html"); -

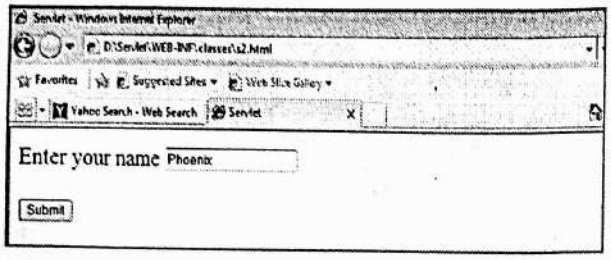
The setContentType() is method of ServletResponse. It is used to set the type of content which we will return to the browser on client side.

String nm = request.getParameter("n");

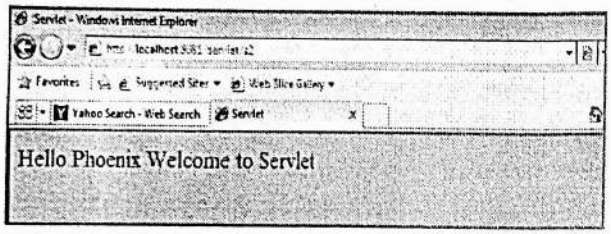
The getParameter() is method of ServletRequest. It takes argument as name of HTML form control and returns data provided by user in it or the status of form control like checked or unchecked. Here the "n" is the name of text field in HTML form.

Output

HTML



Servlet



3.4 FORM GET AND POST ACTIONS

- doGet() and doPost() methods are used to transfer data between client side HTML application and server side Servlet.
- The important difference between them is doGet() handles requests which are sent using get() method by the HTML application while doPost() handles requests which are sent using post() method.

Handling GET and POST Requests

The first thing we are going to do is to create in our site a page "login.html" with the following content:

login.html

```
<html>
<body>
    <form action="login" method="get">
        <table>
            <tr>
                <td>User</td>
                <td><input name="user" /></td>
            </tr>
            <tr>
                <td>password</td>
                <td><input name="password" /></td>
            </tr>
        </table>
        <input type="submit" />
    </form>
</body>
</html>
```

Then, we create a Servlet which receives the request in /login, which is the indicated direction in the action attribute of the tag <form> of login.html

LoginServlet.java

```
package com.edu4java.servlets;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp)
        throws ServletException, IOException {
        String user = req.getParameter("user");
        String pass = req.getParameter("password");
        if ("edu4java".equals(user) &&
"eli4java".equals(pass)) {
            response(resp, "login ok");
        } else {
            response(resp, "invalid login");
        }
    }

private void response(HttpServletResponse resp, String msg)
        throws IOException {
    PrintWriter out = resp.getWriter();
    out.println("<html>");
```

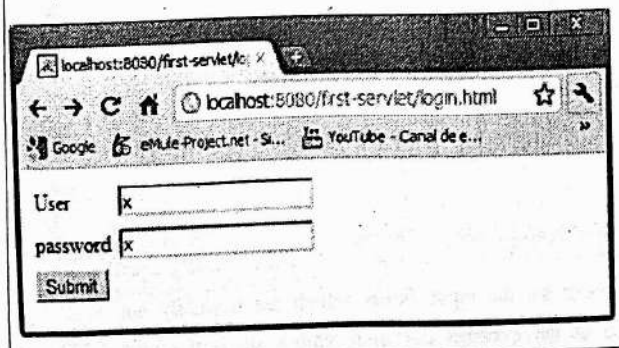
```
    out.println("<body>");
    out.println("<t1>" + msg + "</t1>");
    out.println("</body>");
    out.println("</html>");
}
```

We compiled this Servlet and we include LoginServlet.class in the folder /WEB-INF/classes. We modify web.xml to link /login with this Servlet.

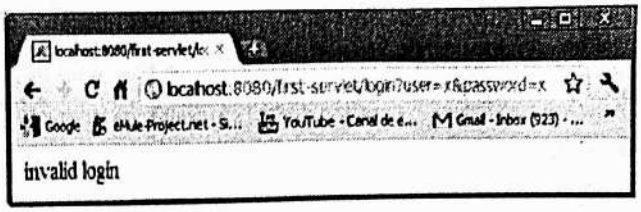
Veb.xml

```
:web-app>
  <servlet>
    <servlet-name>timeservlet</servlet-name>
    <servlet-
ass>com.edu4java.servlets.FirstServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>login-servlet</servlet-name>
    <servlet-
ass>com.edu4java.servlets.LoginServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>timeservlet</servlet-name>
    <url-pattern>/what-time-is-it</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>login-servlet</servlet-name>
    <url-pattern>/login</url-pattern>
  </servlet-mapping>
:/web-app>
```

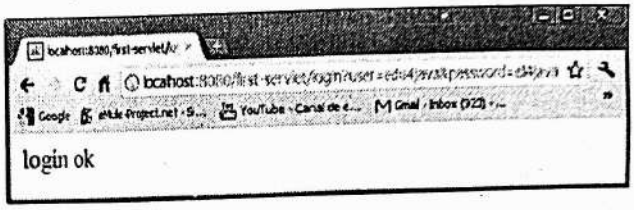
We restart the server, open the page login.html, write an "x" n user, write an "x" in password and click on the submit button.



- As a response we receive



- Above output indicating that the login has failed. If we repeated the operation with "edu4java" as user and "eli4java" as password we obtain



- The problem here is that the password is visible in the URL- .../login?user=edu4java&password=eli4java, it will be kept in the history of the browser and anybody who access the browser after us can obtain it easily. This can be solved changing the way of sending the form and using the method POST in login.html.

Module 3

login.html

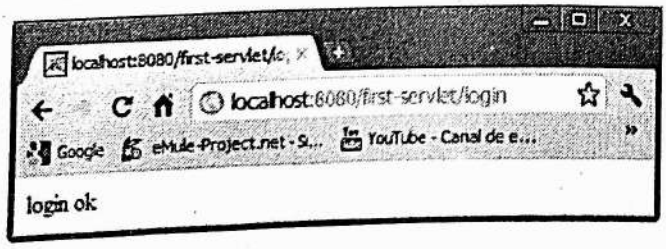
```
<html>
<body>
    <form action="login" method="post">
        <table>
            <tr>
                <td>User</td>
                <td><input name="user" /></td>
            </tr>
            <tr>
                <td>password</td>
                <td><input name="password" /></td>
            </tr>
        </table>
        <input type="submit" />
    </form>
</body>
</html>
```

LoginServlet.java

```
package com.edu4java.servlets;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req,
HttpServletResponse resp)
            throws ServletException, IOException {
        String user = req.getParameter("user");
        String pass = req.getParameter("password");
        if ("edu4java".equals(user) &&
"eli4java".equals(pass)) {
            response(resp, "login ok");
        } else {
            response(resp, "invalid login");
        }
    }

    private void response(HttpServletResponse resp, String
msg)
            throws IOException {
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<t1>" + msg + "</t1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

- The only change is the replacement of doGet for doPost. After the recompilation, the deployment of the Servlet, the restart of the server and the reuse of login.html we obtain



- We can see that in above output window the parameters of the URL have disappeared.

3.5 SESSION HANDLING

GQ. What is session?

- A session can be defined as a series of transactions which are related with each other between a single client and the web servers, which executes over a period of time.
- The session could be series of transactions that a user makes while making updations in a stock portfolio, or the series of requests that are made to check an email account with the help of browser-based email service.
- The session may have more than one request to the same Servlet or of the requests to different resources on the same web site.
- HTTP is a stateless protocol; hence the web server does not have knowledge about which session a specified request belongs to.
- We can get the client to identify itself every time a request is made and store and retrieve data related to that specific client in some storage media on the server. It is possible to send the data to the client, and make the client to send it back with request whenever a request is made.

Ways for Tracking the Session and Maintaining State

GQ. List the session tracking techniques.

There are different ways to track the session and maintain state of a client.

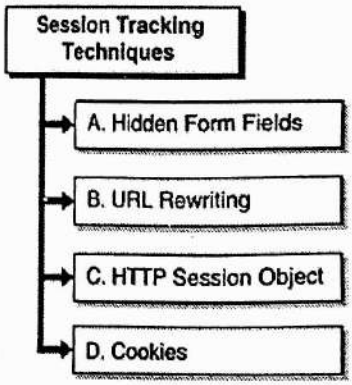


Fig. 3.5.1 : Session tracking techniques

(A) Hidden Form Fields

GQ. Explain Hidden Form Fields with example. List their advantages and disadvantages.

- Hidden fields are the input fields which are basically not displayed on the webpage but their values are sent to the

Servlet just like any other input fields. It is possible to track the session data by storing it in hidden form fields and later on retrieving it with the help of HttpServletRequest object.

xample

```
Input type="Hidden" name="pass'
ALUE="SessionData">
```

Now when the form is submitted, the specified name and value of the created hidden form fields are included in the data of GET or POST method.

In case of hyperlinks, as the form is not submitted like submit button, the Hidden Form Field is not suggestive way to track the session.

Advantage of Hidden Form Field

It will work perfectly every time without considering whether okie is disabled or not.

Disadvantages of Hidden Form Field

The maintenance of Hidden Form Field is done at server side.

) There is need of extra form submission for each page.

i) Sending only text type of data is allowed.

(B) URL Rewriting

Q. Explain "URL Rewriting" with example. List its advantages and disadvantages.

At the end of each URL (Uniform Resource Locator), we can d some extra text related to session. The session identifier is sociated with session data by the server.

xample

http://PhoenixGlobe.com/index.htm;sessionid = 13579, the session identifier is attached as sessionid = = 13579 which is later on can be accessed at server side to identify the client.

This is good option for session tracking. This is generally helpful when browsers do not support cookies or cookies are disabled. The problem with URL re-writing is that dynamically we have to generate each URL to assign a session ID. This is applicable for simple HTML static pages also.

Advantages of URL Rewriting

It will work perfectly every time without considering whether cookie is disabled or not.

) No need of extra form submission for each page.

Disadvantages of URL Rewriting

(i) It will work only with links.

(ii) Sending only text type of data is allowed

(C) HTTP Session Object

- Servlet provides an interface HttpSession for session handling and store information about a particular user. This interface is used by the Servlet container to create session between client and HTTP server. The session remains in the memory for a specified time interval across several request through different web pages by the user.
- Getting the HttpSession object
- There are two ways provided by the HttpServletRequest interface to get the HttpSession object.

```
1. public HttpSession getSession()
2. public HttpSession getSession(boolean create)
```

1. **public HttpSession getSession() :** Returns the current session which is associated with the specified request. If the session is not available for the request, then it creates the session and then returns it.
2. **public HttpSession getSession(boolean create) :** Returns the current session which is associated with the specified request. If the session is not available for the request and the value of *create* attribute is true, then it creates the session and then returns it.

Module 3

Commonly used methods of HttpSession interface

1. **public String getId() :** Returns a string which has the unique identifier value.
2. **public long getCreationTime() :** Returns the creation time of session measured in milliseconds.
3. **public long getLastAccessedTime() :** Returns the time when the client sent the last request associated with the particular session, measured in milliseconds.
4. **public void invalidate() :** Invalidates the particular session then unbinds all the objects bound to it.

Program using HttpSession

- In this program, we will set the attribute in the session scope in one Servlet and retrieve this value from the same session scope in different Servlet.

- HttpSession interface provides method setAttribute() to set the attribute in the session scope, and to get attribute getAttribute() method is used.

Session.html

```
<html><head>
<title>Session Tracking</title>
</head>
<body>
<font color="blue" size=5>
<form method="get"
ACTION="http://localhost:8081/servlet/ServletOne">
Enter the user Name:<input type="text"
name="user_name"/>
<br/> <br/>
<input type="submit" value="Submit"/>
</form>
</font>
</body>
</html>
```

ServletOne.java

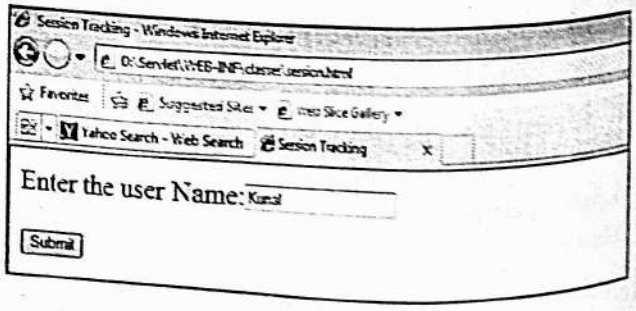
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletOne extends HttpServlet {

public void doGet(HttpServletRequest request,
HttpServletResponse response){
    try{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.print("<font size=6>");

    String nm = request.getParameter("user_name");
    out.print("Welcome To our Web "+nm);

    HttpSession session = request.getSession();
    session.setAttribute("user",nm);
   out.print("<br><br><a
href='http://localhost:8081/servlet/ServletSecond'>Click To
Proceed</a>");
    out.print("</font>");

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

ServletSecond.java

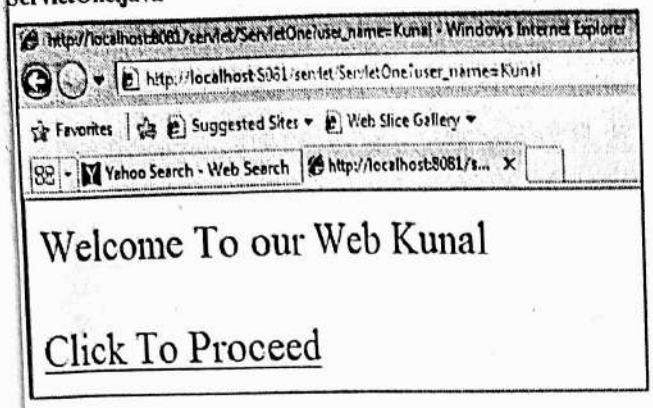
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletSecond extends HttpServlet
{

public void doGet(HttpServletRequest request,
HttpServletResponse response)
{
    try
{

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

    HttpSession session = request.getSession(false);
    String nm = (String)session.getAttribute("user");
      out.print("<font size=6>");
    out.print("Hello " + nm + " welcome to the next page");
    out.print("</font>");

    out.close();

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

 }
```

Output

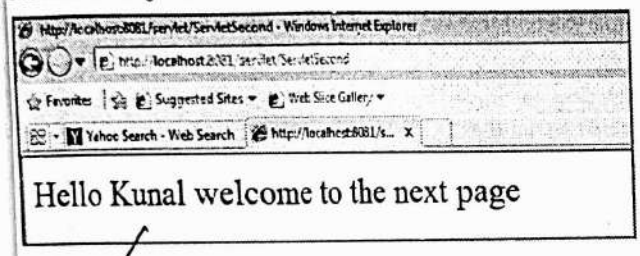
Session.html



ServletOne.java



ServletSecond.java



(D) Cookies

GQ. How cookies are used to track a session?

- Cookies are the small piece of information which is sent by the web server to the client. When next request is made by client, it adds the cookie to the request header. This can be utilized to keep **track of the session.**
- **Advantage :** Cookies are considered as the simplest technique used in session management for storing client state.
- **Storage :** Cookies are stored on client's computer. They have a specific lifespan and are automatically destroyed by the client browser when the lifespan completes.
- We can maintain a session with cookies very efficiently but if the client disables the cookies, then it won't work.
- The Cookie class of the Servlet API is used to create cookies.
- The addCookie() method is used to add the cookies to response object. This method transfers the cookie information over the HTTP response stream.

➢ **Program 3.5.1 :** Write a program to set an expiry time/data for cookie.

HTML

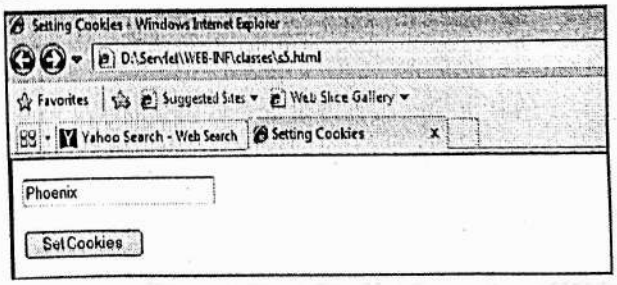
```
<html><head>
<title>
Setting Cookies
</title>
</head>
<body>
<form method ="post"
action="http:\\localhost:8081\servlet\s5">
<input type="text" name="nm" size=25><br><br>
<input type="submit" value="Set Cookies">
</form>
</body>
</html>
```

Servlet

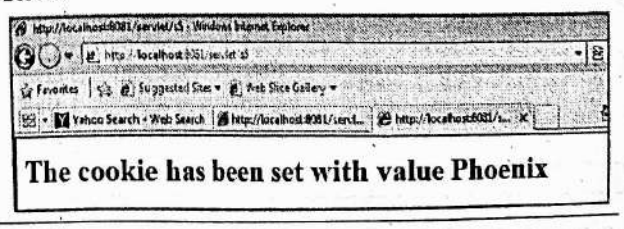
```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class s5 extends HttpServlet
{
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
  throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    String data=request.getParameter("nm");
    Cookie ck=new Cookie("cookienm",data);
ck.setMaxAge(2*24*60*60);
    response.addCookie(ck);
out.println("<b><font size=6>The cookie been set with
value "+data);
  }
}
```

Output

HTML



Servlet



Module 3

➢ **Program 3.5.2 :** Write a program of retrieving cookies.

getCookies() method is used to access the cookies that are added to response object.

HTML

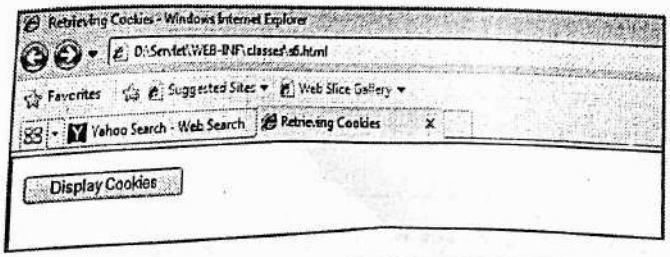
```
<html>
<head>
<title>Retrieving Cookies</title>
</head>
<body>
<form method ="post"
action="http:\\localhost:8081\servlet\s6">
<input type="submit" value="Display Cookies">
</form>
</body>
</html>
```

Servlet

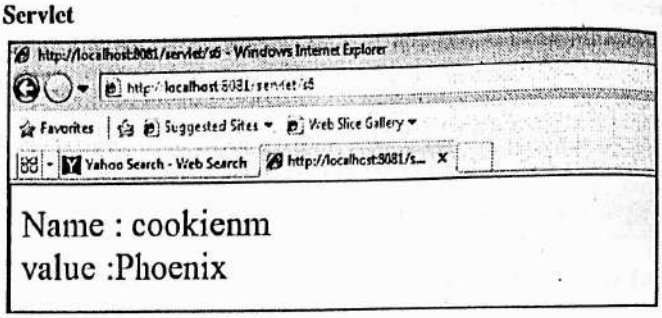
```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class s6 extends HttpServlet
{
public void doPost(HttpServletRequest request,
HttpServletResponse response)
  throws ServletException, IOException
  {
  Cookie ck[] = request.getCookies();
response.setContentType("text/html");
PrintWriter out=response.getWriter();
for(int i=0;i<ck.length;i++)
{
String nm=ck[i].getName();
String val= ck[i].getValue();
out.println("<font size=6>Name : "+nm+"<br>value
:"+val);
}
}
}
```

Output

HTML



Servlet



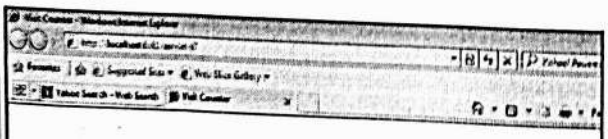
➢ **Program 3.5.3 :** Write a program/example to count visit counter / hit counter of a website using session.

Servlet

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class s7 extends HttpServlet
{

  protected void doGet(HttpServletRequest req,
HttpServletResponse res)

  throws ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Visit Counter</title>");
    out.println("</head>");
    out.println("<body>");
    Integer count;
    HttpSession session = req.getSession(true);

    String heading;
    count =(Integer)session.getAttribute("accessCount");
    if(count==null)
    {
      count = new Integer(1);
```

```
    heading = "Welcome To our Website";
  }
  else
   {
    count = new Integer((count.intValue()) + 1);
    heading = "Welcom Again..";
  }
  session.setAttribute("accessCount",count);
  out.println("<font size=6>");
  out.println("<br>"+heading);
  out.println("<br>SessionID : "+session.getId());
  out.println("<br>Last accesssed : "+new
   Date(session.getLastAccessedTime()));
  out.println("<br>Count : "+count);
  out.println("</body>");
  out.println("</html>");
  out.close();
 }

 protected void doPost(HttpServletRequest req,
HttpServletResponse res)
 throws ServletException, IOException
 {
   doGet(req, res);
 }
}
```

Output



Welcome To out Website
SessionID : BCAFB72446483E0699EAAB49A971C8C4
Last accesssed : Thu Aug 10 11:45:55 GMT 2017
Count : 1

Advantages and disadvantages of cookies

Advantages

(i) It is the simplest technique used to maintain the state.

(ii) Cookies are maintained at client side.

Disadvantages

(i) It will not work if the user disables the cookie from the browser.

(ii) Cookie object can sent only text type of information.

3.6 INSTALLING AND CONFIGURING APACHE TOMCAT WEB SERVER

Module 3

What is Tomcat?

- Apache Tomcat installs several Java Enterprise Edition specs including Java Servlet, JavaServer Pages, Java EL, and WebSocket. It provides for a "pure Java" HTTP web server environment in which Java code can run.
- Tomcat 9 is the latest version of Apache's Tomcat service and can easily be installed on Windows to serve as a convenient way to run Java.
- There are many features and tools that Tomcat 9 has to offer that can make the deployment of web pages more manageable, including the use of Java Server Pages.

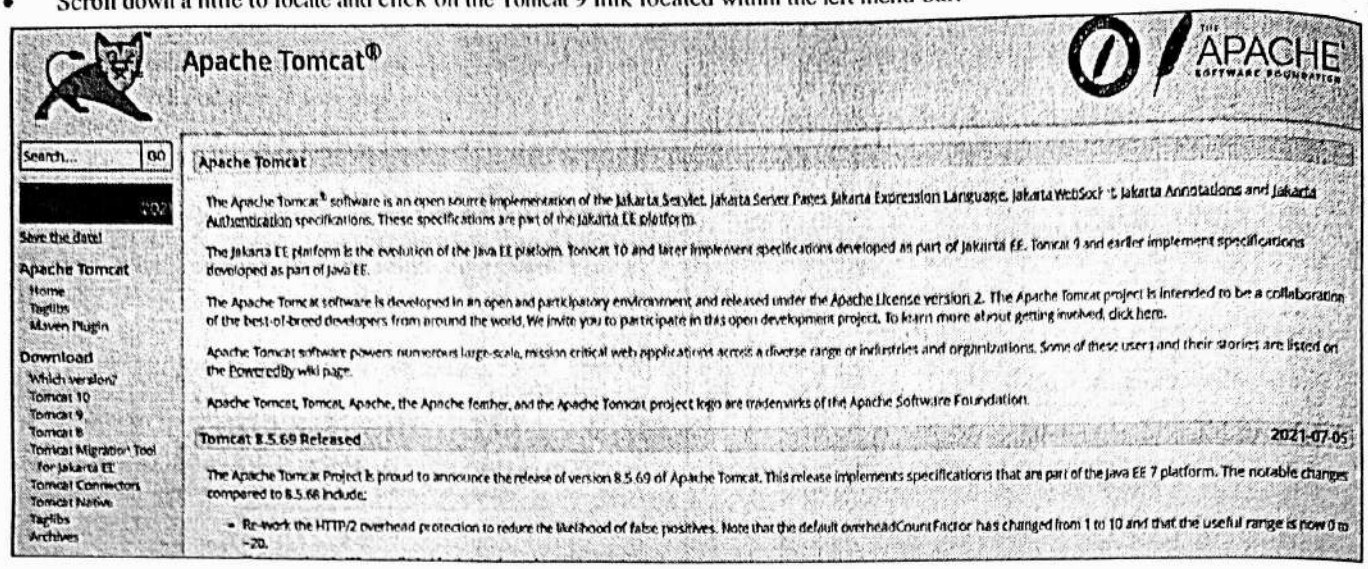
Prerequisites

- JDK or JRE will need to be installed on the Windows Server before you can configure Tomcat 9 on the server. OpenJDK and Amazon Corretto are two examples of open-source Java Development Kit providers.

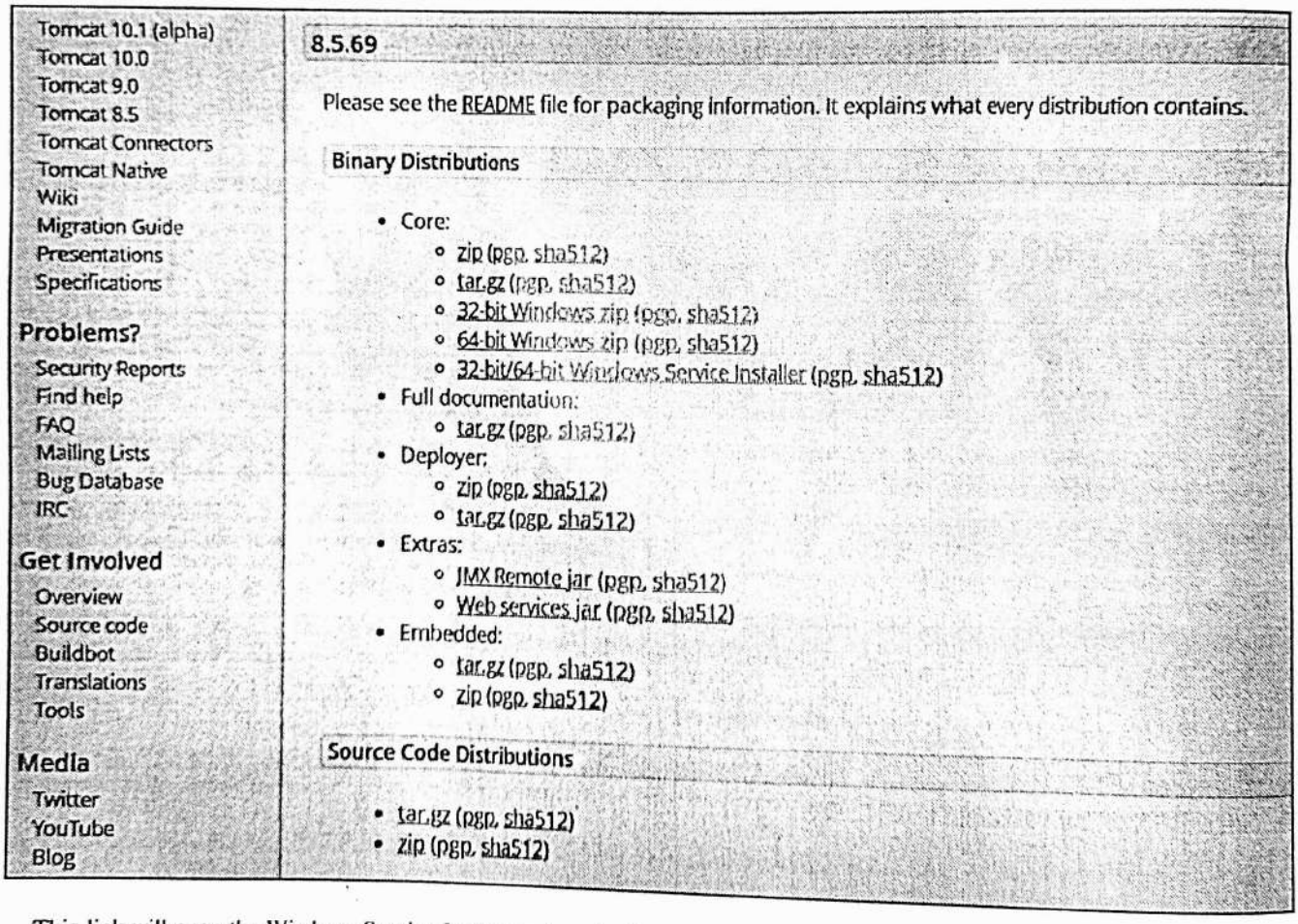
Installing Tomcat 9

- Open your browser and head over to https://tomcat.apache.org.

- Scroll down a little to locate and click on the Tomcat 9 link located within the left menu bar.



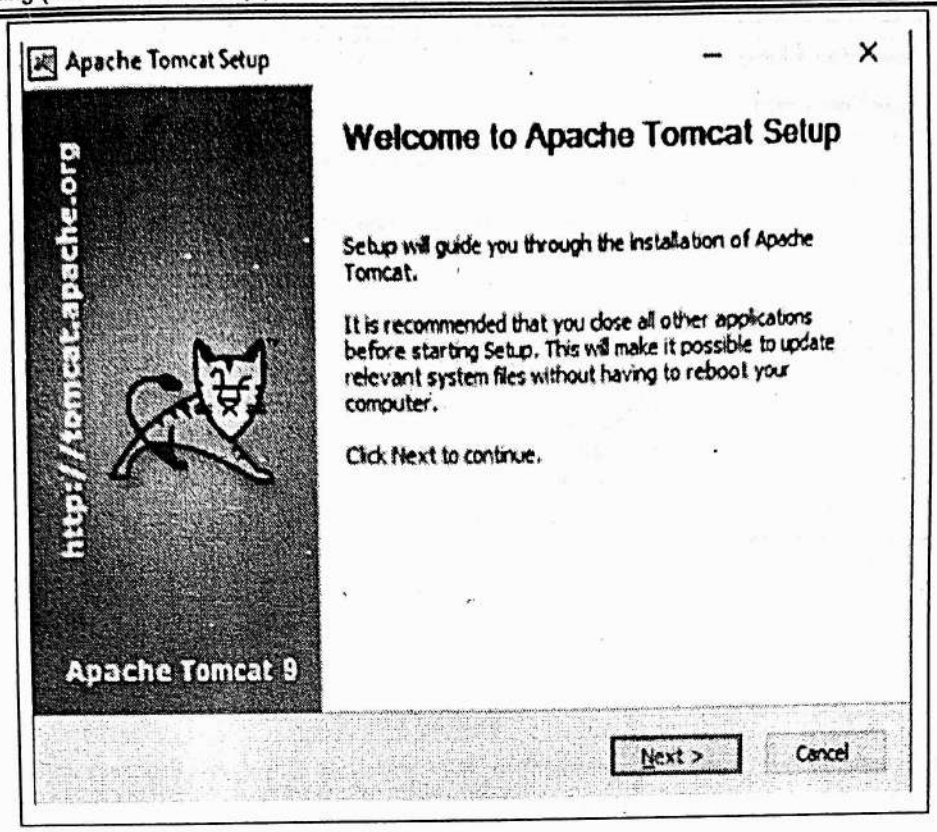
Next, locate the 32-bit/64-bit Windows Service Installer link and click on it.



This link will open the Windows Service Installer automatically. Once the installer window pops up, you will initiate the install by clicking on the Start Download button. Once the download completes, double click the .exe installer to begin the installation.

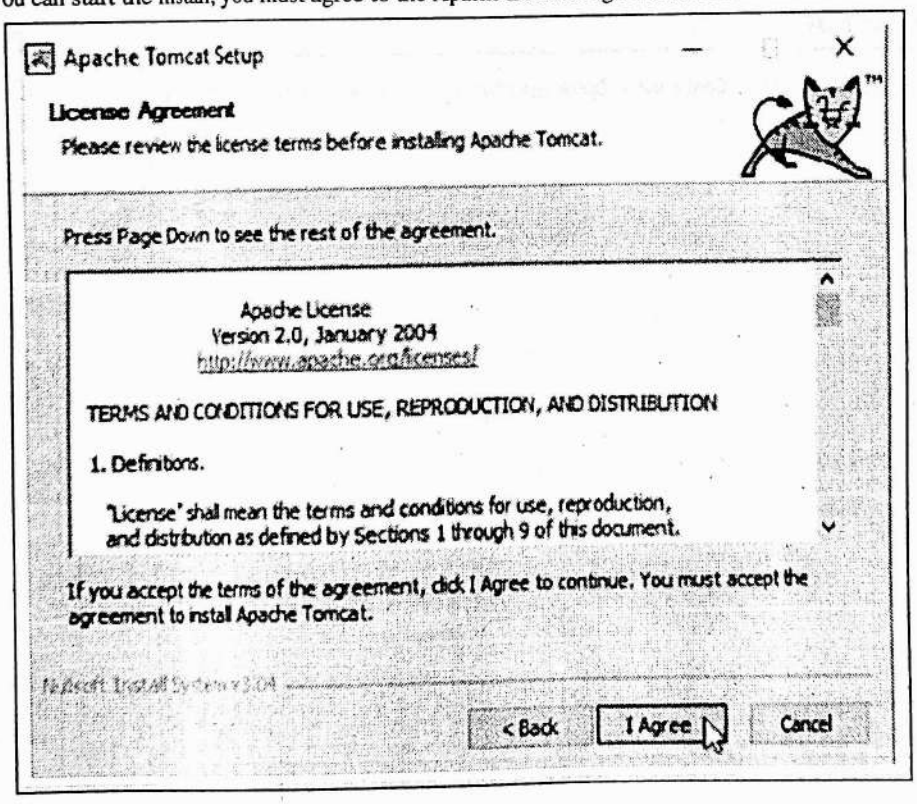
☞ Installation

▶ **Step 1 :** The first page of the install explains what the installer will do and what to expect. Go ahead and click Next on the first page.

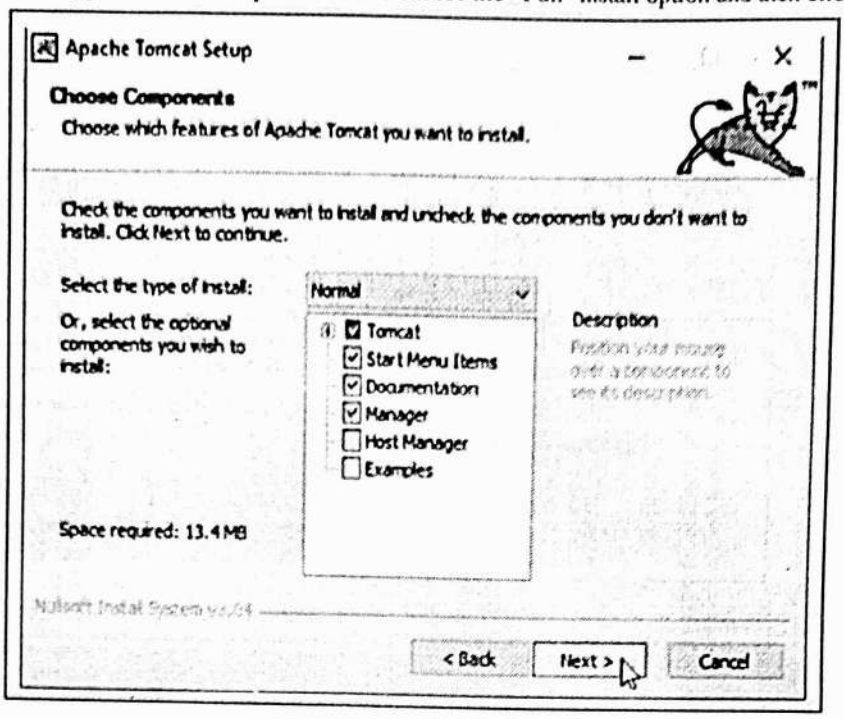


Module 3

▶ **Step 2 :** Before you can start the install, you must agree to the Apache License Agreement for the Tomcat 9 service.



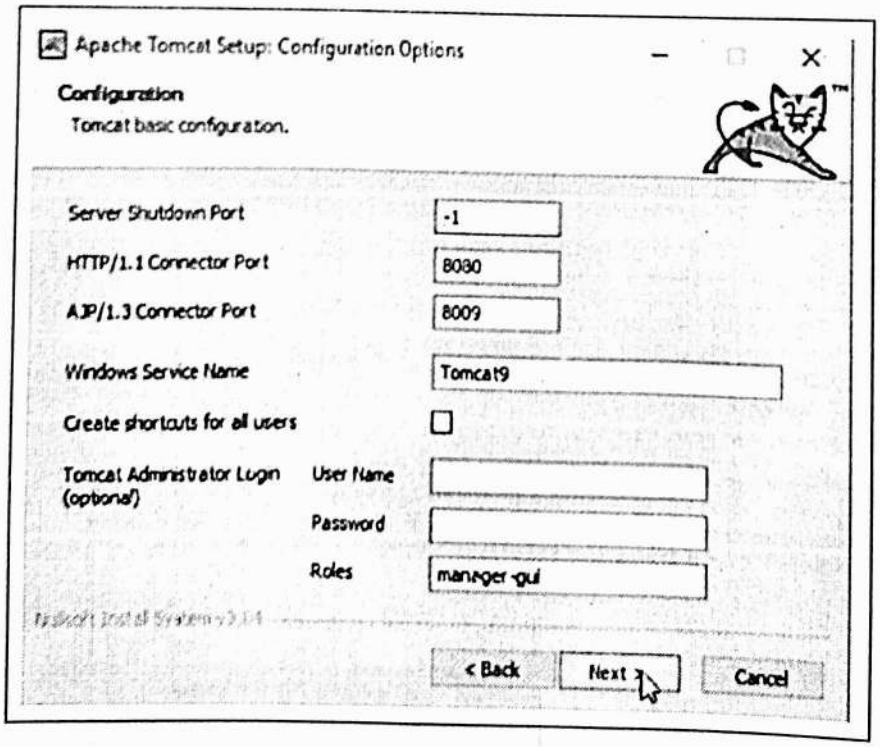
- **Step 3 :** Click on "Select the type of install" dropdown list and choose the "Full" install option and then click Next.



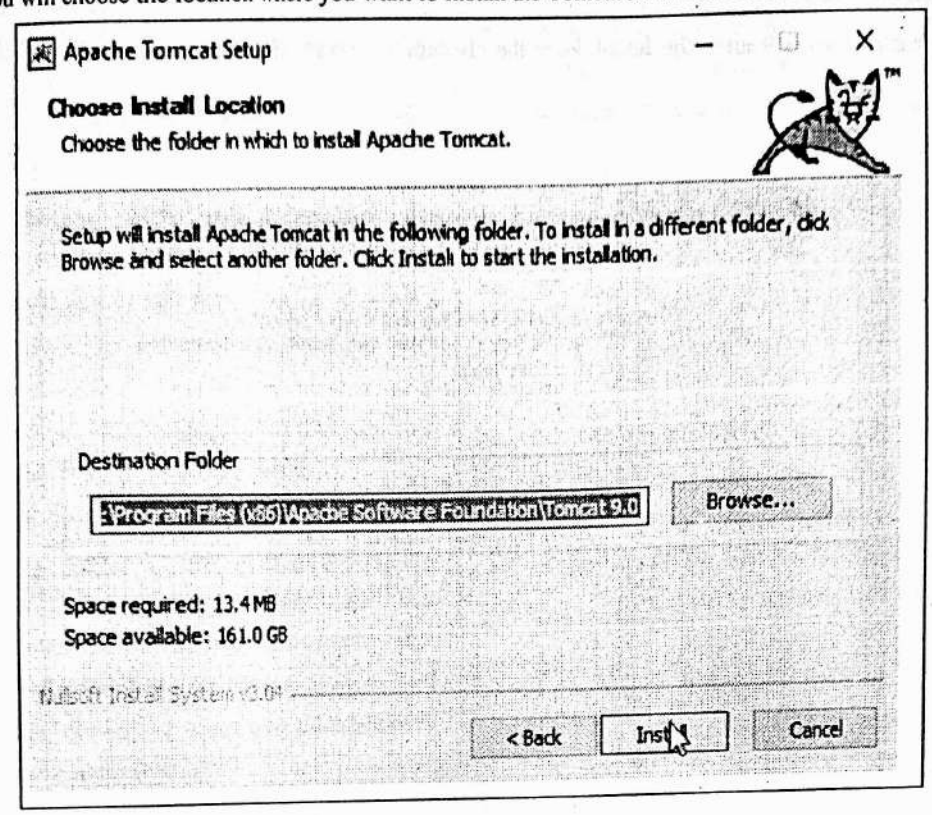
- **Step 4. :** The next screen is the configuration screen. This screen will allow you to set up any default ports that you want the service to connect through, and will also allow you to set an Administrator username and password.

  Please note that you can configure these settings later if you choose to.

  Choose your selections and click Next.

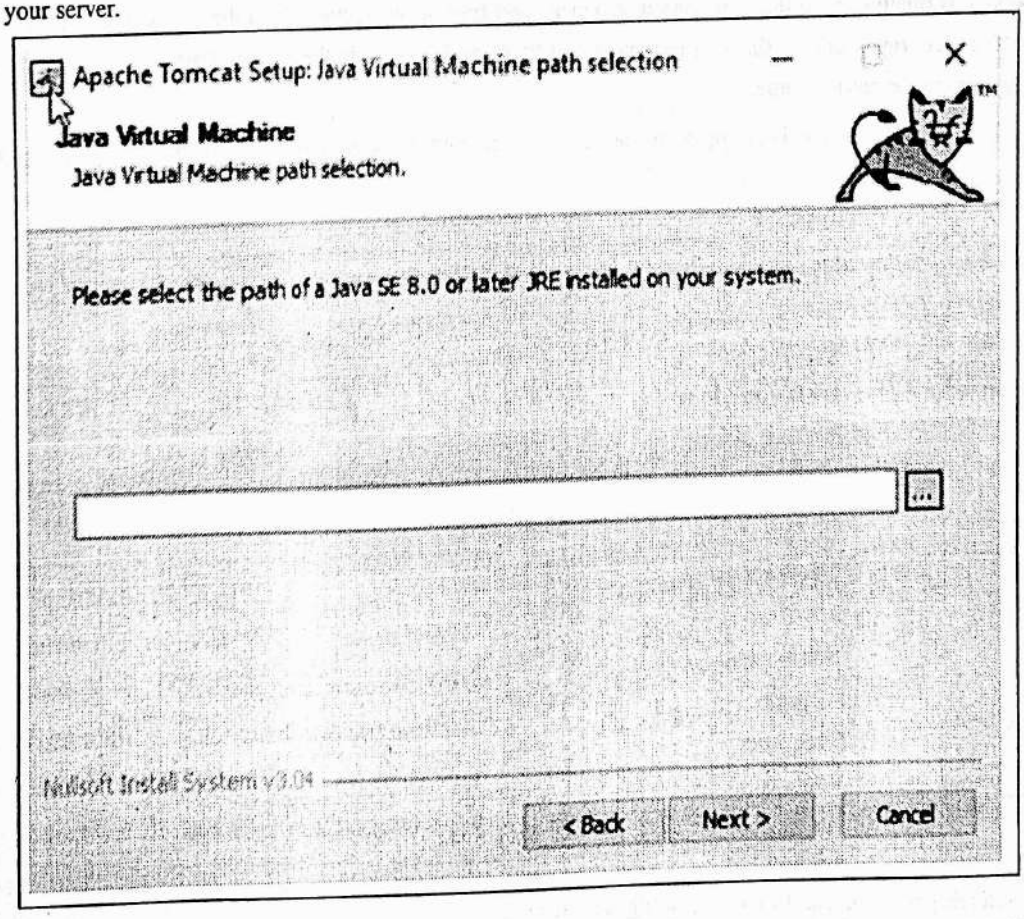


- **Step 5 :** Next, you will choose the location where you want to install the Tomcat 9 service. Click Next.



Module 3

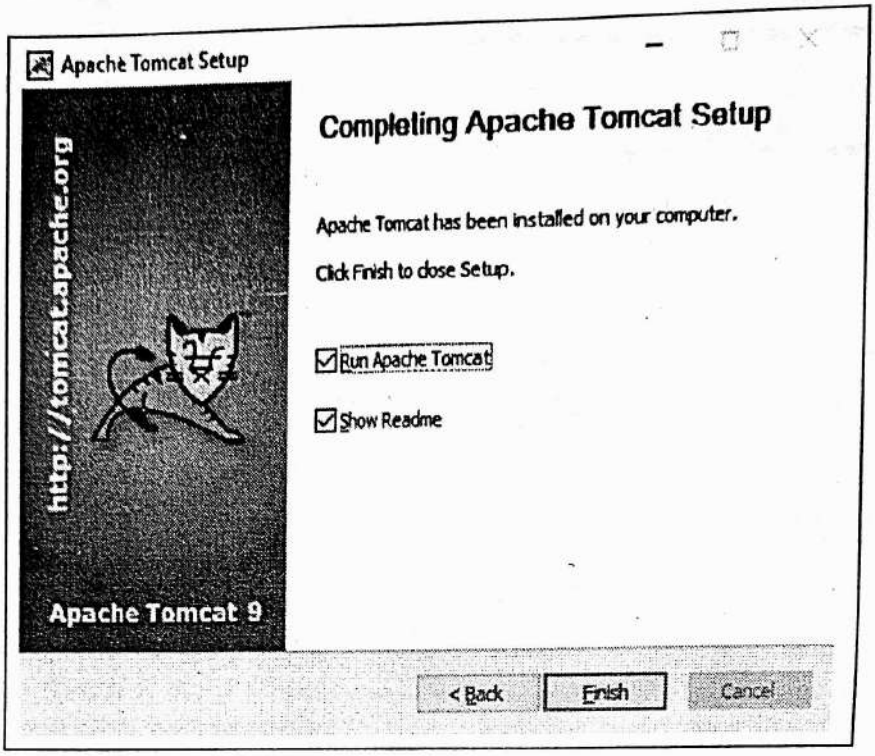
- **Step 6 :** To correctly install Tomcat 9 on your server, the Wizard will want you to choose the location of any other Java related software on your server.



- **Step 7 :** Once the install is complete, click Next.

Lastly, uncheck the Readme checkbox.

If you prefer to start Apache Tomcat 9 after the install, leave the checkbox marked. If not, then you will want to uncheck that checkbox.



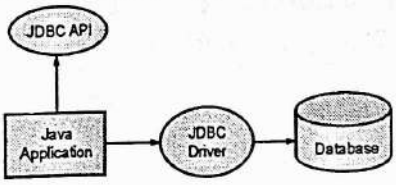
- To make sure the service is running, go to the Windows startup menu and type services.cmd. From the list of available services, find the Apache Tomcat 9 service, right-click on the service name, select Start, and make sure the service starts successfully. You should see a "running" status next to the service name.
- To test the Tomcat 9 install and verify it is running on your server, open and point your browser to localhost:8080 (or whatever custom port you put into the configuration).

The page that loads will tell you if Tomcat 9 was successfully installed.

3.7 DATABASE CONNECTIVITY: JDBC PERSPECTIVES

3.7.1 Introduction to JDBC

- **Java Database Connectivity (JDBC) is an Application Programming Interface (API)** used to connect Java application with Database.
- JDBC is an API developed by Sun Microsystems to enable a standards-based method to access data using the Java language.
- JDBC is used to interact with various type of Database such as Oracle, MS Access, My SQL and SQL Server.
- It is a part of JavaSE (Java Standard Edition).
- JDBC can also be defined as the platform-independent interface between a relational database and Java programming. It allows java program to execute SQL statement and retrieve result from database.
- JDBC API uses JDBC drivers to connect with the database.
- There are four types of JDBC drivers:
  - JDBC-ODBC Bridge Driver
  - Native Driver
  - Network Protocol Driver
  - Thin Driver
- We can use JDBC API to access tabular data stored in any relational database. With the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



- The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The java.sql package contains classes and interfaces for JDBC API.
- A list of popular interfaces of JDBC API are given below:
  1. Driver interface 2. Connection interface
  3. Statement interface 4. PreparedStatement interface
  5. CallableStatement interface
  6. ResultSet interface 7. ResultSetMetaData interface
  8. DatabaseMetaData interface
  9. RowSet interface

Why should we use JDBC?

- Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).
- We can use JDBC API to handle database using Java program and can perform the following activities:
  - Connect to the database
  - Execute queries and update statements to the database
  - Retrieve the result received from the database.

3.7.2 JDBC Driver

- JDBC Driver is required to establish connection between application and database. It also helps to process SQL requests and generating result.
- The following are the different types of driver available in JDBC which are used by the application based on the scenario and type of application.

1. Type-1 Driver or JDBC-ODBC bridge
2. Type-2 Driver or Native API Partly Java Driver
3. Type-3 Driver or Network Protocol Driver
4. Type-4 Driver or Thin Driver

1) JDBC-ODBC bridge driver

- The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.
- In Java 8, the JDBC-ODBC Bridge has been removed.

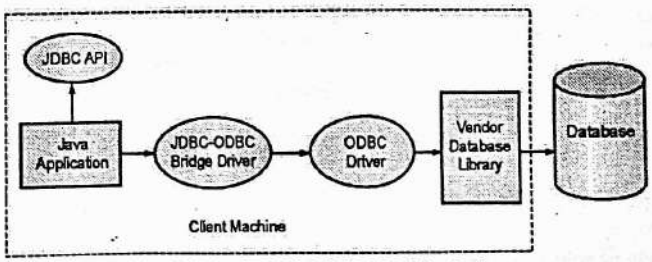


Fig. 3.7.1 : JDBC-ODBC Bridge Driver

Advantages

1. Easy to use
2. Allow easy connectivity to all database supported by the ODBC Driver.

Module 3

Disadvantages

1. Performance degraded because JDBC method call is converted into the ODBC function calls.
2. The ODBC driver needs to be installed on the client machine.

2) Native-API driver

- The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

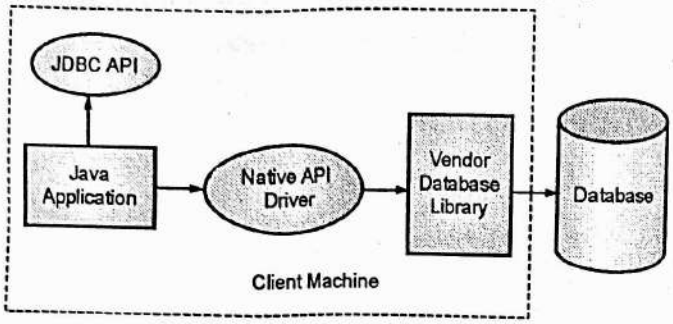


Fig.3.7.2: Native-API Driver

Advantage

1. Performance upgraded than JDBC-ODBC bridge driver.

Disadvantages

1. The Native driver needs to be installed on the each client machine.
2. The Vendor client library needs to be installed on client machine.

3) Network Protocol Driver

- The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

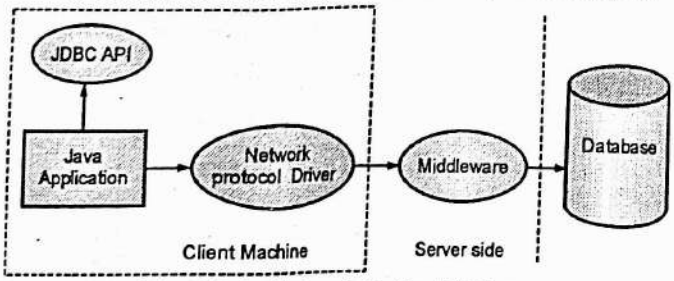


Fig. 3.7.3 : Network Protocol Driver

Advantage

1. No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

Disadvantages

1. Network support is required on client machine.
2. Requires database-specific coding to be done in the middle tier.
3. Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

4) Thin driver

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

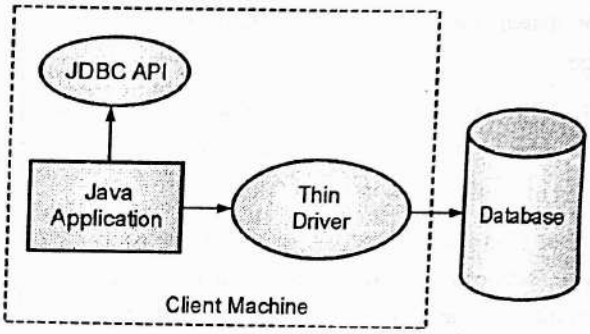


Fig.3.7.4 : Thin Driver

Advantages

1. Better performance than all other drivers.
2. No software is required at client side or server side.

Disadvantage

1. Drivers depend on the Database.

3.7.3 Steps to connect a Java Application to Database

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

Step 1 : Register the Driver
Step 2 : Create a Connection
Step 3 : Create SQL Statement
Step 4 : Execute SQL Statement
Step 5 : Closing the connection

Step 1 : Register the Driver

- It is an essential part to create JDBC connection. JDBC API provides a method **Class.forName()** which is used to load the driver class explicitly.
- For example, if we want to load a jdbc-odbc driver then we call it like following-
- **Example to register with JDBC-ODBC Driver**

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Step 2 : Create a Connection

- After registering and loading the driver in step1, now we will create a connection using getConnection() method of DriverManager class. This method has several overloaded methods that can be used based on the requirement. Basically it require the database name, username and password to establish connection. Syntax of this method is given below.

Syntax

```
getConnection(String url)
getConnection(String url, String username, String password)
getConnection(String url, Properties info)
```

Sample example to establish connection with Oracle Driver

```
Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1
521:XE","username","password");
```

Step 3: Create SQL Statement

- In this step we will create statement object using **createStatement()** method. It is used to execute the sql queries and defined in Connection class. Syntax of the method is given below.

Syntax

```
public Statement createStatement() throws SQLException
Example to create a SQL statement
Statement s=con.createStatement();
```

Step 4 : Execute SQL Statement

- After creating statement, now execute using executeQuery() method of Statement interface. This method is used to execute SQL statements. Syntax of the method is given below.

Syntax

```
public ResultSet executeQuery(String query) throws
SQLException
```

Example to execute a SQL statement

- In this example, we are executing a sql query to select all the records from the user table and stored into resultset that further is used to display the records.

```
ResultSet rs=s.executeQuery("select * from user");
 while(rs.next())
 {
 System.out.println(rs.getString(1)+" "+rs.getString(2));
 }
```

Step 5 : Closing the connection

- This is final step which includes closing all the connection that we opened in our previous steps.
- After executing SQL statement you need to close the connection and release the session. The close() method of Connection interface is used to close the connection.

Syntax

```
public void close() throws SQLException
```

Example of closing a connection

```
con.close();
```

Example : All Steps into one place

```
import java.sql.*;
class Test {
 public static void main(String[] args) {
  try {
  //Loading driver
  Class.forName("oracle.jdbc.driver.OracleDriver");

  //creating connection
  Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1
521:XE", "username", "password");
  Statement s = con.createStatement(); //creating statement

  ResultSet rs = s.executeQuery("select * from Student");
//executing statement

  while (rs.next()) {
   System.out.println(rs.getInt(1) + " " + rs.getString(2));
  }

  con.close(); //closing connection
  } catch (Exception e) {
  e.printStacktrace();
  }
 }
}
```

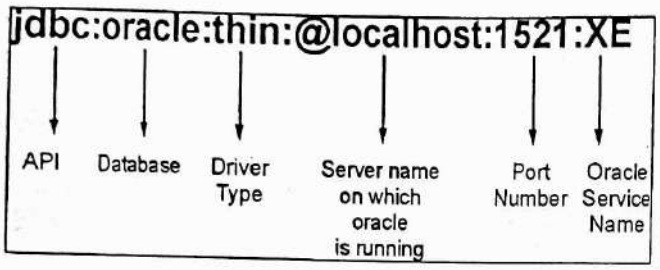
Module 3

3.7.4 JDBC program example

➢ **Program 3.7.1 :** Java Database Connectivity with Oracle Database using Thin Driver

To connect a Java application with Oracle database using Thin Driver. You need to follow the following steps

1. **Load Driver Class :** The Driver Class for oracle database is oracle.jdbc.driver.OracleDriver and Class.forName("oracle.jdbc.driver.OracleDriver") method is used to load the driver class for Oracle database.
2. **Create Connection :** For creating a connection you will need a Connection URL. The Connection URL for Oracle is



You will also require Username and Password of your Oracle Database Server for creating connection.

3. **Loading jar file :** To connect your java application with Oracle, you will also need to load ojdbc14.jar file. This file can be loaded into 2 ways.
4. Copy the jar file into C:\Program Files\Java\jre7\lib\ext folder. or
5. Set it into class path.

Let's see programs for performing simple operations like create, insert and select on database tables.

Create a table in Oracle Database

Use the below SQL command to create a table Student into oracle database. This table has two columns: sid and sname.

create table Student(sid number(10),sname varchar2(20));

Insert some record into the table

After creating the table now insert two records into table using the below sql command. Here we are inserting Alice and Bob named two rows.

```
insert into Student values(101,'Alice');
insert into Student values(102,'Bob');
```

Example : Accessing record

In the above example, we inserted two records using SQL command. Now we are fetching those records using JDBC Java application.

```
import java.sql.*;
class Test
{
    public static void main(String []args)
    {
        try{
            //Loading driver
Class.forName("oracle.jdbc.driver.OracleDriver");

            //creating connection
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:XE","username","passwor
d");

Statement s=con.createStatement();    //creating statement

ResultSet rs=s.executeQuery("select * from Student");

//executing statement
while(rs.next()){
System.out.println(rs.getInt(1)+" "+rs.getString(2));
            }

    con.close();   //closing connection
}catch(Exception e){
    e.printStackTrace();
        }
    }
}
```

Output

```
101 Alice 102 Bob
```

Example : Inserting record

We can insert data into table using Java application using the JDBC. Here we are using Prepared statement and parameterized query.

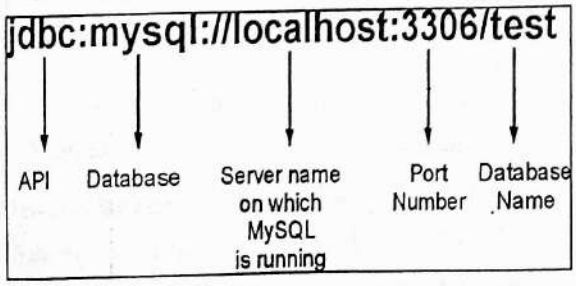
```
import java.sql.*;
class Test
{
```

```
    public static void main(String []args)
    {
        try{
            //Loading driver...
Class.forName("oracle.jdbc.driver.OracleDriver");

            //creating connection...
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:XE","username","passwor
d");

PreparedStatement pst=con.prepareStatement("insert into
Student values(?,?)");

            pst.setInt(1,104);
            pst.setString(2,"Alex");
            pst.executeUpdate();

            con.close();   //closing connection
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

➢ **Program 3.7.2 :** Java Database Connectivity with MySQL Database using Thin Driver

To connect a Java application with MySQL database using Thin Driver. You need to follow the following steps

1. Load Driver Class: The Driver Class for MySQL database is com.mysql.jdbc.Driver and Class.forName ("com.mysql.jdbc.Driver") method is used to load the driver class for MySQL database.
2. Create Connection: For creating a connection you will need a Connection URL. The Connection URL for MySQL is



- You will also require Username and Password of your MySQL Database Server for creating connection.

1. Loading jar file: To connect your java application with MySQL, you will also need to load mysql-connector.jar file. This file can be loaded into 2 ways.
2. Copy the jar file into C:\Program Files\Java\jre7\lib\ext folder. Or
3. Set it into classpath.

Let's see programs for performing simple operations like create, insert and select on database tables.

☞ Create a table in MySQL Database

Use the below SQL command to create a table Student into mysql database. This table has two columns: sid and name.

create table Student(sid int(10),name varchar(20));

Insert some record into the table

After creating the table now insert two records into table using the below sql command. Here we are inserting Nirmit and Abhi named two rows.

insert into Student values(102,'Nirmit');

insert into Student values(103,'Abhi');

Example : Accessing record

In the above example, we inserted two records using SQL command. Now we are fetching those records using JDBC Java application.

```
import java.sql.*;
class Test
{
    public static void main(String []args)
    {
        try{
            //Loading driver
Class forName("com.mysql.jdbc.Driver");

            //creating connection
Connection con = DriverManager.getConnection
("jdbc:mysql:// /localhost:3306/test","username","password");

Statement s = con.createStatement();   //creating statement

ResultSet rs = s.executeQuery("select * from Student");
//executing statement

  while(rs.next()){
```

Module 3

```
        System.out.println(rs.getInt(1)+" "+rs.getString(2));
            }

            con.close();   //closing connection
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Output

```
101 Nirmit 102 Abhi
```

Example : Inserting Record

We can insert data into table using Java application. Here we are using Prepared statement and parameterized query to execute SQL query. The executeUpdate() method is used to execute the insert query.

```
import java.sql.*;
class Test
{
    public static void main(String []args)
    {
        try{
            //Loading driver
Class.forName("com.mysql.jdbc.Driver");

            //creating connection
Connection con = DriverManager.getConnection
("jdbc:mysql://localhost:3306/test","username","password");

PreparedStatement pst=con.prepareStatement("insert into
Student values(?,?)");

            pst.setInt(1,104);
            pst.setString(2,"Alex");
            pst.executeUpdate();

            con.close();   //closing connection
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

3.8 JSP

3.8.1 Introduction to JSP

GQ. What is JSP?

- Java Server Pages (JSP) is a server side technology for developing dynamic web pages.
- JSP can be used for separation of the view layer with the business logic in the web application.
- JSP is an extension of servlets and every JSP page first gets converted into servlet by JSP container before processing the client's request.
- A JSP page consists of HTML tags and JSP tags.
- The JSP pages are easier to maintain than Servlet because we can separate designing and development.
- JSP provides some additional features such as Expression Language, Custom Tags, etc.
- The JSP tags which allow java code to be included into it are <%----java code----%>.

Advantages of JSP over Servlet

(1) **Extension to Servlet :** JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, which makes JSP development easy.

(2) **Easy to maintain :** JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

(3) **Fast Development :** If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

(4) **Less code than Servlet :** In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code.

(5) **Exception Handling :** JSP takes care of exception handling. Servlet does not take care of exception handling. It is the responsibility of programmers.

(6) **Readability :** JSP increases readability of code by using tags. In Servlet, readability of code is less.

(7) **Less Complexity :** JSP is easy to learn and apply. Servlet is comparatively complex to learn and apply.

Servlet vs. JSP

- Like JSP, Servlets are also used for generating dynamic webpages.
- The major difference between them is that servlet adds HTML code inside java while JSP adds java code inside HTML.

Servlet

1. Servlet is a Java program which supports HTML tags too.
2. Generally used for developing business layer (the complex computational code) of an enterprise application.
3. Servlets are created and maintained by Java developers.

JSP

1. JSP program is a HTML code which supports java statements too. JSP embed java in html using JSP tags.
2. Used for developing presentation layer of an enterprise application
3. Frequently used for designing websites and used by web developers.

3.8.2 JSP Life Cycle

GQ. Explain life cycle of JSP in detail.

- JSP Life Cycle is defined as the translation of the JSP Page into the servlet. JSP Page always needs to be converted into servlet page first to process the service requests.
- Every JSP page ends with .jsp extension. Whenever we were given a request to the server for a .jsp file then first .jsp will be converted into a .java file.
- Second, .java file is converted into .class file. Third, .class is loaded by container and managed by the container. Basically .jsp to .java and .java to .class conversion is performed by JSP engine.
- All .class file loading and execution is performed by the container (servlet container). The JSP life cycle always starts with the creation of the JSP page and ends with the disintegration of that.

Stages of Java Server Page

JSP passes through the following phases :

1) Translation of JSP page 2) Compilation of JSP page
3) Class loading 4) Instantiation
5) Initialization 6) Request Processing
7) Destruction

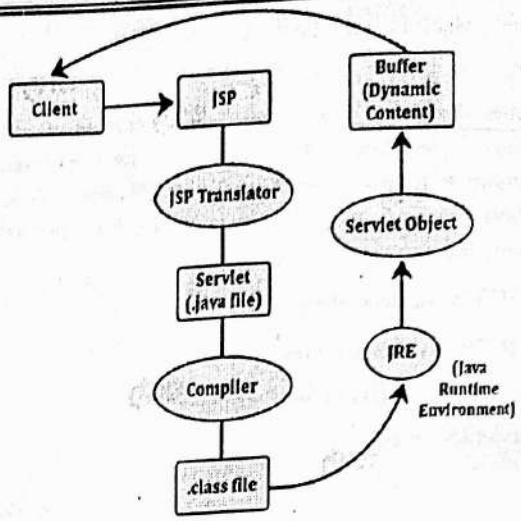


Fig. 3.8.1 : JSP Life Cycle

(1) **JSP Translation :** When the client sends the request to the web server, it goes to the web container. Now evidently, if the JSP page is newer and the servlet class is older, it means that the page got modified. In that case, the translation is done otherwise, this step is skipped so as to improve the performance of the system as it takes time.

(2) **JSP Compilation :** Prior to compilation, JSP engine probes if the page has ever been compiled. If the page has some modifications or if it has never been compiled, then JSP engine compiles the page. Now the compilation follows three steps:

(a) Parsing JSP

(b) Conversion of JSP to Servlet

(c) Compiling the servlet

(3) **Class Loading :** At this stage, the .class file gets loaded by the loader i.e., corresponding servlet class file is loaded.

(4) **Instantiation :** As we compiled the servlet, the object of the generated servlet is created. This process takes place after the class file is loaded in the memory. As soon as the instantiation takes place the objects: ServletConfig and ServletContext get accessible to the JSP class.

ServletConfig is a pre-defined interface used to develop flexible servlets. **ServletContext** is generated by the web container when the project is deployed. ServletConfig object exists as one per servlet program, whereas ServletContext object exists as one per web application.

(5) **Initialization :** After instantiation, container invokes jspInit() method better known as initialization of JSP. It basically initializes the instance we created. It is done to initialize resources needed to process the request i.e. resources like databases, allow JSP pages to read data, create lookup tables, and manage network connections.

Module 3

```
public void jspInit() { //Initialization code
}
```

(6) **Request Processing :** After the initialization of servlet instances, a new thread gets created. All the interactions necessary to process requests occur at this stage. Thus, container invokes _jspService() method. This method contains two parameters:

- HTTPServletRequest req
- HTTPServletResponse res

```
public void _jspService(HTTPServletRequest req,
HTTPServletResponse res) {
 // Enter jsp Service code here
}
```

- This service method is responsible for generating a response to the request. Not only this, but it also generates response to all seven HTTP methods, namely: GET, POST, HEAd, PUT, DELETE, CONNECT and OPTIONS.

(7) **Destruction :** The destruction basically represents cleanup. For the cleanup jspDestroy() method gets invoked. This cleanup means that the JSP page after processing gets removed from the container. This method destroys the instance of the servlet class code, and all the connections to database and network are released. Opened files are also closed. jspDestroy() is overridden whenever the cleanup needs to be done.

```
public void jspDestroy() {
 //Cleanup code
}
```

3.8.3 Creating a simple JSP Page

Simple.jsp

```
<%-- JSP comment --%>
<HTML>
<HEAD>
<TITLE>MESSAGE</TITLE>
</HEAD>
<BODY>
<%out.print("Hello, Sample JSP code");%>
</BODY>
</HTML>
```

Explanation of above code

1) The line <%–JSP Comment–%> represents the JSP element called JSP Comment, While adding comments to a JSP page you can use this tag.

2) Head, Title and Body tags are HTML tags – They are HTML tags, frequently used for static web pages. Whatever content they have is delivered to client(Web browser) as such.

3) <%out.print(" Hello, Sample JSP code ");%> is a JSP element, which is known as Scriptlet. Scriptlets can contain Java codes. syntax of scriptlet is: **<%Executable java code%>**.

As the code in Scriptlets is java statement, they must end with a semicolon(;). out.print(" Hello, Sample JSP code ") is a java statement, which prints" Hello, Sample JSP code".

3.8.4 Elements of JSP Page

There are three types of elements in JSP.

1. Scripting Elements
2. Directives
3. Actions and Templates

1. Scripting Elements

Scripting elements are used to include scripting code (java code) within the JSP. These elements help to perform various operations like declaring variables and methods, include implementation code and evaluate an expression.

Types of scripting elements

GQ. Explain different types of scripting elements in JSP.

A. Declaration

- The declaration section is used to declare class-wide variables and methods. This is block of Java code in JSP. When the JSP page is initialized all the declarations in the file get initialized. These members have class scope means they are accessible within the JSP anywhere to other declarations, expressions or code.
- The declaration section is enclosed in **<%! and %>**.

```
<%! Declaration statements of variables and methods %>
```

Example

```
<html>
<head>
<title>
Scriptlet
</title>
<%!
private int n = 1;
%>
```

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

B. Scriptlets

- The scriptlet is a block of java statements. This block is executed when the client request is processed by the JSP. The scriptlet generates output in output stream to send to client. In a JSP there may be multiple scriptlets as per the requirement.
- A scriptlet is instance of service() method of Servlet. When the JSP is converted into Servlet by the JSP engine, the code of scriptlet is put into service() method. The scriptlet is processed for every client request.
- The scriptlet section is enclosed in <% and %>.

```
<% Java statements; %>
```

Example

```
<html>
<head>
<title>
Scriptlet
</title>
<%!
private int n = 1;
%>
</head>
<body>
<font size=5>
<%
while(n<=10)
{
out.println("<br>Welcome to JSP");
n++;
}
%>
</body>
%>
</html>
```

C. Expressions

- The expression outputs a value in the response stream which is to be sent to client. Expression is considered as shorthand notation of scriptlet. After evaluation of the expression, the result is converted into string format and displayed to the client.
- The expression is enclosed in <%= and %>

Example

```
<html>
<head>
<title>
Scriptlet
</title>
<%!
private int n = 1;
%>
</head>
<body>
<font size=5>
<%
while(n<=10)
{
    out.println("<br>Welcome to JSP");
    n++;
}
%>
<br> Now the value of n is <%=n%>
</font>
</body>
</html>
```

D. Comments

Comments are used to give informative text in the JSP. The comments are ignored by the JSP container.

☞ Syntax

- Following is the syntax of the JSP comments -

```
<%-- This is JSP comment --%>
```

Example

Following example shows the JSP Comments -

```
<html>
  <head><title>Example of comment</title></head>

  <body>
    <h2>Welcome to the world of Web</h2>
    <%-- This is comment which is invisible in the page source --%>
  </body>
</html>
```

Module 3

2. Directives

GQ. Explain various JSP directives with example.

- The directive in JSP serves as messages and directions to the JSP container. It instructs the container the way to handle some important aspects of the JSP processing.
- The entire structure of the Servlet class is affected by the directives.

Syntax

The standard form of JSP directive is as

```
<%@ directive attribute = "value" %>
```

- The directive usually has multiple attributes which can be specified as key-value pair which should be separated by the commas.

Types of directive tag

Sr. No.	Directive	Description
1.	<%@ page ... %>	This directive is used to define page dependent attributes like language, buffer, content type etc.
2.	<%@ include ... %>	Used to include external files during the translation phase.
3.	<%@ taglib ... %>	Used to define a tag library to be used in JSP

1. The Page Directive

The Page Directive gives instructions to the JSP container. These instructions affect the entire current JSP page. As per convention, the Page Directive should be declared at the top of the JSP, but can be declared anywhere in the JSP page.

Different attributes of Page Directive

GQ. Explain different attributes of page directive in JSP.

Sr. No.	Attribute	Purpose
1.	Buffer	Buffer is temporary memory. This attribute is used to specify a buffering model for the output stream to the client.
2.	autoFlush	If set to "true" then the output buffer to the client is flushed automatically.
3.	contentType	Defines the character encoding for the JSP and MIME type for the response of JSP page.
4.	errorPage	Defines the URL of another JSP page. When unchecked runtime exception occurs then the control is transferred to this error page.
5.	isErrorPage	Specifies if this JSP page is a URL mentioned by other JSP page's errorPage attribute.
6.	Extends	Specifies a parent class through which the generated Servlet must be extended.
7.	Import	Specifies a comma separated list of packages or classes for use in the JSP.
8.	info	Defines an information string which can be accessed with the servlet's getServletInfo() method.
9.	isThreadSafe	Defines the model of thread for the generated Servlet. If set to true, multiple request can be handled simultaneously
10.	Language	Defines the scripting language used in the JSP page.
11.	session	Specifies whether or not the JSP page participates in HTTP sessions

2. **The include Directive**

- The include directive is used to include external files during the translation phase. The content of external files can be merged in the current JSP by the container. The include directive can be declared anywhere in the JSP page.

☞ **Syntax**

```
<%@ include file = "relative url" >
```

- In the include directive we have to specify the relative URL of the external file which we want to include in our JSP. If the filename is given without any specific path, then the JSP compiler search for the file in the same directory in which the current JSP is stored.

3. **The taglib Directive**

- This directive allows the JSP page to include and use custom user defined tag library. This tag library is a bunch of user-defined JSP tags which implement custom behavior.
- The taglib directive indicates that the JSP page is going to use some custom tags.

☞ **Syntax**

```
<%@ taglib uri="uri" prefix = "prefixOfTag" >
```

- The URI (Uniform Resource Identifier) attribute identifies the location which the container understands and the prefix attribute informs prefix string which is used to define the custom tags.

☞ **Syntax**

```
<%@ page attribute = "value" %>
```

3. **Actions and Templates**

GQ. Enlist and explain different types of actions in JSP.

JSP actions are used to handle the behavior of the Servlet engine. These actions can perform various tasks like insertion of file dynamically, forward the control to next page, and reuse JavaBeans components or creation of HTML content for the Java plugin.

☞ **Types of Actions**

1. **<jsp:include> Action**

This is same as of include page directive i.e. it is used to insert a JSP file in another file.

<jsp:include> vs include directive

In <jsp:include> the external file is inserted at the time of request processing while in include directive it is inserted at translation phase.

☞ **Syntax of <jsp:include>**

```
<jsp:include page="page URL" />
```

Example

```
<html>
<head>
<title>Example of JSP include Action Tag</title>
</head>
<body>
<h3>Include JSP</h3>
<jsp:include page="test.jsp" flush="false" />
</body>
</html>
```

- page : Page value is test.jsp. It is the page which we want to include in the current file. Only the file name indicates that the test.jsp is in the same directory.
- flush : Its value is false, which indicates that the resource buffer has not been flushed out before inserting in the current page.

2. **<jsp:forward> Action**

<jsp:forward> is used for redirecting the request and transfer the control to another file.

☞ **Syntax of <jsp:forward>**

```
<jsp:forward page="URL_page" />
```

Example

first.jsp

```
<html>
<head>
<title> Forward Action Tag</title>
</head>
<body>
<h3>Forward</h3>
<jsp:forward page="next.jsp" />
</body>
</html>
```

Module 3

3. **<jsp:param> Action**

This action is used to pass parameters to other JSP action tags like JSP include and JSP forward tag. The request object is used to accept those parameters.

☞ Syntax of <jsp:param>

```
<jsp: param name="param_name "
value="value_parameter" />
```

Example

first.jsp

```
<html>
<head>
<title> Param Action Tag</title>
</head>
<body>
<h3>Param Action</h3>
<jsp:forward page="next.jsp">
  <jsp:param name ="date" value="14-08-2017" />
  <jsp:param name ="time" value="10:15AM" />
  <jsp:param name ="data" value="Wave" />
</jsp:forward>
</body>
</html>
```

In this example the page first.jsp is passing three parameters; data, time & data to next.jsp and next.jsp can access these parameters as follows:

```
sDate:<%= request.getParameter("date") %>
sTime:<%= request.getParameter("time") %>
sData:<%= request.getParameter("data") %>
```

4. **<jsp:useBean> Action**

This action is used to use Beans in JSP page. Beans are invoked by this tag.

☞ Syntax of <jsp:useBean>

```
<jsp: useBean id=" bean_id"
class="package_name.class_name" />
```

☞ Example of <jsp:useBean>, <jsp:setProperty> and <jsp:getProperty>

When Bean class is instantiated as shown in above statement, we have to use jsp:setProperty and jsp:getProperty actions to use the bean's parameters.

☞ Test.jsp

```
<html>
  <head>
    <title>Using Beans</title>
  </head>
  <body>
    <h1>Bean Test</h1>
    <jsp:useBean id="sn" class="javabn.StuBean"/>
    <jsp:setProperty name="sn" property="*"/>
    <h1>
      name:<jsp:getProperty name="sn"
property="sname"/><br>
      empno:<jsp:getProperty name="sn"
property="srollno"/><br>
    </h1>
  </body>
</html>
```

☞ StudentBean.java

```
package javabn;
public class StuBean {
  public StuBean() {
  }
  private String sname;
  private int srollno;
  public void setName(String name)
  {
    this. sname =name;
  }
  public String getName()
  {
    return sname;
  }
  public void setRollno(int rollno)
  {
    this.srollno =rollno;
  }
  public int getRollno()
  {
    return srollno;
  }
}
```

5. **<jsp:setProperty> Action**

This action tag is basically used to set the property of a Bean. We have to mention unique name for bean while using this action tag.

☞ Syntax of <jsp:setProperty>

```
<jsp: useBean id=" bean_id"
class="package_name.class_name" />
....
....
<jsp:setProperty name="bean_id"
property="property_name" />
```

OR

```
<jsp: useBean id=" bean_id"
class="package_name.class_name">
....
....
<jsp:setProperty name=" bean_id"
property="property_name" />
</jsp:useBean>
```

It also possible to use '*' in property_name, which indicates that all the request parameters which matches to the Bean's property will be passed to the corresponding setter method.

6. **<jsp:getProperty> Action**

This action helps to retrieve the value of Bean's property.

☞ Syntax of <jsp:getProperty>

```
<jsp: useBean id=" bean_id"
class="package_name.class_name" />
....
....
<jsp:getProperty name=" bean_id "
property="property_name" />
```

OR

```
<jsp: useBean id=" bean_id"
class="package_name.class_name">
....
....
<jsp:getProperty name=" bean_id"
property="property_name" />
</jsp:useBean>
```

☞ Other Action Tags

There are some other tags which are not frequently used.

1. <jsp:plugin> Action

   This tag is used at the situation when there is requirement of a plugin to execute a Bean class or an Applet.

2. <jsp:body> Action
3. <jsp:element> Action
4. <jsp:text> Action
5. <jsp:attribute> Action

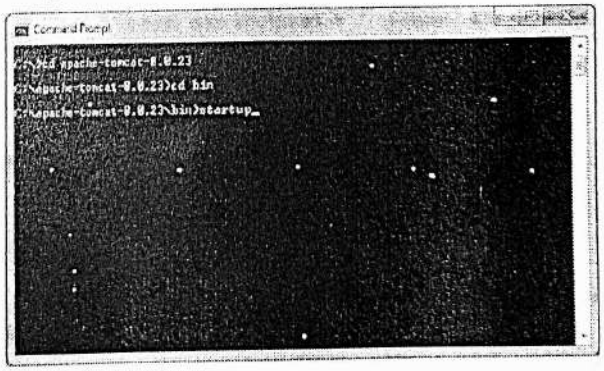
Running JSP Program

```
Index.jsp
<html>
 <head>
<title>First JSP</title>
</head>
<body>
This is my first JSP page.
</body>
</html>
```

☞ Running JSP file

▶ **Step 1 :** Start Tomcat server

- For this, you need to open the **Command Prompt** and go to the **bin** directory of the **Tomcat** folder present in the **C:** Drive, such as **C:\apache-tomcat-9.0.2\bin** and now you type **startup**(a *command which starts the Tomcat Server*) and press **Enter**.



- Tomcat web server is up and running and we don't have to close the server window till the time we are working on Tomcat.

▶ **Step 2 :** Making sure that Tomcat is running

Module 3

- Just to make sure that Tomcat web server is running is to open your web browser and enter the URL - **http://localhost:8080/**, which opens the window with a congratulatory message for successful installation and execution of Tomcat.

☞ Where to save the JSP(.jsp) file?

In order to allow Tomcat web server to easily find this JSP file and execute it, we can store this file in any of the two locations :

(1) In the **ROOT** folder (stored within **webapps** folder of your Apache Tomcat directory). Path to the ROOT folder is - *C:\apache-tomcat-9.0.2\webapps\ROOT* OR

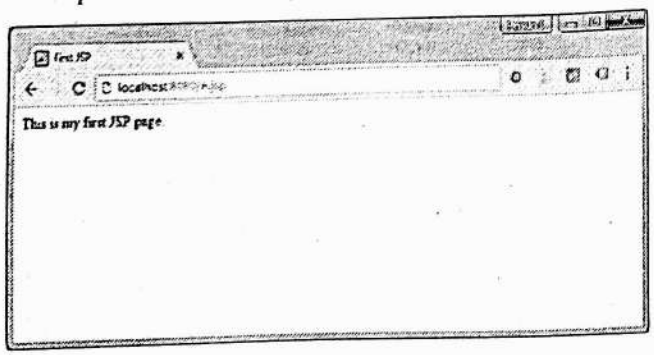
(2) We could create a new folder (with any name) in the **webapps** folder of Tomcat and store this **A.jsp** webpage in it. For example, we can create a new folder **first jsp** within the **webapps** folder of Tomcat and store **A.jsp** within it.

☞ Executing the JSP file

Next, we need to enter the URL of this JSP document in the address bar of our web browser. This URL may look a little different depending on where you store this file.

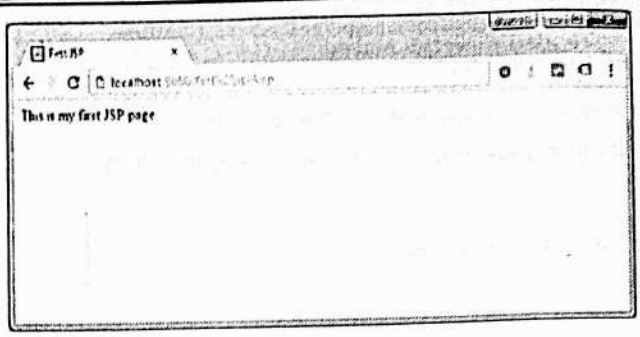
1) If JSP file is stored in the ROOT folder at location - *"C:\apache-tomcat-9.0.2\webapps\ROOT"*

The URL to execute this JSP webpage would be *http://localhost:8080/A.jsp*



2) If JSP file is stored in the a new folder created by you (within webapps folder) then we need to give the full path to the file (including the new folder which contains this JSP file).

For example, if we stored A.jsp in the folder first jsp then the URL to execute this file would be http://localhost:8080/first jsp/A.jsp.



☞ JSP Programs

➢ **Program 3.8.1 :** Write a simple JSP program to print message.

```
hello.jsp
<html>
<head>
<title>JSP Test Page</title>
</head>
<body>
<%
out.println("<h1>Hello World.</h1>");
out.println("<h1>This is my first JSP Program.</h1>");
%>
</body>
</html>
```

Output

Hello World.

This is my first JSP Program.

➢ **Program 3.8.2 :** To display date in JSP.

```
<%@page language="java" isThreadSafe= "false"
import="java.io.*,java.util.*" %>
<html>
  <body bgcolor="white">
<%
    Date now = new Date();
    out.println(now.toString());
    out.println(now.toGMTString());
%>
    </body>
</html>
```

➤ **Program 3.8.3 :** Working with HTML Form in JSP.

form.html

```
<html>
  <head>
    <title>JSP Form Demo</title>
    <style type="text/css">
      label{ margin-right:20px;}
      input{ margin-top:5px;}
    </style>
  </head>
  <body>
    <form action="handleUserInfo.jsp" method="post">
      <fieldset>
        <legend>User Information</legend>
        <label for="fistName">First Name</label>
        <input type="text" name="firstName" /> <br/>
        <label for="lastName">Last Name</label>
        <input type="text" name="lastName" /> <br/>
        <label for="email">Email</label>
        <input type="text" name="email" /> <br/>
        <input type="submit" value="submit">
      </fieldset>
    </form>
  </body>
</html>
```

handleUserInfo.jsp

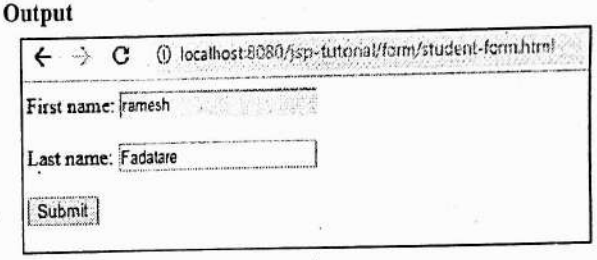
```
<html>
  <head>
    <title>JSP Form Demo</title>
  </head>
  <body>
    <%
String firstName = request.getParameter("firstName");
String lastName = request.getParameter("lastName");
String email = request.getParameter("email");
    %>
    <p>Hi <%=firstName%> <%=lastName%>!,
your submitted email is <%=email%>.</p>
  </body>
</html>
```

➤ **Program 3.8.4 :** Read HTML forms input text field value in JSP

student-form.html

```
<html>
<head>
  <title>Student Registration Form</title>
</head>
<body>
  <form action="student-response.jsp" method="post">
First name: <input type="text" name="firstName" />
    <br/><br/>
Last name: <input type="text" name="lastName"/>
    <br/><br/>
<input type="submit" value="Submit" />
  </form>
</body>
</html>
```

Output

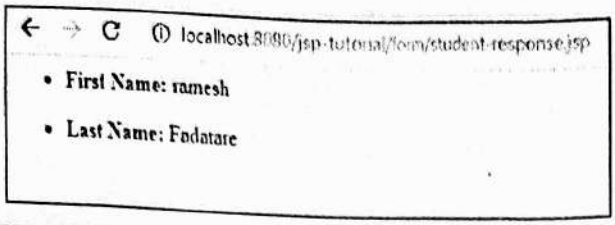


student-response.jsp

```
<html>
<head><title>Student Confirmation Title</title></head>
<body>
    <ul>
      <li><p><b>First Name:</b>
        <%= request.getParameter("firstName") %>
      </p></li>
      <li><p><b>Last  Name:</b>
        <%= request.getParameter("lastName") %>
      </p></li>
    </ul>
</body>
</html>
```

Module 3

Output

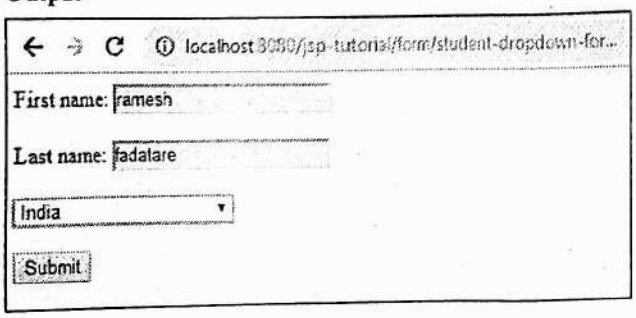


➢ **Program 3.8.5 :** Read HTML forms select box (drop-down list) value in JSP.

student-dropdown-form.html

```
<html>
<head>
  <title>Student Registration Form</title>
</head>
<body>
  <form action="student-dropdown-response.jsp">
    First name: <input type="text" name="firstName" />
    <br/><br/> Last name: <input type="text" name="lastName" />
    <br/><br/>
    <select name="country">
  <option>Brazil</option>
  <option>France</option>
  <option>Germany</option>
  <option>India</option>
  <option>Turkey</option>
  <option>United Kingdom</option>
  <option>United States of America</option>
</select>
    <br/><br/>
    <input type="submit" value="Submit" />
  </form>
</body>
</html>
```

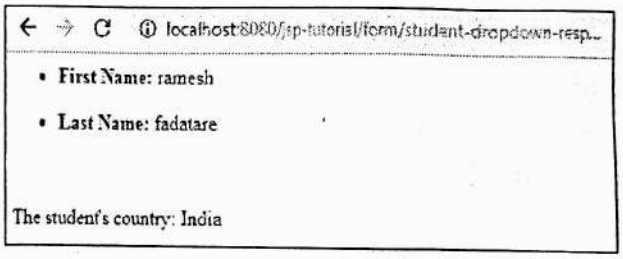
Output



student-dropdown-response.jsp

```
<html>
<head>
<title>Student Confirmation Title</title>
</head>
<body>
  <ul>
    <li><p><b>First Name:</b>
      <%= request.getParameter("firstName") %>
    </p></li>
    <li><p><b>Last Name:</b>
      <%= request.getParameter("lastName") %>
    </p></li>
  </ul>
<br />
<br /> The student's country: ${param.country}
</body>
</html>
```

Output



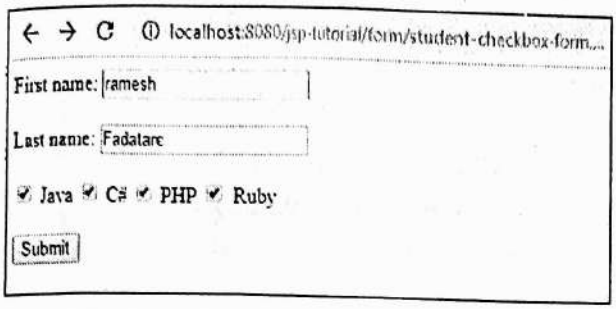
➢ **Program 3.8.6 :** Read HTML Form checkbox field value.

student-checkbox-form.html

```
<html>
<head>
  <title>Student Registration Form</title>
</head>
<body>
  <form action="student-checkbox-response.jsp" method="post">
    First name: <input type="text" name="firstName" />
    <br/><br/> Last name: <input type="text" name="lastName" />
    <br/><br/>
    <input type="checkbox" name="favoriteLanguage" value="Java"> Java
    <input type="checkbox" name="favoriteLanguage" value="C#"> C#
```

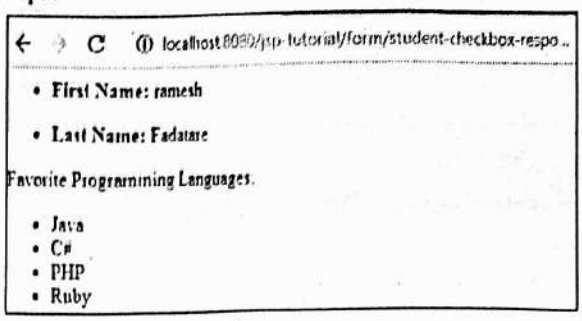
```
    <input type="checkbox" name="favoriteLanguage"
value="PHP"> PHP
    <input type="checkbox" name="favoriteLanguage"
value="Ruby"> Ruby
    <br/><br/>
    <input type="submit" value="Submit" />
  </form>
</body>
</html>
```

Output




```
student-checkbox-form.jsp
<html>
<head><title>Student Confirmation Title</title></head>
<body>
   <ul>
     <li><p><b>First Name:</b>
       <%= request.getParameter("firstName") %>
     </p></li>
     <li><p><b>Last  Name:</b>
       <%= request.getParameter("lastName") %>
     </p></li>
   </ul>
 Favorite Programming Languages: <br/>
  <!-- display list of "favoriteLanguage" -->
  <ul>
  <%
  String[] langs =
request.getParameterValues("favoriteLanguage");
   for (String tempLang : langs) {
  out.println("<li>" + tempLang + "</li>");
  }
 %>
</ul>
</body>
</html>
```

Output



➤ **Program 3.8.7 :** Read HTML Form's radio button value data.

```
student-form-radio.html
<html>
<head>
  <title>Student Registration Form</title>
</head>
<body>
  <form action="student-radio-response.jsp">
    First name: <input type="text" name="firstName" />
    <br/><br/> Last name: <input type="text"
name="lastName" />
    <br/><br/> Favorite Programming Language: <br/>
    <input type="radio" name="favoriteLanguage"
value="Java"> Java
    <input type="radio" name="favoriteLanguage"
value="C#"> C#
    <input type="radio" name="favoriteLanguage"
value="PHP"> PHP
    <input type="radio" name="favoriteLanguage"
value="Ruby"> Ruby
    <br/><br/>
    <input type="submit" value="Submit" />
  </form>
</body>
</html>
student-radio-response.jsp
<html>
<head><title>Student Confirmation Title</title></head>
<body>
  <ul>
    <li><p><b>First Name:</b>
      <%= request.getParameter("firstName") %>
    </p></li>
    <li><p><b>Last  Name:</b>
```

Module 3

```
        <%= request.getParameter("lastName") %>
      </p></li>
    </ul>
<br/> <br/>
The student's favorite programming language:
<%= request.getParameter("favoriteLanguage") %>
</body>
</html>
```

➤ **Program 3.8.8 :** User registration form in JSP.

userinfo.html

```
<html>
 <head>
  <title>User Info Entry Form</title>
 </head>
 <body bgcolor="white">
  <form action="userinfo.jsp" method="post">
   <table>
    <tr>
     <td>Name:</td>
     <td><input type="text" name="userName" >
     </td>
    </tr>
    <tr>
     <td>Birth Date:</td>
     <td><input type="text" name="birthDate" >
     </td>
     <td>(Use format yyyy-mm-dd)</td>
    </tr>
    <tr>
     <td>Email Address:</td>
     <td><input type="text" name="emailAddr" >
     </td>
     <td>(Use format name@company.com)</td>
    </tr>
      <tr>
     <td>Lucky Number:</td>
     <td><input type="text" name="luckyNo" >
     </td>
     <td>(Digits only)</td>
    </tr>
    <tr>
     <td colspan=2><input type="submit"></td>
    </tr>
   </table>
  </form>
 </body>
</html>
```

userinfo.jsp

```
<%@page language="java" isThreadSafe= "false"
import="java.io.*" %>
<html>
  <body bgcolor="white">
<%
    String name, bdate, email;
    int lucky;
    name = request.getParameter("userName");
    bdate = request.getParameter("birthDate");
    email = request.getParameter("emailAddr");
    lucky =
Integer.parseInt(request.getParameter("luckyNo"));
%>
        <ul>
            <li> Name : <%= name %> </li>
            <li> BirthDate : <%= bdate %> </li>
            <li> EMail : <%= email %> </li>
            <li> Lucky : <%= lucky %> </li>
        </ul>
    </body>
</html>
```

3.9 MULTIPLE CHOICE QUESTIONS

Q. 3.1 Which life cycle method is called once in servlet life ?

(a) Class loading (b) init()
(c) service() (d) destroy() ✓ **Ans. : (b)**

Q. 3.2 Which method is called when client request come?

(a) get() (b) post()
(c) init() (d) service() ✓ **Ans. : (d)**

Q. 3.3 Which method doesn't exist in HttpServlet class?

(a) service (b) init()
(c) doGet (d) doPost ✓ **Ans. : (b)**

Q. 3.4 Which interface contains servlet life-cycle method ?

(a) HttpServlet (b) GenericServlet
(c) Service (d) Servlet ✓ **Ans. : (d)**

Q. 3.5 Which of these life cycle method you can override in your class?

(a) init() (b) service()

(c) doGet() (d) All of these ✓ **Ans. : (d)**

Q. 3.6 Which statement is not true about ServletContext?

(a) There is one ServletContext per one application

(b) Generally it is used to get web app parameters

(c) We can get server information through it

(d) There is one ServletContext per Servlet ✓ **Ans. : (d)**

Q. 3.7 If you are not putting method name in your form tag, what is the default method?

(a) get (b) post

(c) service (d) None of the above ✓ **Ans. : (a)**

Q. 3.8 When init parameters are read by container?

(a) When container starts

(b) When doGet()/doPost() called

(c) When Servlet initialized

(d) When constructor called ✓ **Ans. : (c)**

Q. 3.9 Which of the following code is used to get an attribute in a HTTP Session object in servlets?

(a) session.getAttribute(String name)

(b) session.alterAttribute(String name)

(c) session.updateAttribute(String name)

(d) session.setAttribute(String name) ✓ **Ans. : (a)**

Q. 3.10 Which method is used to specify before any lines that uses the PintWriter?

(a) setPageType()

(b) setContextType()

(c) setContentType()

(d) setResponseType() ✓ **Ans. : (c)**

Q. 3.11 What are the functions of Servlet container?

(a) Lifecycle management

(b) Communication support

(c) Multithreading support

(d) All of the above ✓ **Ans. : (d)**

Q. 3.12 Which object of HttpSession can be used to view and manipulate information about a session?

(a) session identifier

(b) creation time

(c) last accessed time

(d) All mentioned above ✓ **Ans. : (d)**

Q. 3.13 Which of the following is true about servlets?

(a) Servlets execute within the address space of web server

(b) Servlets are platform-independent because they are written in java

(c) Servlets can use the full functionality of the Java class libraries

(d) Servlets execute within the address space of web server, platform independent and uses the functionality of java class libraries ✓ **Ans. : (d)**

Q. 3.14 Which JDBC driver Type(s) can be used in either applet or servlet code?

(a) Both Type 1 and Type 2

(b) Both Type 1 and Type 3

(c) Both Type 3 and Type 4

(d) Type 4 only ✓ **Ans. : (c)**

Q. 3.15 Which of the following code retrieves the MIME type of the body of the request?

(a) new MimeType()

(b) request.getContentType()

(c) response.getContentType()

(d) None of the above ✓ **Ans. : (b)**

Q. 3.16 Which methods are used to bind the objects on HttpSession instance and get the objects?

(a) setAttribute (b) getAttribute

(c) Both A & B (d) None of the above

✓ **Ans. : (c)**

Q. 3.17 To send text output in a response, the following method of HttpServletResponse may be used to get the appropriate Writer/Stream object.

(a) getStream (b) getOutputStream

(c) getBinaryStream (d) getWriter ✓ **Ans. : (c)**

Q. 3.18 Which method take a string not a URL?

(a) sendRedirect (b) forward

(c) Both (d) None ✓ **Ans. : (a)**

Q. 3.19 Which of the following are interface?

1. ServletContext 2. Servlet

3. GenericServlet 4. HttpServlet

(a) 1,2,3,4 (b) 1,2

(c) 1,3,4 (d) 1,4 ✓ **Ans. : (b)**

Q. 3.20 How many JDBC driver types does Sun define?

(a) One (b) Two

(c) Four (d) Three ✓ **Ans. : (c)**

Module 3

Q. 3.21 The life cycle of a servlet is managed by
(a) servlet context
(b) servlet container
(c) the supporting protocol (such as http or https)
(d) All of the above ✓ **Ans. : (b)**

Q. 3.22 Which of the following method sets application/context init parameter?
(a) setParameter(String name,String value)
(b) Its not possible in the method
(c) setParameter(String name,Object value)
(d) setParameter(Object name,Object value) ✓ **Ans. : (c)**

Q. 3.23 Which method shows the client what server is receiving?
(a) goGet (b) doOption
(c) doTrace (d) doPost ✓ **Ans. : (a)**

Q. 3.24 JSP stands for ___.
(a) Java Server Pages (b) Java Side Pages
(c) Java Server Phase (d) All of the above
✓ **Ans. : (a)**

Q. 3.25 What is JSP?
(a) Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.
(b) Java Server Pages (JSP) is a client-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.
(c) Both (a) and (b)
(d) None of the above ✓ **Ans. : (a)**

Q. 3.26 Which of the following is the correct syntax to declare comments in JSP?
(a) <%-- This is JSP comment --%>
(b) <!-- This is JSP comment -->
(c) //..
(d) All of the above ✓ **Ans. : (a)**

Q. 3.27 Which of the following is not a directive?
(a) include (b) page
(c) export (d) useBean ✓ **Ans. : (c)**

Q. 3.28 Which technology do we mix our business logic with the presentation logic?
(a) Servlet (b) JSP
(c) Both A & B (d) None of the above
✓ **Ans. : (a)**

Q. 3.29 Which JSP Action tags is used to include the content of another resource, it may be JSP, html or Servlet?
(a) <jsp:include > (b) <jsp:forward >
(c) <jsp:plugin > (d) <jsp:papam > ✓ **Ans. : (a)**

Q. 3.30 A JSP page consists of which tags?
(a) HTML tags
(b) JSP tags
(c) Both A & B
(d) None of the above ✓ **Ans. : (c)**

Q. 3.31 JDBC stands for ___.
(a) Java Database Connectivity
(b) Jar Database Connectivity
(c) Java Database Content
(d) None of the above ✓ **Ans. : (a)**

Q. 3.32 Is JSP technology extensible?
(a) True (b) False ✓ **Ans. : (a)**

Q. 3.33 What are the advantages of JSP?
(a) platform independent
(b) good performance and scalability
(c) easily maintained
(d) All of the above ✓ **Ans. : (d)**

Q. 3.34 Compilation process of JSP page involves:
(a) Parsing of JSP
(b) Turning JSP into servlet
(c) Compiling the servlet
(d) All of the above ✓ **Ans. : (d)**

Q. 3.35 What is the synatx of JSP Page directive?
(a) <%@ page...%> (b) <% page...%>
(c) <!@ page...!> (d) None of the above
✓ **Ans. : (a)**

Q. 3.36 Which syntax is used to defines the programming language (underlying language) being used in the JSP?
(a) <%@ page language="value" %>
(b) <!@ page language="value" !>
(c) <@ page language="value" >
(d) None of the above ✓ **Ans. : (a)**

Q. 3.37 A __ is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client.
(a) JSP expression (b) ASP
(c) PHP (d) None of the above
✓ **Ans. : (a)**

Q. 3.38 Which of the following is true about action?
(a) useBean action first searches for an existing object utilizing the id and scope variables. If an object is not found, it then tries to create the specified object.
(b) useBean is used to load java beans in jsp page.
(c) Both of the above.
(d) None of the above ✓ **Ans. : (c)**

Q. 3.39 Which one of the following is a syntax to create a simple JSP?
(a) <html> <body> <% out.print(2*5); %> </body> </html>
(b) <html> <body> <! out.print(2*5); !> </body> </html>
(c) <html> <body> <%@ out.print(2*5); %> </body> </html>
(d) None of the above ✓ **Ans. : (a)**

Q. 3.40 Which tag should be used to pass information from JSP to included JSP?
(a) Using <%jsp:param> tag
(b) Using <%jsp:page> tag
(c) Using <%jsp:useBean> tag
(d) Using <%jsp:import> tag ✓ **Ans. : (b)**

Q. 3.41 Default value of autoFlush attribute is?
(a) True (b) False ✓ **Ans. : (a)**

Q. 3.42 Which one is the correct order of phases in JSP life cycle?
(a) Compilation, Initialization, Execution, Cleanup
(b) Cleanup, Compilation, Initialization, Execution
(c) Initialization, Cleanup, Compilation, Execution
(d) Initialization, Compilation, Cleanup, Execution
✓ **Ans. : (a)**

Q. 3.43 Request is instance of which one of the following classes in jsp:
(a) ServletRequest (b) HttpServletRequest
(c) Request (d) HttpRequest ✓ **Ans. : (b)**

Q. 3.44 The difference between Servlets and JSP is:
(a) compilation (b) translation
(c) Both A and B (d) syntax ✓ **Ans. : (d)**

Q. 3.45 Which of the scripting of JSP not putting content into service method of the converted servlet?
(a) Declarations (b) Expressions
(c) Scriptlets (d) All of the above
✓ **Ans. : (b)**

Q. 3.46 What is the correct sequence to create a database connection?
i. Import JDBC packages.
ii. Open a connection to the database.
iii. Load and register the JDBC driver.
iv. Execute the statement object and return a query resultset.
v. Create a statement object to perform a query.
vi. Close the resultset and statement objects.
vii. Process the resultset.
viii. Close the connection.
(a) i, ii, iii, v, iv, vii, viii, vi
(b) i, iii, ii, v, iv, vii, vi, viii
(c) ii, i, iii, iv, viii, vii, v, vi
(d) i, iii, ii, iv, v, vi, vii, viii ✓ **Ans. : (b)**

Q. 3.47 Which of the following is not a valid statement in JDBC?
(a) Statement (b) PreparedStatement
(c) QueryStatement (d) CallableStatement
✓ **Ans. : (c)**

Q. 3.48 What should be the correct order to close the database resource?What should be the correct order to close the database resource?
(a) Connection, Statements, and then ResultSet
(b) ResultSet, Connection, and then Statements
(c) Statements, ResultSet, and then Connection
(d) ResultSet, Statements, and then Connection
✓ **Ans. : (d)**

Q. 3.49 Which of the following driver is the fastest one?
(a) JDBC-ODBC Bridge Driver
(b) Native API Partly Java Driver
(c) Network Protocol Driver
(d) JDBC Net Pure Java Driver ✓ **Ans. : (d)**

Module 3

Q. 3.50 Which of the following driver converts the JDBC calls into database-specific calls?

(a) JDBC-ODBC Bridge Driver (Type 1)

(b) Native API-partly Java Driver (Type 2)

(c) Net Protocol-pure Java Driver (Type 3)

(d) Native Protocol-pure Java Driver (Type 4)

✓ Ans. : (b)

Q. 3.51 Which of the following is the correct to register a JdbcOdbcDriver?

(a) jdbc.odbc.JdbcOdbcDriver obj = new sun.jdbc.odbc.JdbcOdbcDriver();

(b) odbc.JdbcOdbcDriver obj = new sun.odbc.JdbcOdbcDriver();

(c) jdbc.JdbcOdbcDriver obj = new sun.jdbc.JdbcOdbcDriver();

(d) jdbc.odbc.JdbcOdbc obj = new sun.jdbc.odbc.JdbcOdbc();

✓ Ans. : (a)

Chapter Ends...

□□□

MODULE 4

CHAPTER 4

Rich Internet Application (RIA)

Syllabus

Characteristics of RIA, Introduction to AJAX: AJAX design basics, AJAX vs Traditional Approach, Rich User Interface using Ajax, jQuery framework with AJAX.

4.1 Rich Internet Application (RIA) 4-2

4.1.1 Characteristics of RIA 4-2

4.1.2 Advantages of RIA 4-3

4.2 Introduction to AJAX 4-3

4.2.1 AJAX Design Basics 4-5

4.2.1.1 Design Patterns of AJAX 4-8

4.2.2 AJAX vs Traditional Approach 4-9

4.3 Rich User Interface using AJAX 4-12

4.4 jQuery framework with AJAX 4-13

4.4.1 jQuery AJAX Introduction 4-13

4.4.2 jQueryajax() Method 4-13

4.5 Multiple Choice Questions 4-15

- **Chapter Ends** 4-16

4.1 RICH INTERNET APPLICATION (RIA)

GQ. Write note on RIA.

- RIAs are web applications that have most of the characteristics of desktop applications, typically delivered through web-browser plug-ins or independently via sandboxes or virtual machines.
- The term "rich Internet application" was introduced in a white paper in March 2002 byMacromedia (now merged into Adobe).
- RIA is built with very powerful development tools which enable it to run faster.
- RIA provides users a better visual experience and more interactivity as compared to traditional browser applications which just use HTML and HTTP.
- RIAs have always been about the user experience, it enhancing the end-user experience in different ways.
- RIAs can run faster and be more engaging.
- RIA is developed using various technologies such as Java, Silverlight, JavaFX, JavaScript, REST/WS etc.
- Rich Internet Applications (RIA) can either run within a web browser with client-side scripts (JavaScript) and a browser plug-in or execute within a secure sandbox as desktop applications (e.g., Flash applications).
- For example, when registering at a web site, a simple validation of the username that it is an email can be done on the browser with a Javascript script, to give an interactive experience. Better still, users may remember search engines that support auto completion of search terms. Such experiences are possible with client-side scripting.
- An RIA normally runs inside a Web browser and usually does not require software installation on the client side to work. However, some RIAs may only work properly with one or more specific browsers.
- For security purposes, most RIAs run their client portions within a special isolated area of the client desktop called a sandbox.
- The sandbox limits visibility and access to the file and operating system on the client to the application server on the other side of the connection.
- This approach allows the client system to handle local activities, calculations, reformatting and so forth, thereby lowering the amount and frequency of client-server traffic, especially as compared to the client-server implementations built around so-called thin clients.

RIA Tools

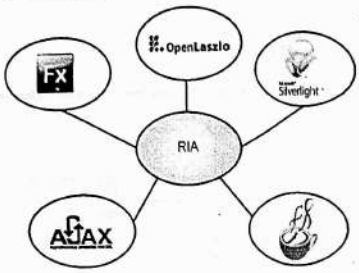


Fig. 4.1.1 : RIA Tools

1. **Adobe Flex :** Highly productive, open source application framework forbuilding and maintaining expressive web applications that deploy consistently onall major browsers, desktops and devices.
2. **OpenLaszlo :** Open source platform for the development and delivery of RichInternet Applications, consists of the LZX programming language and theOpenLaszlo Server.
3. **Microsoft Silverlight :** Powerful development tool for creating engaging,interactive user experiences for Web and mobile applications.
4. **JavaFX :** Software platform for creating and delivering desktop applications,as well as Rich Internet Applications (RIAs) that can run across a wide variety of devices.

Platforms of RIA

1. Adobe Flash
2. Java
3. Microsoft Silverlight
4. AJAX

4.1.1 Characteristics of RIA

GQ. What are the characteristics of RIA?

1. **Performance** - RIAs can often perform better than traditional applications on the basis of the characteristics of network and applications, performance of server also improved by offloading possible processing work to the client system and also perceived performance in terms of UI responsiveness and smoother visual transitions and animations are key aspects of any RIA.

2. **Offline use** - When connectivity is unavailable, it might still be possible to use an RIA. An RIA platform let the user work with the application without connecting to the Internet and synchronizing it automatically when the user goes live.
3. **Consistency of look and feel** - With RIA tools, the user interface and experience with different browsers, devices and operating systems can be more carefully controlled and made consistent.
4. **Security** - RIAs should be as secure as any other web application, and the framework should be well equipped to enforce limitations appropriately when the user lacks the required privileges, especially when running within a constrained environment such as a sandbox.
5. **Advanced Communications**- Sophisticated communications with supporting servers through optimized network protocols can considerably enhance the user experience.
6. **Rapid Development** - An RIA Framework should facilitate rapid development of a rich user experience through its easy-to-use interfaces in ways that help developers.
7. **Direct Interaction** - An RIA can use a wider range of controls that allow greater efficiency and enhance the user experience. In RIAs, for example, users can interact directly with page elements through editing or drag-and-drop tools. They can also do things like pan across a map or other image.
8. **Better Feedback** - Because of their ability to change parts of pages without reloading, RIAs can provide the user with fast and accurate feedback, real-time confirmation of actions and choices, and informative and detailed error messages.
9. **Improved Features** - RIA allow programmers to embed various functionalities in graphics-based web pages that look fascinating and engaging like desktop applications. RIA provide complex application screens on which various mixed media, including different fonts, vector graphic and bitmap files online conferencing etc. are paused by using different modern development tools.
10. **Partial-page updating** - RIAs incorporate additional technologies, such as real-time streaming, high-performance client-side virtual machines, and local caching mechanisms that reduce latency (wait times) and increase responsiveness.

4.1.2 Advantages of RIA

1. **Remotely accessed application :** The basic principle of RIA is the ability to have any new user connect and run the application from any location as long as they are connected to the network.
2. **Full interactive experience :** Unlike Web applications that provide page-by-page interaction and feedback, RIA provides a full interactive end-user experience.
3. **A Comprehensive Solution :** A comprehensive end-to-end solution facilitating full interactive distributed clients and a centralized managed server.
4. **A Single Unified IDE & Paradigm :** The RIA is supported by a single and unified IDE and development paradigm for defining server-side and client-side logic along with the user interface design.
5. **Automatic Logic Partitioning:** The RIA deployment modules facilitate optimized automatic logic partitioning between client and server.
6. **Performance-Aware Development :** The Rich Client Studio is a performance-aware platform for developing response-optimized applications.
7. **Native Look & Feel :** It provides an automatic reflection of the native look and feel of the selected client machine.
8. **Browser free solution :** RIA is independent of browser vendors and browser versions.

4.2 INTRODUCTION TO AJAX

GQ. What is AJAX?

- AJAX is an acronym for Asynchronous JavaScript and XML.
- It is not a programming language. It is a technology for developing better, faster and interactive Web Applications using HTML, CSS, JavaScript and XML.
  - **HTML :** Hypertext Markup Language (HTML) is used for defining the structure of a Web Application.
  - **CSS :** Cascading Style Sheet (CSS) is used to provide look and style to a Web Application.
  - **JavaScript :** JavaScript is used for making a Web Application interactive, interesting and user friendly.
  - **XML :** Extensible Markup Language (XML) is a format to store and transport data from the Web Server.
- It is a group of inter-related technologies like JavaScript, DOM, XML, HTML/XHTML, CSS, XMLHttpRequest etc.
- AJAX allows you to send and receive data asynchronously without reloading the web page.

Module 4

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

What's the meaning of Asynchronous in AJAX?

Asynchronous means that the Web Application could send and receive data from the Web Server without refreshing the page. This background process of sending and receiving data from the server along with updating different sections of a web page defines Asynchronous property/feature of AJAX.

Benefits of Ajax

1. Using AJAX you can create better, faster, and more user-friendly web applications.
2. Ajax is based on JavaScript, CSS, HTML and XML etc. So you can easily learn.
3. Ajax behavior and works is like a desktop application. So Ajax use for creating a rich web application.

Working of AJAX

It creates more interactive techniques for faster and more efficient web applications by using JavaScript, XML, CSS, and HTML. For various web applications, Ajax uses various techniques like this.

- In Ajax, when the user needs to create content, XHTML is used while CSS is used for presenting the user request purpose; the document object model (DOM) and javascript both will be used to display the content dynamically.
- By using synchronous methods in web applications, information can be transmitted and received effectively, for example, when you fill a form and submit it. You will be automatically directed to the new server with that page information.
- After hitting the submit button in the background, javascript sends a request, and with the response generated, it will update to the current screen. In this process, the user will be unaware of the background XML code requests.
- XML is used as a format to generate and receive the server data in any format.
- While most of the web browsers are dependent on web server technology, it is independent of web server software.

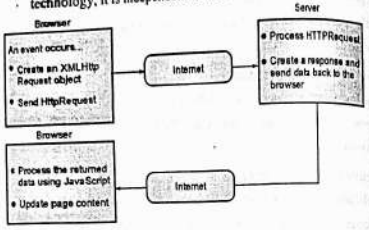


Fig. 4.2.1 : AJAX Working

Where to Use AJAX?

Below mentioned are the places where Ajax is used.

- **Login forms :** Eg: user can type their login credentials in the original page form; their software will send a request to the server to logged in, and the page will be updated as needed.
- **Auto-complete :** When you run a query in the Google search bar with the help of auto fill settings, suggestions will be shown in drop down below
- **Rating and Voting :** The voting can decide the site's main content in web pages like Digg and Reddit by bookmarking them.
- **Updating with user content :** When a user posts a tweet, it will be added to their feed, and everything is updated.
- Now it is used by tweeter to run their trending topics page
  - Form submission and validation
  - It makes web applications quicker, and the numbers of responses are also reduced.
  - Light-boxes are used nowadays instead of pop-ups
  - Using Ajax with flash application

Examples of Ajax Application

Given below are the lists of web applications that commonly use Ajax

1. Google suggests that auto-complete options will be offered while typing when a user enters the search query in the Google search bar. Suggestions given by Google can be navigated by using operational keys.

2. Yahoo maps are easier while operating, and user experiences more fun. This Map uses Ajax to drag the entire map with the mouse without using buttons that will be at ease to the user.
3. Google maps are general applications that use Ajax. This is a real-time application in which the user can manipulate the data and change the view settings. Ajax directly works on a web browser without any plugin installations. Firstly, only Microsoft internet explorer used Ajax, but due to its reliability, more web applications like chrome, Mozilla... etc. using this.

Advantages and Disadvantages of Ajax

Advantages

1. Reduces the server traffic and increases the speed
2. It is responsive, and the time taken is also less
3. Form validation
4. Bandwidth usage can be reduced
5. Asynchronous calls can be made; this reduces the time for data arrival.

Disadvantages

1. Open-source
2. Active x request is created only in internet explorer and a newly created web browser.
3. For security reason, you can only access information from the web host that serves pages. Fetching information from other servers is not possible with Ajax.

4.2.1 AJAX Design Basics

GQ. Explain about the object that helps AJAX reload parts of a webpage without reloading the whole page.

- AJAX uses the **XMLHttpRequest** object to communicate with the server. The Fig. 4.2.2 shows the flow of AJAX. Here you will get exact idea about the working of AJAX.
- An important role is played by the XMLHttpRequest object in AJAX processing.
- Request is sent by the user through UI (User interface) and a JavaScript call goes to XMLHttpRequest object.
- Using the XMLHttpRequest object, the HTTP Request sent to the server.
- Now the server interacts with the database using any of the server side scripting language like JSP, PHP, Servlet, or ASP.net.

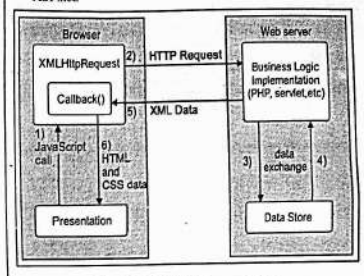


Fig. 4.2.2 : AJAX Communication

- Data is retrieved from the database as per the requirement.
- Server sends the data in the form of XML or JSON through the XMLHttpRequest callback function.
- Then the data is displayed using HTML and CSS on the browser.

AJAX Processing Steps and Ajax Script

GQ. Write steps to process AJAX.

Module 4

Steps to process AJAX

▶ **Step 1 :** An event by the client is fired.

When an event is fired, a JavaScript method gets called.

Example

validateID() JavaScript method is mapped as an event handler to an *onKeyPress* event on input form field. The id here is *"myid"*

```
<input type="text" size="25" id="myid" name="id"
onkeyPress="validateID();">.
```

▶ **Step 2 :** An XMLHttpRequest object is created.

```
var ajaxReqObj;
function ajaxFun(){
 try{
   ajaxReqObj = new XMLHttpRequest();
 }
catch (e)
{
```

This variable makes AJAX possible

Opera 8.0+, Firefox, Safari

```
    try{

      ajaxReqObj = new
activeXObject("Mexml2.XMLHTTP");
                    Internet Explorer Browser
    }
catch (e)
{
    try{
      ajaxReqObj = new
ActiveXObject("Microsoft.XMLHTTP");
    }catch (e){
      // Exception occurs
      alert("Your browser broke!");
      return false;
    }
  }
}
```

▶ **Step 3 :** The XMLHttpRequest object is configured.

1. In this step, we will write a function that will be triggered by the client event and a callback function processReqFun() will be registered.

```
function validateID() {
  ajaxFun();

  ajaxReqObj.onreadystatechange = processReqFun;
                    processReqFun() is callback function
  if (!target) target = document.getElementById("myid");
  var url = "validate?id=" + escape(target.value);
  ajaxReqObj.open("GET", url, true);
  ajaxReqObj.send(null);
}
```

▶ **Step 4 :** An asynchronous request is made by the XMLHttpRequest to the Webserver.

Here we will make request to webserver. This request can be made using XMLHttpRequest object *ajaxRequest*.

```
function validateID() {
  ajaxFun();
  ajaxReqObj.onreadystatechange = processReqfun;
                    processReqFun() is callback function
  if (!target) target = document.getElementById("myid");
  var url = "validate?id=" + escape(target.value);

  ajaxReqObj.open("GET", url, true);
  ajaxReqObj.send(null);
}
```

New consider in the textfield, Kunal as userid, then in the request, the url will be "validate?id=Kunal".

▶ **Step 5 :** The Webserver returns the result in XML format.

- The server side script can be implemented in any server side language using the following logic.
- Accept request from the client side.
- Parse the input given by the client.
- Execute necessary processing.
- Send the output to the client in response.
- We will use the Servlet for this purpose.

```
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException,
ServletException
{
  String ID = request.getParameter("id");

  if ((ID != null) && !accounts.containsKey(ID.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");
  }
}
```


▶ **Step 6 :** The callback() function is called by the XMLHttpRequest object and processes the result.

- When the state of *XMLHttpRequest* object is changed to the *readyState*, then the XMLHttpRequest object was configured to invoke (call) the processReqFun() method.
- The result returned by the server is received by this function and required processing will be done. Now see in the given example, this function will set message depending upon the webserver's returned value.

```
functionprocessReqFun()
{
if(req.readyState==4)
{
if(req.status==200)
{
varmsg=...;
...
}
```

▶ **Step 7 :** The HTML DOM is updated.

Now this is the last step. Here the HTML document will be updated. Process will be as follows:

- UsingDOMAPI, the JavaScript receives a reference to any element in the web page.

document.getElementById("userIdMsg"),

// " userIdMsg " is the ID attribute of an element present in the HTML document

- Using JavaScript, the attributes of element can be modified. The style properties can be modified or child elements can be added, modified or removed.

```
<scripttype="text/javascript">
<!--
functionsetMsgDOM(msg){
varuserMessageElement=document.getElementById("userIdM
sg");
varmsgText;
if(msg=="false")
{
userMessageElement.style.color="red";
msgText="Wrong UserId";
}
else
{
userMessageElement.style.color="green";
msgText="ValidUserId";
}
varmsgBody=document.createTextNode(msgText);

if(userMessageElement.childNodes[0])
{
                    If the messageBody element is already created
                    then just replace it otherwise append the new
                    element
userMessageElement.replaceChild(msgBody,userMessageEle
ment.childNodes[0]);
}
else
{
userMessageElement.appendChild(msgBody);
}
}
-->
</script>
<bodybgcolor="gray">
<divid="userIdMsg"><div>
</body>
```

GQ. What are two methods in ajax to update part of webpage? Describe them in detail.

- In Ajax, we frequently want to update just part of a Web page, not the whole page.
- Two methods enable us to do this: the **insertAdjacentHTML** method, which lets us to insert HTML next to an element that already exists, and the **insertAdjacentText** method, which lets us to insert text in the same way.
- We can determine where the new text or HTML will go with respect to the already existing element by passing the constants BeforeBegin, AfterBegin, Before End, or AfterEnd to insert AdjacentHTML and insert Adjacent Text.
- The example *insertAdjacent.html* uses the insert Adjacent HTMLmethod.
- This page starts with just a button, as shown in Fig. 4.2.3. When you click the button, the code inserts a new text field into the Web page, as shown in Fig. 4.2.4.

Module 4

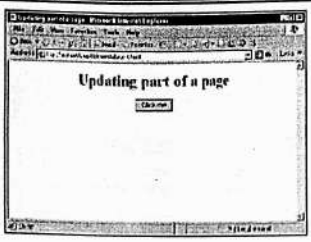


Fig. 4.2.3 :The insertAdjacent.html application

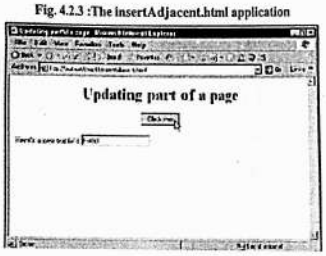


Fig. 4.2.4 : Inserting a new text field into a page

- Here's how it works: the application contains a element with the ID div1:

```
<body>
<center>
<h1>    Updating part of a page    </h1></center>
<div>    .    </div>
</body>
```

- And that element contains the button, which is connected to a JavaScript function named update:

```
<body>
<center>
<h1>    Updating part of a page    </h1></center>
<div>
<input type=button value="Click Me!"onclick="update()">
</div>
</body>
```

- In the update function, we can call the element's **insertAdjacentHTML** method to insert the text field next to the element :

```
<head>
<title>    Updating part of a page    </title>
<script language="JavaScript">
function update()
{
div1.insertAdjacentHTML(...);
}
</script>
</head>
```

- Here's how we can insert the new text field after the end of the element:

```
<head>
<title>    Updating part of a page    </title>
<script language="JavaScript">
function update()
    {
div1.insertAdjacentHTML("AfterEnd",    "<p>Here's A new
text field: <input type=text value='Hello!'></p>");
}
</script></head>
```

- Using insertAdjacentHTML and insertAdjacentText, we can insert HTML and text into a Web page, adjacent to another Web page item.

4.2.1.1 Design Patterns of AJAX

> **GQ.** Explain the following AJAX patterns :
> 1) Submission Throttling
> 2) Predictive Fetch 3) Fallback patterns

The three important design patterns of AJAX are discussed below.

1. Submission Throttling

- If retrieving data from the server is one part of the problem sending data to server is another.
- Science in AJAX page refreshes needs to be avoided, it is important to know when to send user data to the server.
- One approach that could be taken is to send data to server on every user interaction. But this results in a lot of requests submitted to the server is a short period.
- In case of submission throttling, the data to be send to the server is buffered on the client.
- This data is then sent to the server at predefined times.

- The delay from typing to sending data is fine-tuned such that it doesn't seem like a delay to the user.
- Then a client side function is invoked that begins buffering the data.
- It can be sent at a predefined time interval.
- This determination depends on the use case being used.
- After the data is sent, the application continuous to gather data.

2. Predictive Fetch

- In case of traditional web applications, the application reacts only when there is an interaction.
- This is called "fetch on demand". The user action tells the server what data should be retrieved.
- In the predictive fetch algorithm, the application guesses what the user is going to do next and retrieves the appropriate data.
- Determining the future action of the user is just a guess based on the user's intentions.
- For eg: say a user is reading an online article of 3 pages. It can be assumed here that if the user is reading the 1st page for few seconds, the person will also be interested in reading the 2nd page.
- Hence the 2nd page can be downloaded at the background before the user explicitly clicks on the 'Next'.
- Therefore when the user clicks on next, the 2nd page instantaneously appears reducing the response time.
- Similarly the 3rd page can be downloaded when the user reads 2nd page for a few seconds.
- This extra data being downloaded is cached on the client.
- Some approach can be applied in emails. If a person starts composing a mail, it is logical to anticipate that the mail would be sent to someone in the address book so this can be preloaded and kept.
- By using AJAX to fetch information related to any possible next step, can overload the server.
- Therefore this algorithm has to be implemented only when it is logical to assume that information will be requisite to completing the user's next request.

3. Fallback patterns

- All the above methods work fine when there is no problem at the server side.
- The following problems can occur :
  - The requestmight never make it to the server.
  - An error might occur at the server.

Cancel Pending Requests.

- If error occurs on the server like a file not fount or an internal server error, then it doesn't make sense to trying again after a few minutes.
- Such problems need an administrator to fix it.
- In such a situation all the pending requests are simply cancelled.
- This can be done by setting a flag somewhere in the code that says "don't send any more request".
- This solution has maximum impact on the periodic refresh pattern.

Try Again

- Another option is to silently keep trying for either a specified amount of time or a particular number of files.
- This problem is handled behind the scenes without generally notifying the user.
- Unless the AJAX functionality is key to the user's experience, it is not needed to notify user about the failures.

4.2.2 AJAX vs Traditional Approach

Module 4

> **GQ.** Draw the diagram of AJAX application model and traditional applications web model and compare them.

(A) Traditional Web Applications

- Fig. 4.2.5 represents the typical model of client and server interactions in a traditional web application, such as one which includes user registration form.
- In the first step, the form is filled by the user and then submitted. (*Step 1*).
- A request is generated by browser and sent to server. The server receives the request and processes it (Step 2).
- After processing the request, a response is generated by the server and the response containing the exact page that the browser will render (Step 3) is sent to the client.
- This entire process forces the browser to load the new page (*Step 4*) and for the time being makes the browser window blank.

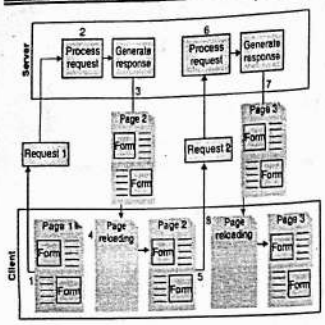


Fig. 4.2.5 : Traditional Web Application

- Remember that meanwhile the client has to wait for the server to give response and reloads the whole page containing the data from the response (Step 4).
- In time being when the asynchronous request is processing on the server, the user is not able to interact with the client web page.
- Frequent long periods of waiting impacts the site performance and led some users to refer to the World Wide Web as the "World Wide Wait."
- If the further user interacts with another form and submits it, the process is repeated (Steps 5-8).
- Basically this model was designed for a web of hypertext documents.
- As the web evolved into a full-scale applications platform, the model represented in Fig. 4.2.5 leads to irregular application performance.
- The full-page refresh needs the users to reset their understanding of the full-page contents.
- Users started to insist a model which would generate the responsive feel of desktop applications.

(B) Ajax Web Applications

GQ. Explain in detail AJAX Web application model with neat diagram.

- AJAX Web application model uses JavaScript and XMLHttpRequest object for asynchronous data exchange. The JavaScript uses the XMLHttpRequest object to exchange data asynchronously over the client and server.
- AJAX Web application model resolve the major problem of synchronous request-response model of communication associated with classical Web application model, which keeps the user in waiting state and does not provide the best user experience.
- AJAX, a new approach to Web applications, which is based on several technologies that help in developing applications with better user experience. It uses JavaScript and XML as the main technology for developing interactive Web applications.

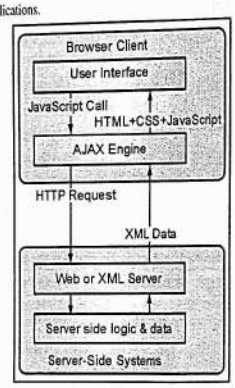


Fig. 4.2.6 : AJAX Web Application

- The AJAX application eliminates the start-stop-start-stop nature or the click, wait, and refresh criteria of the client-server interaction by introducing intermediary layer between the user and the Web server.
- Instead of loading the Web page at the beginning of the session, the browser loads the AJAX engine written in JavaScript.
- Every user action that normally would generate an HTTP request takes the form of a JavaScript call to the AJAX Engine.
- The server response comprises data and not the presentation, which implies that the data required by the client is provided by the server as the response, and presentation is implemented on that data with the help of markup language from Ajax engine.
- The JavaScript does not redraw all activities instead only updates the Web page dynamically.

- In JavaScript, it is possible to fill Web forms and click buttons even when the JavaScript has made a request to the Web server and the server is still working on the request in the background. When server completes its processing, code updates just the part of the page that has changed. This way client never has to wait around. That is the power of asynchronous requests.
- AJAX Engine between the client and the application, irrespective of the server, does asynchronous communication. This prevents the user from waiting for the server to complete its processing.
- The AJAX Engine takes care of displaying the UI and the interaction with the server on the user's behalf.
- In traditional Web applications, the synchronous mode of communication existed between the client and the server as shown in following Fig. 4.2.7.

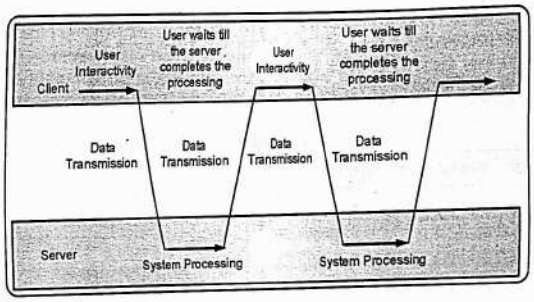


Fig. 4.2.7 : Synchronous mode of communication

- Since AJAX is essentially used for a partial update and asynchronous communication, the AJAX model used for programming and it is not restricted for use with specific data exchange format, specific programming language, or the specific communication mechanism.

Module 4

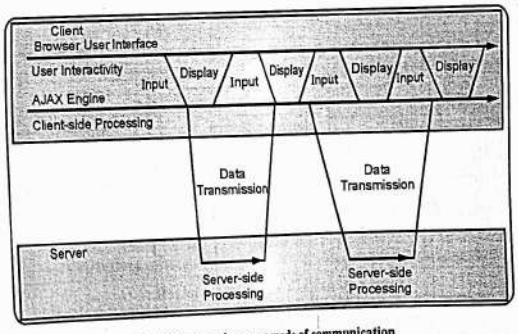


Fig. 4.2.8 : Asynchronous mode of communication

- Above diagram clarify that every user action generates an HTTP request that takes the form of a JavaScript to call the AJAX engine.
- Responses to the user actions do not involve the trip back to the server as in the classical Web application model. Instead, the AJAX Engine handles on its own, such as data validation, some navigation and editing data in memory.
- If the AJAX Engine needs something from the server, like retrieving new data or loading additional interface code, then the engine makes the asynchronous interaction with the server using JavaScript and XMLHttpRequest object for asynchronous data exchange.
- The engine's interaction with the server does not interrupt the user's interaction with the application. In this way, the asynchronous communication is done with the help of the AJAX engine.

4.3 RICH USER INTERFACE USING AJAX

GQ. Explain in detail RUI implementation using AJAX with neat diagram.

The architecture of RUI using AJAX works as follows.

- The web browser uses HTTP to send request for a page to the server which responds to the request in a standard manner.
- At this stage, no mashup content is present when the page is first loaded by the browser. The browser downloads some JavaScript and the HTML for the page.
- The browser issues another request to the server for additional content which is responded by the server in an asynchronous manner.
- The server acts as a proxy and is responsible for forwarding requests coming in from the browser to the intended recipients.
- Amazon receives a SOAP request.
- Fig. 4.3.1 shows that the architecture proceeds in further steps, where amazon responds the request with appropriate data.
- A REST request is forwarded by the server to Yahoo, which returns the response in an XML format.
- The web server then performs all the mashing operations, combines the responses and then sends back to the browser in the standard format.

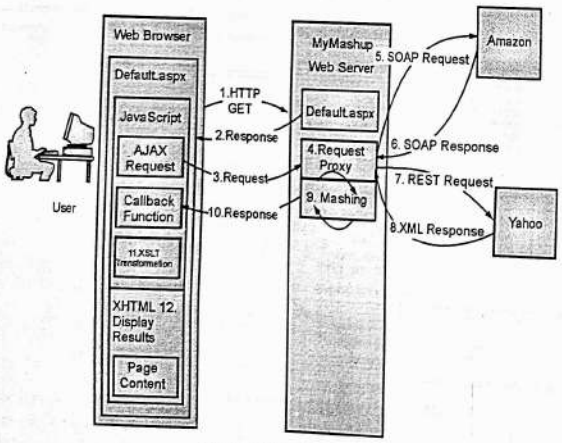


Fig. 4.3 1 : RUI using AJAX

4.4 JQUERY FRAMEWORK WITH AJAX

4.4.1 jQuery AJAX Introduction

- AJAX stands for "Asynchronous JavaScript and XML". JavaScript includes features of sending asynchronous http request using XMLHttpRequest object. Ajax is about using this ability of JavaScript to send asynchronous http request and get the xml data as a response (also in other formats) and update the part of a web page (using JavaScript) without reloading or refreshing entire web page.
- The jQuery library includes various methods to send Ajax requests. These methods internally use XMLHttpRequest object of JavaScript. The following table lists all the Ajax methods of jQuery.

jQuery Ajax Methods	Description
ajax()	Sends asynchronous http request to the server.
get()	Sends http GET request to load the data from the server.
Post()	Sends http POST request to submit or load the data to the server.
getJSON()	Sends http GET request to load JSON encoded data from the server.
getScript()	Sends http GET request to load the JavaScript file from the server and then executes it.
load()	Sends http request to load the html or text content from the server and add them to DOM element(s).

4.4.2 jQueryajax() Method

- The ajax() method in jQuery performs an AJAX request. It sends an asynchronous HTTP request to the server.
- JQuery provides a rich set of AJAX methods for developing web applications. It is widely used for the requests.

Syntax

```
$.ajax({name:value, name:value,... })
```

Parameter values

This method can have several name/value pairs for the AJAX requests. The names and values are defined in the following table.

Name	Value
async	It is a Boolean value. Its default value is **true**, which means that by default, the request is handled asynchronously. For synchronous requests, we can set it to **false**.
beforeSend(xhr)	It is a callback function that executes before a request is sent.
cache	It is a Boolean value. Its default value is **true**. It represents whether the browser cache the requested pages. On setting it to **false**, it forces the pages not to be cached by the browser.
complete(xhr, status)	It is a callback function executed when the request is finished. It has two arguments that are ***xhr (XMLHttpRequest)*** and a *status*. The *satu* can be "success", "notmodified", "nocontent", "error", etc.It is a callback function executed when the request is finished. It has two arguments that are xhr (XMLHttpRequest) and a status. The status can be "success", "notmodified", "nocontent", "error", etc.
contentType	It is used when sending the data to the server. Its default value is "application/x-www-form-urlencoded".
data	It is data sent to the server. It could be a string, an array, or a JSON object.
dataFilter(data, type)	It is a function that handles the raw response data of the XMLHttpRequest. It accepts two parameters.
dataType	It is the type of data we are expecting from the server.

Module 4

Name	Value
error(xhr, status, error)	It is a callback function that gets executed when a request fails. It accepts three arguments.
global	It is a Boolean value. Its default value is true. It represents whether to trigger a global AJAX event handler or not.
ifModified	It is a Boolean value. Its default value is false. It specifies a request to be successful if the response has changed since the last header.
jsonp	It is a string that overrides the callback function name in a jsonp request.
jsonpCallback	It is a string that contains the name of the callback function for a jsonp request.
password	It is used to specify a password that is to be used in an HTTP access authentication request.
processData	It is a Boolean value. Its default value is true. It is used to specify whether or not the data sent with the request should be converted into a query string.
scriptCharset	It specifies the charset for the request. It applies only when the "script" transport is used.
success(result, status, xhr)	As its name implies, this callback function executes when the request succeeds. It accepts three arguments.
timeout	It is the timeout for the request in terms of milliseconds. If it is set to the value 0, it means that there is no timeout.
traditional	It is a Boolean value. We can set it to true if we want to use the traditional style of param serialization.
type	It specifies the type of http request, such as POST, PUT, and GET. Its default value is GET.
url	It is the URL to which the request is sent. Its default value is the current page.
username	It is a username that is to be used in an HTTP access authentication request.
xhr	It is a callback function that is used for creating the XMLHttpRequest object.

Example

- First, we have to either download a jQuery library or find a link of jQuery's latest version by going to its official website.
- Second, we need to create an HTML document, including the jQuery library.
- Then, within the **<script>** tag write jQuery's **ready()** function, and write the **ajax()** method in its body. Now, we can pass the parameters to the **ajax()** function. We must have to include the URL to send the request as the first property. After **URL**, we can set any optional properties such as **type, async, cache, success, error, complete**, etc. to the ajax() function.
- Now, let's see the code of using the **ajax()** method. In this example, we are including the **URL** parameter and the optional type parameter. In this example, the **ajax()** method is used to add the content using ajax request.
- Here, we are passing the *test.html* to the **URL** parameter of the **ajax()** method.

```
test.html
<h1> Hello World </h1>
Example1.html
!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.
min.js"></script>
<script>
$(document).ready(function(){
 $("button").click(function(){
jQuery.ajax({
url: 'test.html',
type: 'GET',
success: function(data) {
  $("#para").html(data);
 }
});
```



```
});
});
</script>
</head>
<body>
<h3> This is an example of using the jQuery'sajax() method.
</h3>
<h4> Click the following button to see the effect. </h4>
<button> Click me </button>
<p id = "para"> </p>
</body>
</html>
```

Output

This is an example of using the jQuery's ajax() method.

Click the following button to see the effect.

Click me

After the execution of the above code, the output will be

Hello World

4.5 MULTIPLE CHOICE QUESTIONS

Q. 4.1 What of the following is the full form of RIA?
(a) RESTful Interface Application
(b) Rich Internet Application
(c) Rich Interface Application
(d) Residual Internet Application. ✓Ans. : (b)

Q. 4.2 What does AJAX stands for?
(a) Application on JavaScript and XML
(b) Applied JavaScript and XML
(c) Asynchronous JavaScript and XML
(d) None of the above ✓Ans. : (c)

Q. 4.3 Which of the following characteristics is supported by RIAs?
(a) Partial update of web pages
(b) No feedback to users
(c) Full update of web pages
(d) No interaction between user and application ✓Ans. : (a)

Q. 4.4 What sever support AJAX ?
(a) SMTP (b) WWW
(c) HTTP (d) None of the above ✓Ans. : (c)

Q. 4.5 What makes Ajax unique?
(a) It works as a stand-alone Web-development tool.
(b) It works the same with all Web browsers.
(c) It uses C++ as its programming language.
(d) It makes data requests asynchronously. ✓Ans. : (d)

Q. 4.6 What does the XMLHttpRequest object accomplish in Ajax?
(a) It's the programming language used to develop Ajax applications.
(b) It provides a means of exchanging structured data between the Web server and client.
(c) It provides the ability to asynchronously exchange data between Web browsers and a Web server.
(d) It provides the ability to mark up and style the display of Web-page text. ✓Ans. : (c)

Q. 4.7 Which of the following technologies provides the ability to dynamically interact with Web page layout?
(a) JavaScript. (b) XML.
(c) Document Object Model. (d) HTML. ✓Ans. : (c)

Q. 4.8 Ajax stands for Synchronous JavaScript and XML.
(a) True (b) False ✓Ans. : (b)

Q. 4.9 Ajax is a programming language.
(a) True (b) False ✓Ans. : (b)

Q. 4.10 Regarding the XMLHttpRequest Object, does the order in which one calls the "open" method and assign a function to the "onreadystatechange" event handler matter?
(a) Yes (b) No ✓Ans. : (b)

Q. 4.11 Assuming that "xmlhttp" is a user-defined variable that carrys the Ajax requesting object, xmlhttp.status can carry a value of 4 after a successful request.
(a) True (b) False ✓Ans. : (b)

Q. 4.12 Assuming that "xmlhttp" is a user-defined variable that carrys the Ajax requesting object, xmlhttp.status can carry a value of 0 after a successful request.
(a) True (b) False ✓Ans. : (b)

Q. 4.13 What combination of technologies gives AJAX its name?
(a) ASP and XAML
(b) Asynchronous JavaScript and X
(c) Autonomic Computing and DH
(d) Atlas and XML

Module 4

Q. 4.14 Which Web browser is the least optimizeqd for Microsoft's version of AJAX?

(a) Firefox (b) Opera
(c) Safari (d) Internet Explorer ✓**Ans. : (c)**

Q. 4.15 Which one of these technologies is NOT used in AJAX?

(a) CSS (b) DOM
(c) DHTML (d) Flash ✓**Ans. : (d)**

Q. 4.16 AJAX made popular by

(a) Microsoft (b) IBM
(c) Sun Microsystem (d) Google ✓**Ans. : (d)**

Q. 4.17 Using AJAX we can made our web page

(a) More interactive and faster
(b) Easy to connect web page with server
(c) More dynamic
(d) Static ✓**Ans. : (a)**

Q. 4.18 AJAX allows web page to dynamically

(a) Control other pages (b) Change content
(c) Reload at times
(d) Connect to other addresses ✓**Ans. : (b)**

Q. 4.19 Which methods are used to send a request to a server?

(a) open() and send() (b) open() and close()
(c) send() and back() (d) None of the above
✓**Ans. (a)**

Q. 4.20 jQuery is a

(a) JavaScript method (b) JavaScript library
(c) JSON library (d) PHP method ✓**Ans. : (b)**

Q. 4.21 Which of the following sign is used as a shortcut for jQuery?

(a) the % sign (b) the & sign
(c) the $ sign (d) the @ sign ✓**Ans. : (c)**

Q. 4.22 In Internet Explorer, from the following, which one is the right option to obtain XMLHttpRequest object?

(a) if (window.XMLHttpRequest) { httpReq = new XMLHttpRequest(); }
(b) if (window.ActiveXObject) { httpReq = new ActiveXObject("Microsoft.XMLHTTP"); }
(c) if (window.ActiveXObject) { httpReq= new GetObject("Microsoft XMLHTTPRequest"); }
(d) if (window.XMLHttpRequest) { httpReq = new XMLHttpRequest("IE"); } ✓**Ans. : (a)**

Q. 4.23 What will be the primary step when you have to send a request using JavaScript to a server?

(a) You will call the connect() method of the XMLHttp object
(b) You will call the execute() method of the XMLHttp object
(c) You will call the open() method of the XMLHttp object
(d) None of above ✓**Ans. : (c)**

Q. 4.24 _____method is use to send http GET request to load the data from the server.

(a) ajax() (b) getJSON()
(c) get() (d) load() ✓**Ans. : (c)**

Q. 4.25 _____method is use to send http GET request to load the JavaScript file from the server and then executes it.

(a) getScript() (b) load()
(c) ajax() (d) None of the above
✓**Ans. : (a)**

Chapter Ends...

❑❑❑

5 Web Extension PHP and XML

Syllabus

XML : DTD (Document Type Definition), XML Schema, Document Object Model, Presenting XML, Using XML Parsers : DOM and SAX, XSL-eXtensible Stylesheet Language.

Introduction to PHP : Data Types, Control structures, built-in functions, building web application using PHP – tracking users, PHP and MySQL database connectivity with example.

5.1 Introduction to XML

The XML acronym stands for Extensible Markup Language. As opposed to HTML tags, which specify how to display the data, XML tags identify the data and store it. XML is a text-based markup language based on Standard Generalized Markup Language (SGML).An XML document is a software- and hardware-independent method of storing and transporting data.

5.1.1 What is XML?

- XML is a markup language much like HTML
- The XML format was designed to store and transport data
- XML was designed as a self-describing language
- XML is an industry-standard recommended by W3C

XML is not a programming Language

A programming language consists of syntax and its own character set and keywords and identifiers which are used to create computer programs.

These programs tell the computer what to do. Since XML does not perform any computation or algorithms, it does not qualify as a programming language. It is typically stored in a simple text file and is processed by special software that can interpret XML.

XML Does Not DO Anything

Maybe it is a little hard to understand, but XML does not DO anything.

This note is a note to Ram from Ramesh, stored as XML :

```
<note>
 <to>Ram</to>
 <from>Ramesh</from>
 <heading>Inform</heading>
 <body>There is an important meeting on Friday!</body>
</note>
```

The XML above is quite self-descriptive:

- It has sender information.
- It has receiver information
- It has a heading
- It has a message body.
- XML is just information wrapped in tags.

5.1.2 Difference Between XML and HTML

HTML	XML
Is not case sensitive.	Is case sensitive.
Doubles up as a presentation language.	Is not a presentation language nor a programming language.
Has its own predefined tags.	Tags are not predefined it is flexible as tags can be defined whenever needed.
Closing tags are not necessarily needed. (Not required for all tags)	Closing tags are used mandatorily.
White spaces are not preserved.	Capable of preserving white spaces.
Used for displaying data.	Used for storing and transferring data.
Static in nature.	Dynamic in nature.
HTML was designed to display data - with focus on how data looks	XML was designed to carry data - with focus on what data is

5.1.3 XML is Simplified way of storing and transferring data

- Data on many computer systems is stored in incompatible formats. Web developers spend a lot of time exchanging data between incompatible systems (or upgraded systems).
- Converting large amounts of data often results in the loss of incompatible data. XML simplifies data sharing, data transportation, platform changes, and data availability. Data is stored in plain text format in XML. In other words, it offers a software- and hardware-independent way to store, transport, and share data.

- It is also simpler to upgrade to a new operating system, a new application, or a new browser without losing data through XML.
- XML allows digital data to be accessed by a wide range of devices, including people, computers, voice machines, RSS feeds, etc.

5.1.4 How to use XML?

- The use of XML by web developers is widespread. Separating data from the presentation is often done with XML. XML does not carry any information about how the data should be displayed.
- It does not have any information about the presentation of data. The same XML data can be used in many different presentation scenarios. Due to this, with XML there is a complete separation between data and presentation.

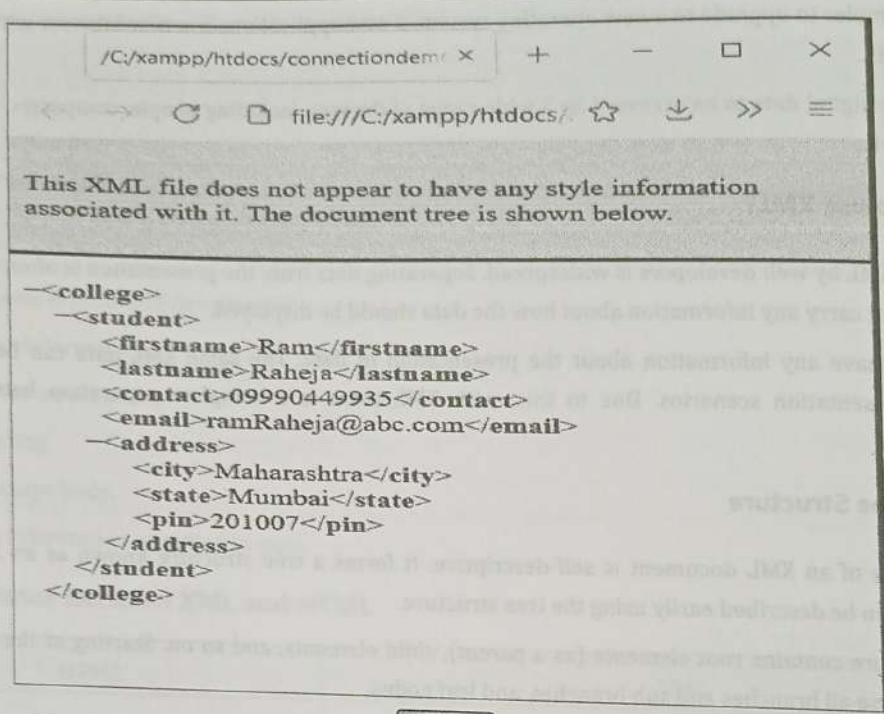
5.1.5 XML Tree Structure

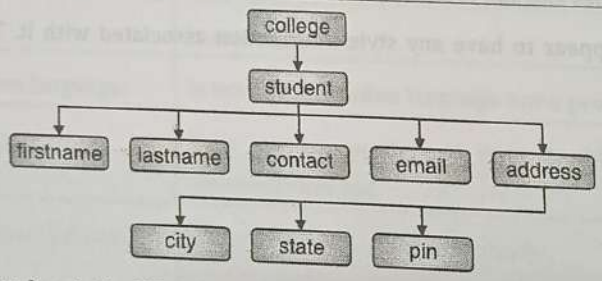
- The structure of an XML document is self-descriptive. It forms a tree structure known as an XML tree. XML documents can be described easily using the tree structure.
- A tree structure contains root elements (as a parent), child elements, and so on. Starting at the root, it is very easy to traverse all branches and sub branches, and leaf nodes.
- This XML file does not appear to have any style information associated with it. The document tree is shown below.

Syntax of tree structure

```
<root>
 <child>
  <subchild>.....</subchild>
 </child>
</root>
```

```
<?xml version="1.0"?>
<college>
 <student>
   <firstname>Ram</firstname>
   <lastname>Raheja</lastname>
   <contact>09990449935</contact>
   <email>ramRaheja@abc.com</email>      <address>
      <city>Maharashtra</city>
      <state>Mumbai</state>
      <pin>201007</pin>
   </address>
 </student>
</college>
```





In the above example, first line is the XML declaration. It defines the XML version 1.0. Next line shows the root element (college) of the document. Inside that there is one more element (student).

Student element contains five branches named <firstname>, <lastname>, <contact>, <Email> and <address>.<address> branch contains 3 sub-branches named <city>, <state> and <pin>.

5.1.6 Comments in XML

The syntax for writing comments in XML is similar to that of HTML:

```
<!-- This is a comment -->
```

Two dashes in the middle of a comment are not allowed:

```
<!-- This is an invalid -- comment -->
```

5.1.7 XML Elements

An XML document contains XML Elements.

What is an XML Element?

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

```
<price>29.99</price>
```

An element can contain:

text

attributes

other elements

or a mix of the above

```
<bookstore>
 <book category="children">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
 <book category="web">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
 </book>
</bookstore>
```

In the example above:

<title>, <author>, <year>, and <price> have **text content** because they contain text (like 29.99).

<bookstore> and <book> have **element contents**, because they contain elements.

<book> has an **attribute** (category="children").

Empty XML Elements

An element with no content is said to be empty.

In XML, you can indicate an empty element like this:

```
<element></element>
```

You can also use a so called self-closing tag:

```
<element />
```

5.1.8 XML Naming Rules

XML elements must follow these naming rules :

- Element names are case-sensitive

- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

Best Naming Practices

Create descriptive names, like this: <person>, <firstname>, <lastname>.

Create short and simple names, like this: <book_title> not like this: <the_title_of_the_book>.

Avoid "-". If you name something "first-name", some software may think you want to subtract "name" from "first".

Avoid ".". If you name something "first.name", some software may think that "name" is a property of the object "first".

Avoid ":". Colons are reserved for namespaces (more later).

Naming Styles

There are no naming styles defined for XML elements. But here are some commonly used:

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

If you choose a naming style, it is good to be consistent! XML documents often have a corresponding database. A common practice is to use the naming rules of the database for the XML elements.

5.1.9 XML DTD

XML DTD is acronym for document type definition which is used to define the structure of XML document. The XML document is called as Valid and Well Formed if it satisfies Document type Definition.

Following is the example of Document Type Definition

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE address [
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
]>
```

```
<address>
<name>TanmayPatil</name>
<company>XUYZ</company>
<phone>8785456655</phone>
</address>
! DOCTYPE address - Defines that the root element of the document is address
! ELEMENT address - Defines that the note element must contain the elements: "name, company, phone "
! ELEMENT name - Defines the name element to be of type "#PCDATA"
! ELEMENT company - Defines the company element to be of type "#PCDATA"
! ELEMENT Phone - Defines the phone element to be of type "#PCDATA"
```

5.1.10 XML Schema

Like Document Type Definition an XML Schema also describes the structure of an XML document. The XML document is called as Well-Formed document if it has correct syntax. IF An XML document is validated against an XML schema we can call it both "Well Formed" and "Valid".

XML Schema

XML Schema is an XML-based alternative to DTD :

```
<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:string"/>
 </xs:sequence>
</xs:complexType>
</xs:element>
```

The Schema above is interpreted like this:

```
<xs:element name="address">defines the element called "address"
<xs:complexType> the "note" element is a complex type
<xs: sequence> the complex type is a sequence of elements
<xs: element name="name" type="xs: string"> the element "name" is of type string (text)
<xs: element name="company" type="xs: string"> the element "company" is of type string
<xs: element name="phone" type="xs: string"> the element "phone " is of type string
```

XML Schemas are More Powerful than DTD

XML Schemas are written in XML

XML Schemas are extensible to additions

XML Schemas support data types

XML Schemas support namespaces

5.1.11 XML DOM : XML Document Object Model

- DOM (Document Object Model) is the foundation of XML. XML documents contain nodes, which are hierarchical units of information; DOM describes those nodes and their relationships.
- A DOM document is a collection of nodes or pieces of information arranged in a hierarchy. A developer can find specific information in the tree by navigating through this hierarchy. The DOM is called a tree because it is based on a hierarchy of information.
- A developer can add, edit, move, or remove a node within the XML DOM at any point in the process of creating an application using the API provided by XML DOM.
- All XML elements can be accessed through the XML DOM.
- This code retrieves the text value of the first <firstname> element in an XML document:

```
<?xml version="1.0"?>
<college>
 <student>
   <firstname>Ram</firstname>
   <lastname>Raheja</lastname>
   <contact>09990449935</contact>
   <email>ramRaheja@abc.com</email>
<address>
      <city>Maharashtra</city>
      <state>Mumbai</state>
      <pin>201007</pin>
   </address>
 </student>
</college>
```

Example

```
txt = xmlDoc.getElementsByTagName("firstname")[0].childNodes[0].nodeValue;
```

Explanation of Example

xmlDoc - the XML DOM object created by the parser.

getElementsByTagName("title")[0] - get the first <title> element

childNodes[0] - the first child of the <title> element (the text node)

nodeValue - the value of the node (the text itself)

5.1.12 Programming Interface

- **The DOM models XML as a set of** node objects. The nodes can be accessed with JavaScript or other programming languages. In this tutorial we use JavaScript.
- The programming interface to the DOM is defined by a set standard properties and methods.
- **Properties** are often referred to as something that is (i.e. nodename is "book").
- **Methods** are often referred to as something that is done (i.e. delete "book").

XML DOM Properties

These are some typical DOM properties:

x.nodeName - the name of x

x.nodeValue - the value of x

x.parentNode - the parent node of x

x.childNodes - the child nodes of x

x.attributes - the attributes nodes of x

Note : In the list above, x is a node object.

XML DOM Methods

x.getElementsByTagName(*name*) - get all elements with a specified tag name

x.appendChild(*node*) - insert a child node to x

x.removeChild(*node*) - remove a child node from x

5.1.13 XML Parser

XML Parsers are software libraries or packages that allow client applications to interact with XML documents. Checks the format of the XML document as well as validates it if necessary. Browsers today have XML parsers built-in.XML parsers convert XML into a readable format.

Example XML Parser

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
```

```
<script>
var parser, xmlDoc;
var text ="<college><student>"+
    "<firstname>Ram</firstname>"+
    "<lastname>Raheja</lastname>"
    "<contact>09990449935</contact>"+
    "<email>ramRaheja@abc.com</email>"+
 "</student></college>"
parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");
document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("lastname")[0].childNodes[0].nodeValue;
</script>
</body>
</html>
```

Output

```
Raheja
:
```

Explanation of Example

A text string is defined:

```
var text ="<college><student>"+
    "<firstname>Ram</firstname>"+
    "<lastname>Raheja</lastname>"
    "<contact>09990449935</contact>"+
    "<email>ramRaheja@abc.com</email>"+
 "</student></college>"
```

An XML DOM parser is created:

```
parser = new DOMParser();
```

The parser creates a new XML DOM object using the text string:

```
xmlDoc = parser.parseFromString(text,"text/xml");
```

5.1.14 XSLT Example

The recommended style sheet language for XML is XSLT (eXtensible Stylesheet Language Transformations).XSLT is much more sophisticated than CSS. XSLT allows you to add and remove elements and attributes from an output file. Also, you can rearrange and sort elements, perform tests and make decisions about which elements to display and hide, and much more.

XPath is used by XSLT to find information in an XML document.

We will use the following XML document

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<College>
<Student>
<name>Ramesh</name>
<Subjects>C programming </Subjects>
<Hobbies>Singing.</Hobbies>
</Student>
<Student>
<name>Ram</name>
<Subjects>Python </Subjects>
<Hobbies>Dancing.</Hobbies>
</Student>
</College>
```

XLST

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Info</h2>
<table border="1">
<trbgcolor="#9acd32">
<th>Student</th>
<th>Subjects</th>
</tr>
```

```
<tr>
<td><xsl:value-of select="College/Student/name"/></td>
<td><xsl:value-of select="College/Student/Subjects"/></td>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

5.2 Introduction to PHP

- PHP is an acronym for "Hypertext Preprocessor". Originally it was named as Personal Home Page .PHP is an HTML embedded; general purpose server-side scripting language used for web development.In 1994 RasmusLerdorf designed PHP.PHP was developed by Zend Technologies.
- PHP is server-side scripting language which allows developer to build logic into creation of web page. PHP scripts are embedded within the HTML documents that are severed by web server.

5.2.1 What is scripting Language?

- All the scripting languages are programming languages. In scripting, language instructions are written for run time environment that they do not require compilation rather they are interpreted. Scripting language translates the instructions into machine-level language at the run time.
- It does not have an intermediate compilation step. Scripting languages are used in web applications. It is used in server side as well as client side. Server-side scripting languages are: JavaScript, PHP, Perl etc. and client-side scripting languages are: JavaScript, VBScript etc.

5.2.2 Features of PHP language

1. **Simple**

   It is very simple and easy to use, compared to another scripting language it is very simple and easy, this is widely used all over the world.

2. **Interpreted**

   It is an interpreted language, i.e. there is no need for compilation.

3. **Faster**

   It is faster than other scripting languages e.g. asp and jsp.

4. **Open Source**

   Open source means you no need to pay for using PHP, you can free download and use.

5. **Platform Independent**

PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.

6. **Case Sensitive**

PHP is case sensitive scripting language at the time of variable declaration. In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

5.3 Data Types, Control Structure, Built in Functions etc.

5.3.1 My First PHP program to print "Hello World"

Example 5.3.1

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP Example</h1>
<?php
echo "Welcome to PHP!";
?>
</body>
</html>
```

My first PHP page

HTML Code

Welcome To PHP!

PHP Script

Explanation of above Program

As you can see in the above example PHP script normally contains some HTML tags and some scripting code. We can put script anywhere in the document.

A PHP script starts with **<?php** and ends with**?> Basic Syntax is as Follows:**

```
<?php//indicates start of PHP code
// PHP code goes here
?>//indicates end of PHP code
```

The default extension of PHP file is **.php**. Above Example also uses built in PHP function **"echo"** to print the text "Welcome to PHP" on the web page.IN PHP all the Statements ends with **semicolons (;)**.

5.3.2 Comments in PHP

Comments are the non-executable program statements written to make the program more understandable and readable. Comments Let others understand your code. Comments also useful to remind yourself of what you did - IF programmer is coming back to their work a year or two later and has to re-figure out what they did. Comments can remind him of what he was thinking when he wrote the code.

Example 5.3.2

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "// symbol is used for single line comment ";
// This is a single-line comment
echo"<br>";
echo "# symbol is used for single line comment ";
# This is also a single-line comment
echo"<br>";
echo "/* */ symbol is use for multiple line comment ";
/*multiple lines of comments */
  ?>
</body>
</html>
```

Example of Commnets in PHP

// symbol is used for single line comment
symbol is used for single line comment
/* */ symbol is use for multiple line comment

Explanation:

The above program shows us how to add single and multi-line comments to PHP scripts. Single-line comments can be added with // (double black slash) and # (hash). In multiline comments, /*------- */ (a star symbol and a slash symbol) can be used.

5.3.3 Variable in PHP

In programming, variables are used to hold information. A variable is a name that represents a value. Variables are used to store data, such as numbers, characters, strings, and memory addresses, so that we can use them in other parts of the program.

In PHP, a variable starts with the $ (dollar) sign, followed by the name of the variable:

Example 5.3.3 : Variable in PHP

```
<!DOCTYPE html>
<html>
<h1> Variable Demo </h1>
<body>
<?php
$str = "Sting Variable!"; //Assigning String to the variable
$num = 5;//Assigning integer number to the variable
$decimalno = 10.5;//Assigning decimal no to the variable
echo $str;  //Displaying Value of variable
echo "<br>";
echo $num;
echo "<br>";
echo $decimalno;
?>
</body>
</html>
```

Variable Demo

Sting Variable!
5
10.5

Explanation

As a result of the statements above, the variable $str will hold the value String Demo! the variable $num will hold the value 5, and the variable $decimal no will hold the value 10.5. To assign text values to variables, you must put quotes around them. As we have done for variable $str. There is no separate command in PHP for declaring variables like in other programming languages. They are created when a value is assigned to them.

Rules for PHP variables

- A variable can have a short name (like a and b) or a more descriptive name (age, firstname, marks).
- A variable start with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

Variable names are case-sensitive ($age and $AGE are two different variables)

Example 5.3.4 : Displaying value of variable

```
<!DOCTYPE html>
<html>
<h1> Displaying value of variable</h1>
<body>
<?php
$str = "Variable Demo";
echo "Variable str contains $str! value";
?>
</body>
</html>
```

Example 5.3.4 shows how to display value of the variable on the screen

Displaying value of variable

Variable str contains Variable Demo! value

Explanation

The echo statement in the above example is used to display the output on the screen. It allows text and values within the variable to be displayed on the screen. In the example above, echo is used to display both the variable's value and text as well.

Example 5.3.5 : Example Adding two numbers

```
<!DOCTYPE html>
<html>
<h1> Example adding two numbers </h1>
<body>
<?php
```

```
$x = 5;
echo "Value of x= $x <br>";
$y = 4;
echo "Value of x= $x <br>";
$z=$x + $y;
echo "Value of z= $z";
?>
</body>
</html>
```

Example adding two numbers

Value of x= 5
Value of x= 5
Value of z= 9

Explanation

The data types of the variables x, y, and z have not been declared in the example above. PHP automatically assigns a data type to the variables based on their values. Due to the fact that the data types are not set up strictly, you can add a string to an integer without causing an error. In PHP 7, type declarations were added. This allows you to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" if a type mismatch occurs.

5.3.4 PHP Data Types

A variable can store various types of data, and different types of data can perform different tasks.

The following data types are supported by PHP:

1. String
2. Integer
3. Float (floating point numbers - also called double)
4. Boolean
5. Array
6. Object
7. NULL
8. Resource

1. PHP String

The term string refers to a sequence of characters, such as "String in PHP!".

Strings can be any text enclosed in quotes. It's up to you whether to use single or double quotations :

Example

```
<?php
$str1 = "String in PHP!";
$str2 = Sequence of characters!';
echo $str1;
echo "<br>";
echo $str2 ;
?>
```

2. PHP Integer

The integer data type represents a non-decimal number between -2,147,483,648 and 2,147,483,647.

For integers, follow these rules :

(a) At least one digit must be present in an integer

(b) Integers cannot have decimal points

(c) An integer can be either positive or negative

(d) You can specify integers in decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

$x is an integer in the following example. The PHP var_fun () function returns the following data types and values :

Example

```
<?php
$x = 500;
echo "value of x is =$x<br>" ;//displaying the value of integer variable x
var_fun($x) ;//passing value of variable x to the function var_fun
?>
```

3. PHP Float

The term float (a floating-point number) refers to a number with a decimal point or an exponential number. $x is a float in the following example. The PHP var_fun () function returns the data type and value: Example

```
<?php
$x ="value of float variable x is =$x<br>";//displaying the value of integer variable x
var_fun($x); //passing value of variable x to the function var_fun
?>
```

4. PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter

5. **PHP Array**

An array stores multiple values in one single variable.

In the following example $student is an array. The PHP var_fun () function returns the data type and value:

Example

```
<?php
$student = array("ram","ramesh","radha","","radhika",);
var_fun ($student);
?>
```

You will learn a lot more about arrays later in this chapter

Example 5.3.6 : Data types in PHP

```
<!DOCTYPE html>
<html>
<h2>Data types in PHP</h2>
<body>
<?php
echo "Displaying value of string variable<br>";
$var="var is string";//here var is assigned a string value so data type of var is string
echo $var;
echo"<br>";
echo "Displaying value of Integer variable<br>";
$var=5;//here var is assigned a integer value so data type of var is integer
echo $var;
echo"<br>";
echo "Displaying value of Float variable<br>";
$var=55.5;//here var is assigned a float value so data type of var is float
echo $var;
echo"<br>";
echo "Displaying value of Boolean variable<br>";
$var=true;//here var is assigned a Boolean value so data type of Boolean is integer
echo $var;
echo"<br>";
?>
</body>
</html>
```

Output

```
Data types in PHP
Displaying value of string variable
var is string
Displaying value of Integer variable
5
Displaying value of Float variable
55.5
Displaying value of Boolean variable 1 indicates True Value
1
```

5.3.5 PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

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

1. Arithmetic operators

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

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

Example 5.3.7 : Arithmetic operators

```
<!DOCTYPE html>
<html>
<h2>Arithmetic Operator + - * / % **</h2>
<body>
<?php
$x = 10;
$y = 6;
$z=$x + $y;//Addition operator
echo "Addition of X and Y is = $z <BR>";
$z=$x - $y;//Subtraction operator
echo "Subtraction of X and Y is = $z <BR>";
$z=$x * $y;//Multiplication operator
echo "Multiplication of X and Y is = $z <BR>";
$z=$x / $y;//Division operator
echo "Division of X and Y is = $z <BR>";
$z=$x % $y;//Modulus operator
echo "Modulus of X and Y is = $z <BR>";
$z=$x ** $y;//Exponential operator
echo "Exponential of X Raise to Y is = $z <BR>";
?>
</body>
</html>
```

Airthmetic Operator + - * / % **

Addition of X and Y is = 16
Subtraction of X and Y is = 4
Multiplication of X and Y is = 60
Division of X and Y is = 1.6666666666667
Modulus of X and Y is = 4
Exponential of X Raise to Y is = 1000000

2. PHP Assignment Operators

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

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

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

Example 5.3.8 : Assignment Operator

```
<!DOCTYPE html>
<html>
<h2> Assignment Operator </h2>
<body>
<?php
$x = 20;
$x += 100;//Composite Assignment (Addition) Operator += performs x=x+100;
echo "X=$x<br>";
$y=30;
$y-=10;//Composite Assignment (Subtraction) Operator -= performs y=y-10;
echo "Y=$y<br>";
$z=10;
$z*=2;//Composite Assignment(multiplication) Operator *= performs z=z*10;
echo "Z=$z<br>";
$m=50;
$m/=5;//Composite Assignment (Division) Operator /= performs m=m/5;
echo "M=$m<br>";
$n=15;
$n%=2;//Composite Assignment (Modulus) Operator %= performs n=n%2;
echo "N=$n<br>";
?>
</body>
</html>
```

Assignment Operator

X = 120

Y = 20

Z = 20

M = 10

N = 1

3. PHP Comparison Operators

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

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

4. PHP Increment / Decrement Operators

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

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

Example 5.3.9 : Increment/Decrement Operartor

```
<!DOCTYPE html>
<html>
<h2>Increment Decrement Operator</h2>
<body>
<?php
$x = 10; $y=10; $z=10; $w=10;
echo "Pre-Increment"; echo ++$x ;//Pre-Increment
echo"<br>";
echo "post-Increment";echo $y++;//post-Increment
echo"<br>";
echo "Pre-Decrement"; echo --$z;//Pre-Decrement
echo"<br> "; echo "post-Decrement";
echo $w--;//post-Decrement
echo"<br>";
?>
</body></html>
```

Increment Decrement

- Pre-Increment 11
- Post-Increment 10
- Pre-Decrement 9
- Post-Decrement 10

5. PHP Logical Operators

The PHP logical operators are used to combine conditional statements. By using logical operator, we can combine more than one conditional statements. Following is the list of logical operators supported by PHP

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

Example 5.3.10 : logical Operator

```
<!DOCTYPE html>
<html>
<h2> PHP logical Operator</h2>
<body>
<?php
$x = 100;
$y = 50;
if ($x == 100 and $y == 50) {
echo "Both the conditions are true thtas why result of logical AND operator is true <br> ";
}
if ($x == 100 or $y == 50)
echo "Both the conditions are true thtas why result of logical OR operator is true ";
if ($x == 100 Xor $y == 50)
echo "Both the conditions are true thtas why result of logical OR operator is true ";
?>
</body>
</html>
```

Output

PHP logical Operator

- Both the conditions are true thetas why result of logical AND operator is true.
- Both the conditions are true thtas why result of logical OR operator is true
- The Condition ($x=100 and $y=50) is true, that why respective message in if block is executed Condition ($x=100 OR $y=50) is true, that's why respective message in if block is executed, Condition ($x=100 Xor $y=50) is true, that's why respective message in if block is not executed.

6. PHP String Operators

PHP has two operators that are specially designed for strings.

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

Example 5.3.11 : String Operator

```
<!DOCTYPE html>
<html>
<h2>string concatination using string operator</h2>
<body>
<?php
$txt1 = "String Concatination";
$txt2 = " Using dot operator!";
echo $txt1 . $txt2;
?>
</body>
</html>
```

string concatenation using string operator

string concatenation Using dot operator

7. PHP Array Operators

The P/HP array operators are used to compare arrays.

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

8. PHP Conditional Assignment Operators

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

Operator	Name	Example	Result
?/:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE

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

Example of Ternary Operator

5.3.6 Control Structure

- In computing, control flow is the flow of instructions in which they are executed. **Programs use Control Structures to specify the flow of control.**
- It is much easier to understand any algorithm or program if it uses self-contained modules called logic or control structures. Based on certain parameters and conditions, it analyses a program's flow and determines how to proceed. There are three types of logic flow :

  (A) Sequence logic, or sequential flow

  (B) Selection logic, or conditional flow

  (C) Iteration logic, or repetitive flow

(A) Sequential Control Structure

The instructions are executed in the sequence they have written. Simple program of addition of two numbers is the example of sequential control structure.

(B) Selection logic, or conditional flow

In selection logic or conditional flow, the instructions are executed best on the result of some condition. It's very common when you write code to perform different actions depending on the situation. You can use conditional statements in your code to do this. In PHP we have the following conditional statements :

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

1. PHP - The if Statement

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

Syntax

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

Example 5.3.12 : if statement

```
<!DOCTYPE html>
<html>
<h2> Example of if condition</h2>
<body>
<?php
$age = 28;
if ($age >= 18) {
echo "You are eligible for voting";
}
?>
</body>
</html>
```

You are eligible for voting

2. PHP - The if...else Statement

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

Syntax

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

Example 5.3.13 : If else statement

```
<!DOCTYPE html>
<html>
<h2> Example of if condition</h2>
<body>
```

```
<?php
$age = 15;
if ($age >= 18) {
echo "You are eligible for voting";
}
else{
    echo "You are not eligible for voting";
}
?>
</body>
</html>
```

You are not eligible for voting

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

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

Syntax

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

5.3.7 Developing Web Application Using PHP

1. Introduction

To begin with we Need to know following Basic technologies required to develop Web Application Using PHP

(a) **HTML**: is the markup language used for structuring and presenting content on the World Wide Web. (Done in chapter 1)

(b) **CSS :** describes how HTML elements are to be displayed on a screen, paper, or in other media. (Done in chapter 1)

(c) **PHP :** is a highly popular server-side scripting language. It is very flexible and has a lot of innovative features. It is the fastest resource available for creating database-enabled dynamic websites.

(d) **MySQL** : is a database management system based on Structured Query Language (SQL) which is used in web applications. (Will learn in the later in this chapter)

For front end development we will require HTML & CSS

For Back-end Development we will require PHP & MySQL

Following is the list of the software which we will require for development of web application using PHP & MYSQL

(a) Text Editor (HTML and CSS)

(b) Web Server (PHP)

(c) Back End Database (MySQL)

(d) Web Browser

Text Editor : First, you will need a text editor to create and edit your program. There are many options on the internet, but I prefer to use notepadd++ because it is free and easy to use.

Local host : Typically, the local host in computer networking is the computer you are working on, also called "this PC". Local is useful for testing purposes if you are a developer, network administrator, Developers use localhost most often when creating web apps or programs that require an internet connection. As part of the development process, we test the application on localhost to ensure that it functions properly.

Setting up Server

Having PHP and MySQL installed on your system is necessary to work on it locally. The first step is to have a server on which we are going to run our application. It is possible to use XAMPP (Cross-Platform, Apache, MySQL, Perl, PHP) because it installs both PHP and MySQL for us.

Once you have done with installation part,

you should go to the directory where you installed XAMPP perform following steps.

(a) go to htdocs folder.

C:\Program Files\XAMPP\htdocs

(b) Inside of your htdocs, you should create your folder.

C:\Program Files\XAMPP\htdocs\my_folder

(c) Then, open your Notepadd++ and create a new file inside the folder and name it

C:\Program Files\XAMPP\htdocs\my_folder\

(d) Inside the file Write your code and save it as .php extension

2. Web Application for Alumni registration using PHP.

Alumni students will be able to register themselves to the college database online, and colleges will be able to collect data on their alumni students through the web application.

Write down the following Code in Text Editor and Save it By Registration.php

```
<html>
<head>
<title>Alumni Registration Form</title>
<style>
```

```
table {
 width: 30%;
 border-collapse: collapse;
 text-align: left;

th, td {
        padding: 0px;
        spacing: 0px;
        background-color: none;
      }
td{
  text-align: left;
}
</style>
<script>
function form validator()
{
  var i;
  var n=document.frm1.txtfnm.value;
  if(n==="")
  {
    alert("Enter some first name.");
    document.frm1.txtfnm.focus();
    return false;
  }
  else
  {
  //alert("aaaddd");
    document.frm1.action="http://localhost/Sample/add_alu.php";
    document.frm1.submit();
    return true;
  }
}
```

Java Script For Form Validation

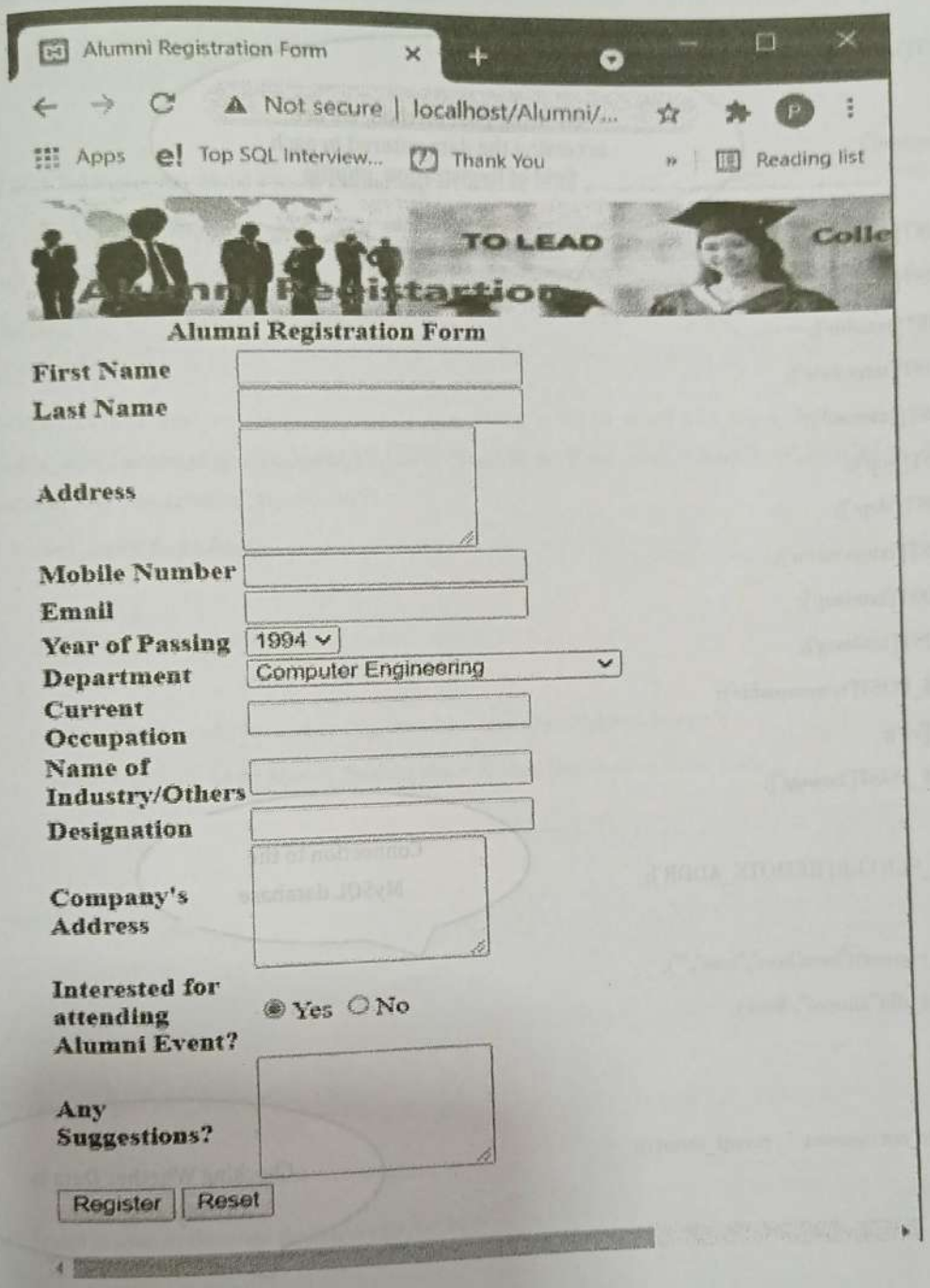
```
</script>
</head>
<body>
<img src="college-banner.jpg" width=700 height=75> </img>
<form name="frm1">
<table>
<tr>
<th colspan=2 align="center">Alumni Registration Form</th>
</tr>
<tr>
<th colspan=2></th>
</tr>
<tr>
<th>First Name</th>
<td><input type=text name=txtfnm></td>
</tr>
<tr>
<th>Last Name</th>
<td><input type=text name=txtlnm></td>
</tr>
<tr>
<th>Address</th>
<td><textarea name=txtaddr rows=5 cols=16> </textarea></td>
</tr>
<tr>
<th>Mobile Number</th>
<td><input type=text name=txtmobile></td>
</tr>
<tr>
<th>Email</th>
<td><input type=text name=txtemail></td>
</tr>
<tr>
```

PHP Script for Year of Passing List

```
<th>Year of Passing</th>
<td>
<select name=yop>
<?php
for ($x=1994; $x<=2013; $x++)
 {
  echo "<option value=$x>$x</option>";
  echo "The number is: $x <br>";
 }
?>
</select>
</td>
</tr>
<tr>
<th>Department</th>
<td>
<select name=dept>
<option value="Computer Engineering">Computer Engineering</option>
<option value="Information Technology">Information Technology</option>
<option value="Instrumentation">Instrumentation</option>
<option value="Electronics & Telecommunication">Electronics & Telecommunication</option>
<option value="Mechnical Engineering">Mechnical Engineering</option>
</select>
</td>
</tr>
<tr>
<th>Current Occupation</th>
<td><input type=text name=txtcurroccu></td>
</tr>
<tr>
<th>Name of Industry/Others</th>
<td><input type=text name=txtcomp></td>
</tr>
```

```
<tr>
<th>Designation</th>
<td><input type=text name=txtdesig></td>
</tr>
<tr>
<th>Company's Address</th>
<td><textarea name=txtcompaddr rows=5 cols=16> </textarea></td>
</tr>
<tr>
<th>Interested for attending Alumni Event?</th>
<td><input type=radio name="r1" value=yes checked>Yes</input>
<input type=radio name="r1" value=no>No</input></td>
</tr>
<tr>
<th>Any Suggestions?</th>
<td><textarea name=txtsugg rows=5 cols=16> </textarea></td>
</tr>
<tr>
<th colspan=2></th>
</tr>
<tr>
<td colspan=2>
<input type="button" name=b1 value="Register" onClick="return formvalidator()">
<input type=Reset name=b2 value="Reset">
</left></td>
</tr>
</table>
</form>
</body>
</html>
```

Output



Upon Fill the Form the Data will be Submitted to The Database. We have to create the Database and the table inside the database with the required number of fields. We are going to use MySQL database for storing and retrieving data to and from database.

Following PHP Script, we have used to perform Alumni Registration by adding the data to the alumni database.

```
<?php
$bt=$_POST['b1'];

if($bt=='Register')
{
$fnm=$_POST['txtfnm'];
$lnm=$_POST['txtlnm'];
$add=$_POST['txtaddr'];
$mob=$_POST['txtmobile'];
$eml=$_POST['txtemail'];
$yop=$_POST['yop'];
$dept=$_POST['dept'];
$curr=$_POST['txtcurroccu'];
$comp=$_POST['txtcomp'];
$desig=$_POST['txtdesig'];
$compaddr=$_POST['txtcompaddr'];
$r1=$_POST['r1'];
$suggestion=$_POST['txtsugg'];

$ipaddress=$_SERVER['REMOTE_ADDR'];

$con = mysql_connect("localhost","root","");
  mysql_select_db("alumni", $con);
    if (!$con)
    {
     die('Could not connect: ' . mysql_error());
    }
    else
    {
 $sq="SELECT * FROM alumni_reg where a_email_id='$eml'";
 $res1=mysql_query($sq,$con);
 if(mysql_num_rows($res1)>0)
 {
```

BY using post method, we are accessing the data entered in each field of Registration .phpfile

Connection to the MySQL database

Checking Whether Data is Already Available

```
    echo "ALUMNI already exists.";
  }
  else
  {
    $sql="SELECT MAX(txn_no) AS ID FROM alumni_reg";
    $res=mysql_query($sql,$con);
    $row=mysql_fetch_array($res);
    $mid=$row['ID'];
    $mid=$mid+1;
    $sql1="INSERT INTO alumni_reg (txn_no,a_fnm,a_lnm,a_addr,a_mobile,a_email_id,a_yop,a_dept,a_curr_occu,a_indus_
nm,a_desig,a_indus_addr,Interest,suggestion,ipaddr) VALUES($mid,'$fnm','$lnm','$add','$mob','$eml',$yop,'$dept','$curr','$com
p','$desig','$compaddr','$r1','$suggestion','$ipaddress')";
        if (!mysql_query($sql1,$con))
          {
            die('Error: ' . mysql_error());
          }
            echo "<body><h1>You have registered successfully.</h1></body>";
            echo "<br><a href=Alumni_Student.php> Alumni Registration Form </a>";
      }
    }
    mysql_close($con);
  }
  else
  {
    header('Location:/Sample/add_alu.php');
  }
  ?>
```

Inserting Data To The alumni_reg Table

Following PHP Script is used to Generate the Report Report.PHP

```
<?php
$bt=$_POST['b2'];
$con=mysqli_connect("localhost","vcetvasa_alumni","alumni","vcetvasa_alumni");
$yop=$_POST['yop'];
$dept=$_POST['dept'];
$r1=$_POST['r1'];
```

```
// Check connection
if (mysqli_connect_error())
 {
 echo "Failed to connect to MySQL: " . mysqli_connect_error();
 }
$result = mysqli_query($con,"SELECT * FROM alumni_reg where Interest='yes'");
echo "<table border='1'>
<tr>
<th>Sr. No.</th>
<th>Firstname</th>
<th>Lastname</th>
<th>Address</th>
<th>Mobile</th>
<th>Email</th>
<th>Year Of Passing</th>
<th>Department</th>
<th>Current Occupation</th>
<th>Name Of Industry</th>
<th>Designation</th>
<th>Company Address</th>
<th>Interested?</th>
<th>Suggestions</th>
</tr>";
$i = 1;
while($row = mysqli_fetch_array($result))
 {
 echo "<tr>";
 echo "<td>" . $i . "</td>";
 echo "<td>" . $row['a_fnm'] . "</td>";
 echo "<td>" . $row['a_lnm'] . "</td>";
 echo "<td>" . $row['a_addr'] . "</td>";
 echo "<td>" . $row['a_mobile'] . "</td>";
 echo "<td>" . $row['a_email_id'] . "</td>";
```

```
echo "<td>" . $row['a_yop'] . "</td>";
echo "<td>" . $row['a_dept'] . "</td>";
echo "<td>" . $row['a_curr_occu'] . "</td>";
echo "<td>" . $row['a_indus_nm'] . "</td>";
echo "<td>" . $row['a_desig'] . "</td>";
echo "<td>" . $row['a_indus_addr'] . "</td>";
echo "<td>" . $row['Interest'] . "</td>";
echo "<td>" . $row['suggestion'] . "</td>";
echo "</tr>";
$i = $i + 1;
}
$i = $i - 1;
echo "</table>";
echo "<h5> Total Count: " . $i . "</h5>";
echo "<br><a href=First.php> Display Records of Alumni Registered </a>";
mysqli_close($con);
?>
```

5.3.8 PHP and MYSQL database Connectivity

- A database is a structured collection of records or data. The data in the database is organized in such a way that it is often quickly searched and knowledge are often rapidly retrieved. The SQL in MySQL stands for Structured command language.
- This language is l based on English and also utilized in other databases like Oracle and Microsoft SQL Server. It's designed to permit simple requests from a database via commands like

  SELECT title FROM publications WHERE author = 'ABC';

  A MySQL database contains one or more tables, each of which contains records or

  rows. Within these rows are various columns or fields that contain the info itself.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server which Can be used for both small and large applications.
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows. Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

How to connect PHP to MySQL to store and retrieve data to and from database in web application.

You can use PHP to connect to MySQL to store and retrieve data in a web application. Following is the syntax to connect PHP to MySQL

Open a Connection to MySQL

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

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

MySQLi and PDO both have their advantages:

MySQLi only supports the MySQL database, whereas PDO can connect to 12 different databases. Therefore, if you need to switch your project to use another database, PDO makes it simple. Only the connection string and a few queries need to be changed. When using MySQLi, you will need to rewrite the entire code, including the queries. Both databases are object-oriented, but MySQLi also offers a procedural API.

Example (MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
 die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
try {
 $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
 // set the PDO error mode to exception
 $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
 echo "Connected successfully";
} catch(PDOException $e) {
 echo "Connection failed: " . $e->getMessage();
}
?>
```

Close the Connection

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

MySQLi Object-Oriented:

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

MySQLiProcedural

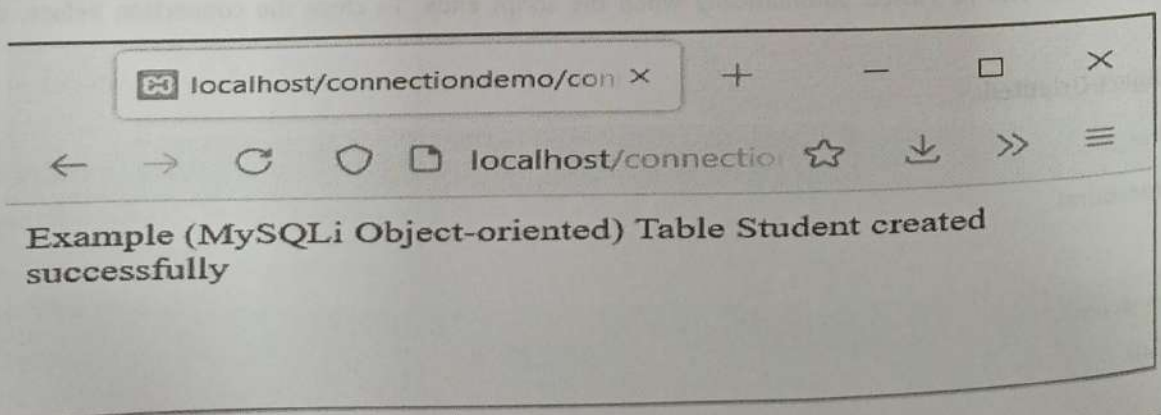
PDO

```
mysqli_close($conn);
$conn = null;
```

Create Table in Database using PHP script

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "connection_demo";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}
// sql to create table
$sql = "CREATE TABLE Students (
Rollno INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50))";
if ($conn->query($sql) === TRUE) {
 echo "Table Student created successfully";
} else {
 echo "Error creating table: " . $conn->error;
}
$conn->close();
?>
```

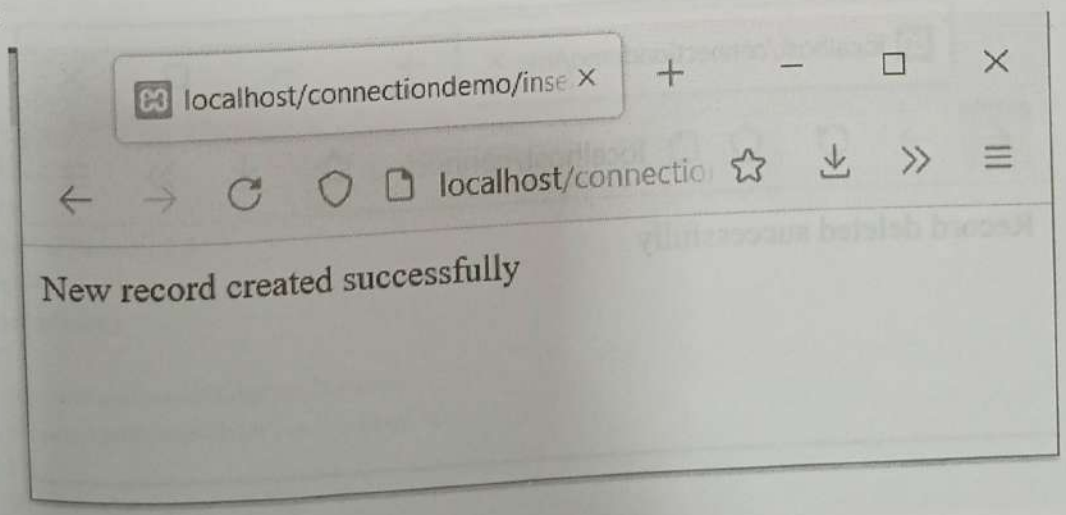
Output



Insertion of Record in the Table using PHP script

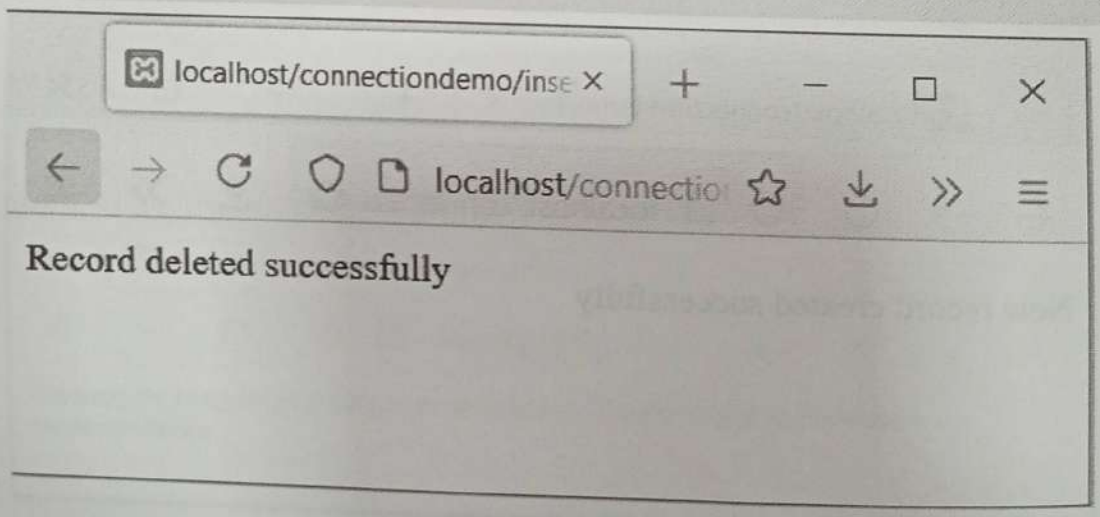
```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "connection_demo";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO students (Rollno,firstname, lastname, email)
VALUES ('1', 'John', 'Doe', 'john@example.com')";
if ($conn->query($sql) === TRUE) {
 echo "New record created successfully";
} else {
 echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

Output



Deletion of record using PHP script

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "connection_demo";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
// sql to delete a record
$sql = "DELETE FROM students WHERE Rollno=1";

if ($conn->query($sql) === TRUE) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . $conn->error;
}
$conn->close();
?>
```



Login Form HTML Code

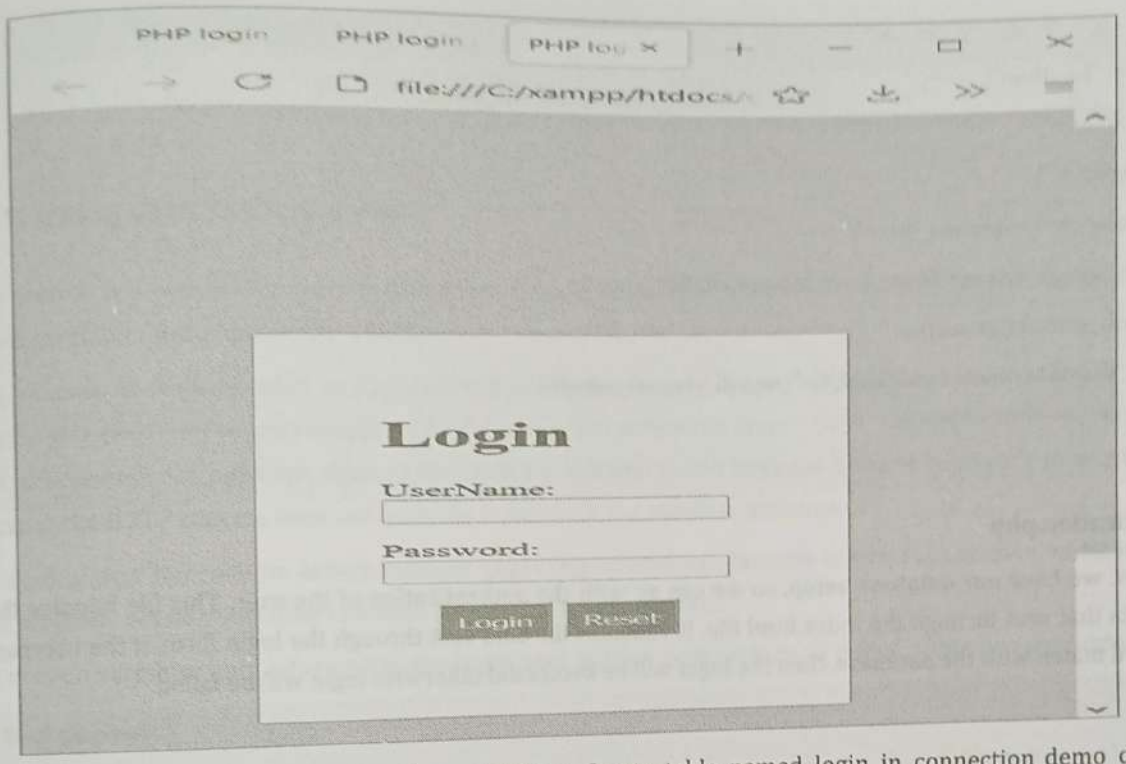
```
<html>
<head>
  <title>PHP login system</title>
  // insert style.css file inside index.html
  <link rel = "stylesheet" type = "text/css" href = "style.css">
</head>
<body>
  <div id = "frm">
    <h1>Login</h1>
    <form name="f1" action = "authentication.php"  method = "POST">
      <p>
        <label> UserName: </label>
        <input type = "text" id ="user" name  = "user" />
      </p>
      <p>
        <label> Password: </label>
        <input type = "password" id ="pass" name  = "pass" />
      </p>
      <p>
        <table >
          <tr align="right" cell-padding:"25%" cell-spacing="25%">
            <td align="right">
              <input type =  "submit" id = "btn" value = "Login" onclick="return validation()"/>
            </td>
            <td align="right">
              <input type =  "submit" id = "btn" value = "Reset"  onclick="reset()"/>
            </td>
          <tr>
        </table>
      </p>
    </form>
  </div>
  // validation for empty field
  <script>
    function reset()
     {
      document.getElementById("user").value=" ";
      document.getElementById("pass").value=" ";
      }
      function validation()
      {
```

```
        var id=document.f1.user.value;
        var ps=document.f1.pass.value;
        if(id.length==="" && ps.length==="") {
           alert("User Name and Password fields are empty");
           return false;
        }
        else
        {
           if(id.length==="") {
              alert("User Name is empty");
              return false;
           }
           if (ps.length==="") {
           alert("Password field is empty");
           return false;
           }
        }
     }
  </script>
</body>
</html>
```

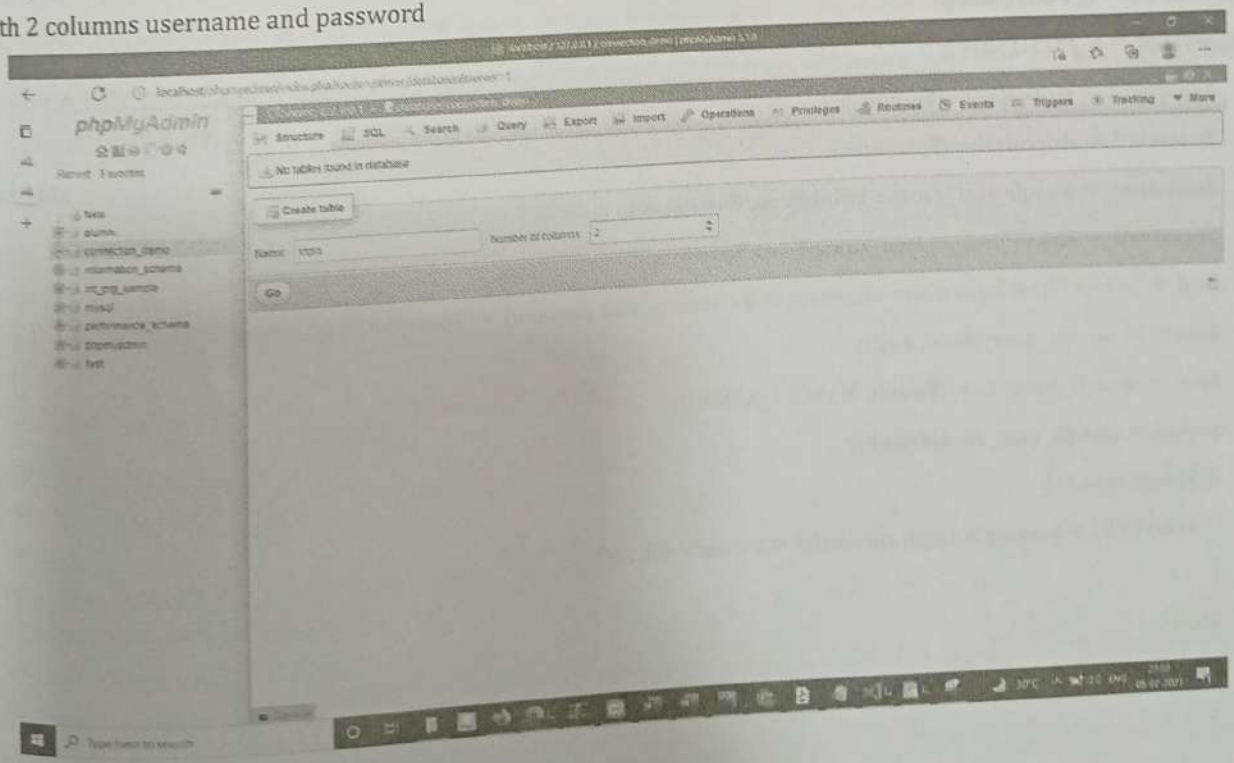
Code For Style.CSS

```
body {
background: Pink;
}
#frm {
border: solid gray 1px;
width:35%;
border-radius: 2px;
margin: 200px auto;
background: white;
padding: 50px;
}
#btn {
color: #fff;
background: #337ab7;
padding: 7px;
margin-left: 40%;
align:center;
}
```

Output



Create database named connection_demo database .Create table named login in connection_demo database with 2 columns username and password



connection.php

Next step is to do the connectivity of login form with the database which we have created in the previous steps.

We will create connection file for which code is given below :

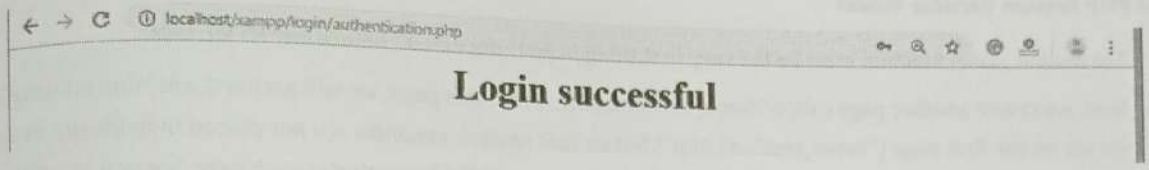
```
<?php
  $host = "localhost";
  $user = "root";
  $password = '';
  $db_name = "connection_demo";
  $con = mysqli_connect($host, $user, $password, $db_name);
  if(mysqli_connect_errno()) {
     die("Failed to connect with MySQL: ". mysqli_connect_error());
  }
?>
```

authentication.php

Now, we have our database setup, so we can go with the authentication of the user. This file handles the login form data that sent through the index.html file. It validates the data sent through the login form, if the username and password match with the database, then the login will be successful otherwise login will be failed.

```
<?php
  include('connection.php');
  $username = $_POST['user'];
  $password = $_POST['pass'];
     $username = stripcslashes($username);
     $password = stripcslashes($password);
     $username = mysqli_real_escape_string($con, $username);
     $password = mysqli_real_escape_string($con, $password);
     $sql = "select *from login where username = '$username' and password = '$password'";
     $result = mysqli_query($con, $sql);
     $row = mysqli_fetch_array($result, MYSQLI_ASSOC);
     $count = mysqli_num_rows($result);
     if($count == 1){
        echo "<h1><center> Login successful </center></h1>";
     }
     else{
        echo "<h1> Login failed. Invalid username or password.</h1>";
     }
?>
```

Output



5.3.9 Tracking user: Session in PHP

- The session is a way to store information (in variables) that can be used across multiple pages. The information is not saved on the computer like a cookie.
- The process of working with an application involves opening it, making changes, and then closing it. This is similar to a session. You are recognized by the computer. It knows when you start the application and when you end. On the internet, however, there is one problem: the web-server does not know who you are or what you do because the HTTP address does not maintain a state as it is a stateless protocol.
- By storing user information across multiple pages (e.g. usernames, favorite colors, etc.) session variables solve this problem. Session variables last until the user closes the browser.
- So, session variables hold information about one user, and are accessible to all pages within one application.

Start a PHP Session

- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.
- Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

Get PHP Session Variable Values

- The session_start() function must be the very first thing in your document. Before any HTML tags.
- Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).
- Also notice that all session variable values are stored in the global $_SESSION variable:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Modify a PHP Session Variable

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
```

To change a session variable, just overwrite it:Example

Destroy a PHP Session

To remove all globa' session variables and destroy thesession,use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
</body>
</html>
```

Review Questions

Q. 1 Explain XML and XSL with example

Q. 2 Distinguish between XML & HTML

Q. 3 What is XML? Explain how to write an XML document? What are the goals of XML? Clearly explain the XML Schema and XML parsing in detail.

Q. 4 Explain Built in functions used in PHP

Q. 5 Create a DTD for a catalogue of cars, where each car element has the childelements make, model, year, colour, engine and doors. The engine elementhas the child elements no_of_cylinders and fuel_type.

Q. 6 What is the purpose of the implicit arrays $_POST and $_GET in PHP? Considerthat a web page displays a form containing two text boxes (named num1 andnum2), where the user enters numeric data. Write a PHP script which collectsthis form data, finds the sum, difference and the product of the two numbers andthen displays the same with suitable messages. Assume that the script is to beembedded in the web page specified by the action attribute of the form and thatthe method used when the form is submitted is GET.

Q. 7 Write a code for insertion of Record in the Table using PHP script

Q. 8 What are the different ways to create an array in PHP? Explain with example?

❑❑❑

6.1 INTRODUCTION

6.1.1 What is ReactJS?

Declarative, Component-Based, and Reusable

React has three main characteristics :

- *It uses declarative views.*
- *It is built on components that manage their own states.*
- *Its code can be reused for different browsers and technologies.*
- ReactJS is a JavaScript library for creating user interfaces that is declarative, fast, and adaptable. It is an open-source, component-based front-end library which is responsible only for the view layer of the application.
- It was created by Jordan Walke, who was a software engineer at Facebook.
- It was initially developed and maintained by Facebook and was later used in its products like Whats App & Instagram.
- A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code.
- The components are the heart of all React applications.These Components can be nested with other components to allow complex applications to be built of simple building blocks.

6.1.2 React Features

1. **JSX :** It is a JavaScript syntax extension. JSX stands for JavaScript XML. Although it is not necessary to use JSX, it is recommended to use it.
2. **Components :** Components are the core of React. They are essential when you are handling the code for large-scale projects.
3. **One-way data-binding :** React applies a one-way data-binding or a unidirectional data flow. The advantage of this is that it gives the user better control over the application. Flux ensures the unidirectional flow of data. It increases efficiency by improving the application's flexibility.
4. **Simplicity :** The JSX file in ReactJS simplifies the application.
5. **Performer :** React's performance is excellent. The main reason behind it is that it uses a virtual DOM, unlike the other frameworks.
6. **License :** ReactJS is licensed under Facebook Inc.
7. **Virtual DOM :** React creates a virtual DOM, i.e., in-memory data -structure cache. Only the final changes of DOM have later updated in the browsers DOM.
8. **JavaScript Expressions :** JS expressions can be used in the jsx files using curly brackets, for example { }.

6.1.3 Advantages of ReactJS

- It is user-friendly. React provides its users with helpful documentation, training resources, and in-depth tutorials.
- Reactjs is an open-source javascript library, so it is easy to start with.
- You can quickly build dynamic web applications with ReactJS as it offers high-functionality. It uses the JSX (JavaScript Extension) that allows you to create an app with minimal coding.
- React's components are reusable, which makes it easier to develop and maintain your apps.
- React provides its developers several handy tools.
- ReactJS uses virtual DOM that makes use of in-memory data-structure cache, and only the final changes are updated in browsers DOM. This makes the app faster.
- You can use React on the client, server-side, and other frameworks as well.
- The data and component patterns in React optimize readability, which is useful when handling larger apps.
- You can create components of your choice by using the react component feature. The components can be reused and also helpful in code maintenance.
- ReactJS has become very popular in a short span and maintained by Facebook and Instagram. It is used by many famous companies like Apple, Netflix, etc.
- ReactJS can be used to develop rich UI for both desktop and mobile apps.
- Easy to debug and test as most of the coding is done in JavaScript rather than on Html.

6.2 REACTJS INSTALLATION

6.2.1 Installation

Installing Node and npm

In order to setup ReactJS development environment, you will need NodeJS and NPM installed.

- **Step 1 :** Download the Node.js from the official Node website: https://nodejs.org/en/
- **Step 2 :** Run the installer (the .msi file you downloaded in the previous step.)
- **Step 3 :** Follow the installation wizard to install Node.js and npm.
- **Step 4 :** Restart your computer. You will not be able to run Node.js until you start your computer again.

Setting Up a ReactJS Project

- **Step1 :** Install React using the npm package manager.

npm install -g create-react-app

- **Step2 :** Create a new React project using the create-react-app command.

create-react-app reactproject

We can combine the above two steps in a single command using npx. The npx is a package runner tool which comes with npm 5.2 and above version.

npx create-react-app reactproject

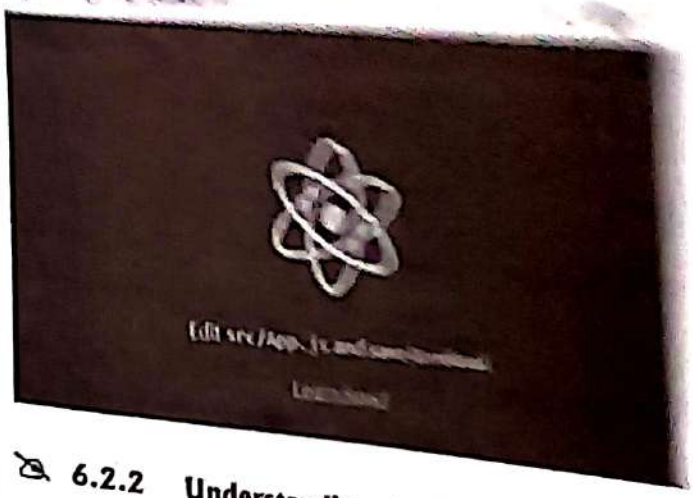
- **Step 3 :** After you complete the installation process, you can now start the server.

npm start

- **Step 4 :** Now open the following URL in browser to view the output:

  http://localhost:3000

- **Step 5 :** You will see the following screen.



6.2.2 Understanding the React Project Structure

Open the test-project folder using your preferred editor, and you will see the project structure as shown below :

Files and Folders	Description
node_modules	In this folder NPM will save all dependent files and libraries required for our project. When we install new packages, those packages will get stored in in the node_modules folder.
public	This folder contains the static parts of our application.
public/favicon.ico	This is the icon shown in the browser's address bar and is used for bookmarks. By default the favicon.ico is the React logo, which we need to replace with our own favicon.ico file.
public/index.html	This is the HTML file that is loaded by the browser. It contains an element in which the application is displayed and a script element that loads the application's JavaScript files.
public/manifest.json	This provides meta information about our site app. It is generally used for devices to give an Add to Home Screen option that will load an icon for our site to our device's home screen. It is required to modify the short_name, name, theme_color and background_color to suite our project needs.
src	The src folder is the most prominent in the project because it is where the application's code and content files are placed and where you will desing and develop new features required for our project.
src/App.css	This file contains the placeholder CSS styles for new projects.
src/index.css	This file contains the global CSS styles for the application.
src/App.js	Contains the definition of the main component of the sample application.
src/index.js	This is our entry point for our app and where Create React App will start looking to for other files to import and bundle. It also loads any high level functionality like service workers.
src/App.test.js	Stores the basic unit test involving the App component.
src/serviceWorker.js	They are a Web API that leverages caches to allow for sites to work offline or faster on slow connections. This file is used by progressive web applications, which can work offline.
src/logo.svg	This image file contains the React logo.
.gitignore	This file is used to exclude files and folders from the Git revision control package.
package.json	The initial set of packages is defined in the package.json file using the dependencies and devDependencies properties. The dependencies section is used to list the packages that the application will require to run. This file contain NPM scripts we can run with our app. We need to modify the name or version of our app here.
README.md	This document consists of notes to all you need to start building a React-based application. You need to organize or overwrite it with your own documentation.

6.3 REACTJS HELLO WORLD APPLICATION

Creating ReactJS Hello World Component

In this example we will create a component that will print "Hello World!" message on browser.

src/App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
 render() {
  return (
<div className="App">
<h1>Hello Wolrd!</h1>
</div>
  );
 }
}
export default App;
```

Here, the first statement is used to include the react module, next we are creating a "App" Component by creating a "App" class that extends the Component class.

Inside of "App" class we have created a render method which is a part of React.Component class, it return a "Hello World!" message enclosed in <h1></h1> tag.

Next, we have following statement -

export default App;

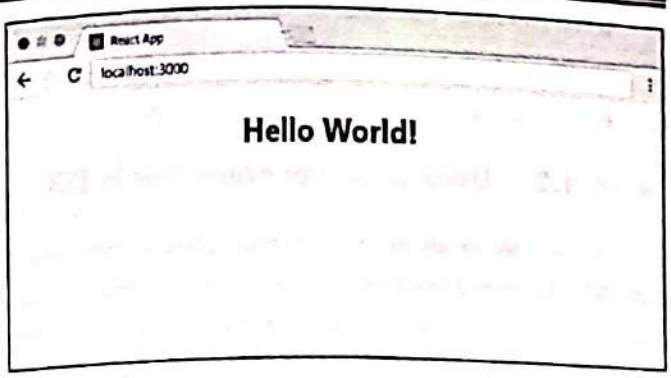
Here, we are App component to be mounted to a DOM Element that we have available in public/index.html file

To compile the final file run following command:

npm run start

When you run above command, it will compile the files and notify you if any error, if all looks good it will open the browser and the run the index.html file at http://localhost:3000/index.html.

The URL http://localhost:3000 will open in the browser once the code is compiled and display a message "Hello World!".



6.4 INTRODUCTION TO JSX

What is JSX?

- JSX stands for JavaScript XML.
- JSX is a language that allows you to mix JavaScript and HTML-like tags to define user interface (UI) elements and their functionality.
- JSX is an extension for JavaScript that allows for writing what looks like HTML and XML in our JavaScript.
- JSX is a shorthand for calling React.createElement() to create elements and components.
- JSX looks like HTML and offers a friendly syntax for creating DOM elements.
- JSX accepts any valid JavaScript expression inside curly braces.

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. But instead of using regular JavaScript, React code should be written in something called JSX.

Example

const ele = <h1>This is sample JSX</h1>;

- The above code snippet somewhat looks like HTML and it also uses a JavaScript like variable but is neither HTML nor JavaScript, it is JSX.
- JSX is basically a syntax extension of regular JavaScript and is used to create React elements. These elements are then rendered to the React DOM.

6.4.1 Why use JSX?

- It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript.
- Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both.

Module 6

- It is type-safe, and most of the errors can be found at compilation time.
- It makes easier to create templates.

6.4.2 Using JavaScript expressions in JSX

In React we are allowed to use normal JavaScript expressions with JSX. To embed any JavaScript expression in a piece of code written in JSX we will have to wrap that expression in curly braces { }. Consider the below program, written in the index.js file:

```
import React from 'react';
import ReactDOM from 'react-dom';

const name = "Students";

const element = <h1>Hello,
{ name }.Welcome to React.< /h1>;

ReactDOM.render(
  element,
  document.getElementById("root")
);
```

Output

```
Hello, Students Welcome to React.
```

In the above program we have embedded the javascript expression const name = "Students"; in our JSX code.

We embed the use of any JavaScript expression in JSX by wrapping them in curly braces except if-else statements. But we can use conditional statements instead of if-else statements in JSX. Below is the example where conditional expressing is embedded in JSX :

```
import React from 'react';
importReactDOM from 'react-dom';

leti = 1;

const element = <h1>{ (i == 1) ? 'Hello World!': 'False!'}
< /h1>;

ReactDOM.render(
  element,
  document.getElementById("root")
);
```

Output

```
Hello World!
```

In the above example, the variable i is checked if for the value 1. As it equals 1 so the string 'Hello World!' is returned to the JSX code. If we modify the value of the variable i then the string 'False' will be returned.

6.4.3 JSX Attributes

- JSX use attributes with the HTML elements same as regular HTML.
- JSX uses camelcase naming convention for attributes rather than standard naming convention of HTML such as a class in HTML becomes className in JSX because the class is the reserved keyword in JavaScript.
- We can also use our own custom attributes in JSX.
- For custom attributes, we need to use data- prefix.

In the below example, we have used a custom attribute data-demoAttribute as an attribute for the<p> tag.

Example

```
import React, { Component } from 'react';
class App extends Component{
render(){
return(
<div>
<h1>Hello</h1>
<h2>ReactJs</h2>
<p data-demoAttribute = "demo">Welcome to
ReactJS!!</p>
</div>
    );
  }
}
export default App;
```

In JSX, we can specify attribute values in two ways:

1. As String Literals: We can specify the values of attributes in double quotes:

```
var element = <h2 className = "firstAttribute">ReactJS
</h2>;
```

Example

```
import React, { Component } from 'react';
class App extends Component{
render(){
return(
<div>
<h1 className = "hello" >ReactJS</h1>
<p data-demoAttribute = "demo">This website contains the best ReactJS tutorials.</p>
</div>
    );
  }
}
export default App;
```

Output

```
ReactJS
This website contains the best ReactJS tutorials.
```

2. **As Expressions :** We can specify the values of attributes as expressions using curly braces { }:

```
var element = <h2 className = {varName}>Hello ReactJS</h2>;
```

Example

```
import React, { Component } from 'react';
class App extends Component{
render(){
return(
<div>
<h1 className = "hello" >{25+20}</h1>
</div>
    );
  }
}
export default App;
```

Output

```
45
```

6.4.4 JSX Comments

JSX allows us to use comments that begin with /* and ends with */ and wrapping them in curly braces { } just like in the case of JSX expressions.

Example

```
import React, { Component } from 'react';
class App extends Component{
render(){
return(
<div>
<h1 className = "hello" >Hello ReactJS</h1>
    {/* This is a comment in JSX */}
</div>
    );
  }
}
export default App;
```

6.4.5 JSX Styling

React always recommends to use inline styles. To set inline styles, you need to use camelCase syntax. React automatically allows appending px after the number value on specific elements.

Example

```
import React, { Component } from 'react';
class App extends Component{
render(){
varmyStyle = {
fontSize: 80,
fontFamily: 'Courier',
color: '#003300'
    }
return (
<div>
<h1 style = {myStyle}>www.sample.com</h1>
</div>
    );
  }
}
export default App;
```

Module 6

6.5 SIMPLE APPLICATION USING JSX

6.5.1 Create React Element using JSX

Open a new command prompt, navigate to the folder where you want to create your React project and run the following command:

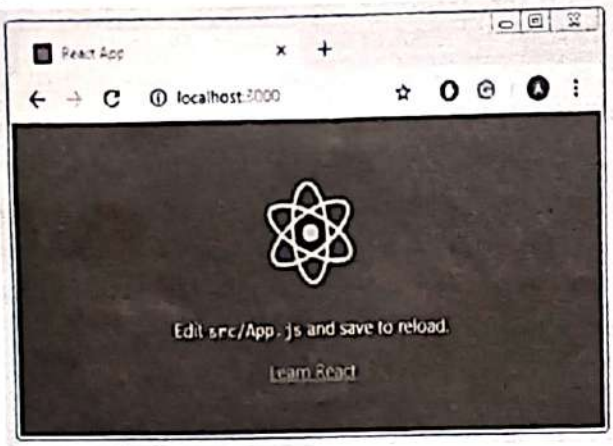
create-react-app example1

When the folder is created, navigate to it by typing:

cd example1

To run the app, type command: **npm start**

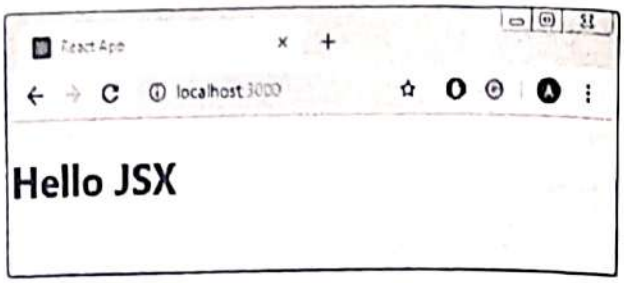
The above command launches the server, earlier the initial preparation for the project is finished, a new browser window will visible and display the URL **http://localhost:3000** and publish the placeholder content as shown below:



Now replace the placeholder content in the App.js file with a component who returns some simple HTML elements.

```
            src\App.js
import React, { Component } from "react";

class App extends Component {
    render(){
        return (
            <h1>Hello JSX</h1>
        );
    }
}
export default App;
```

In the render method, we specify the JSX that will be inserted into the DOM as HTML when the component's view is rendered.



6.5.2 Using JSX in Return Statement of Component

Let's create a Vehicles component which returns valid JSX statement.

```
        src\Vehicles.js
import React from "react";
export default function Vehicles() {
    return(
        <h1>
            Car
        </h1>
    );
}
```

The above example shows both opening and closing paired tags in action.

It is suggested to write HTML tags always in lowercase this helps to distinguishes them from custom JSX tags.

Now update App.js, It is the view component that controls our entire app or page.

```
src\App.js
import React, { Component } from "react";
import Vehicles from "./Vehicles";
class App extends Component {
    render(){
        return (
            <Vehicles />
        );
    }
}
export default App;
```

When we use a function or class to create our components, we will always name them using capitalize and call them using the JSX syntax like below:

```
<Vehicles />
```

Important Questions

Q. 1 What is ReactJS?

Q. 2 What are the features of React?

Q. 3 What are the advantages of using React?

Q. 4 List out the installation steps of react.

Q. 5 Create a Hello World application in ReactJS.

Q. 6 What is JSX?

Q. 7 Explain JSX Attributes with an example.

6.6 MULTIPLE CHOICE QUESTIONS

Q. 6.1 React is a _____

(a) Javascript library (b) Javascript framework

(c) Both of above (d) None of above

✓ **Ans. : (a)**

Q. 6.2 ReactJS covers _____

(a) User Interface layer in an application

(b) Data layer in an application

(c) Both a and b

(d) None of above

✓ **Ans. : (a)**

Q. 6.3 ReactJS uses _____ to increase performance.

(a) Virtual DOM (b) Original DOM

(c) Both of above (d) None of above ✓ **Ans. : (a)**

Q. 6.4 React is _____

(a) Component based (b) Module based

(c) Both of above (d) None of above

✓ **Ans. : (a)**

Q. 6.5 Consider following example :

```
var Helloword=(props)=>
{
return(
<div>
hello world 1
</div>
<div>
Hello world 2
</div>
)
}
ReactDOM.render(<Helloworld/>,mountNode)
```

Q. 6.6 What will be the output:

(a) Hello world 1

(b) Hello world 2

(c) Hello world 1 Hello world 2

(d) Error

✓ **Ans. : (d)**

Q. 6.7 Who is the father of React.js?

(a) Jordan mike (b) Jordan Walke

(c) Nile (d) Nike John ✓ **Ans. : (b)**

Q. 6.8 Who develop react.js?

(a) Apple (b) Facebook

(c) Twitter (d) Google ✓ **Ans. : (b)**

Q. 6.9 Which of the following is true about React?

(a) React is a JavaScript library for building user interfaces.

(b) React is used to build single page applications.

(c) React allows us to create reusable UI components.

(d) All of the above ✓ **Ans. : (d)**

Q. 6.10 JSX allows us to write HTML elements in JavaScript and place them in the DOM without any?

(a) createElement

(b) appendChild

(c) getElementById

(d) Both A and B ✓ **Ans. : (d)**

Q. 6.11 Which command is used Install create-react-app?

(a) npm install -g create-react-app

(b) npm install create-react-app

(c) npm install -f create-react-app

(d) install -g create-react-app ✓ **Ans. : (a)**

Q. 6.12 React renders HTML to the web page by using a function called?

(a) ReactDOM.render()

(b) render()

(c) ReactDOM_render()

(d) React.render() ✓ **Ans. : (a)**

Q. 6.13 With JSX you can write expressions inside?

(a) () (b) []

(c) " " (d) {} ✓ **Ans. : (d)**