UNIT I

CHAPTER 1

Web Essentials & Mark-up Language-HTML

Syllabus

The Internet, basic internet protocols, the World Wide Web, HTTP Request message, HTTP response message, web clients, web servers. **HTML :** Introduction, history and versions.**HTML elements :** headings, paragraphs, line break, colors and fonts, links, frames, lists, tables, images and forms, Difference between HTML and HTML5. **CSS :** Introduction to Style Sheet, CSS features, CSS core syntax, Style sheets and HTML, Style rule cascading and inheritance, text properties. Bootstrap.

HTTP - Hypertext Transfer Protocol
HTML - Hypertext Markup language.

1.1 INTRODUCTION TO WEB TECHNOLOGY

1.1.1 Internet and WWW

1.1.1(A) Internet

GQ. Explain the term Internet. **(2 Marks)**

- Now a days, we observe that internet is everywhere. There may be hardly any mobile used by young generation which does not have internet service. Number of TV shows and magazines are devoted to internet.
- Internet becomes a basic need of people. If for some time internet gets disconnected, people really get very disturbed and nervous just like some big problem get arise. Up till now you have used internet for various purposes, but now in this subject we are going to learn how to use this internet to become a web developer.

❒ **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.

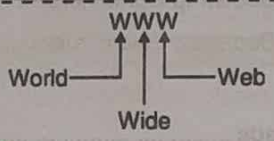
- The Internet is linked by a large array of wireless, electronic and optical technologies of networking.
- The Internet provides a broad range of information resources and services like the World Wide Web (WWW) applications, telephony, email , sharing of files and connected hypertext documents.
- In 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 book and newspaper. 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.

☞ **Uses of Internet**

(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

1.1.1(B) WWW

GQ. Explain the term WWW. **(2 Marks)**



❒ **Definition :** 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 which we will learn in HTML.
- 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.

1.1.2 Web Technology

1.1.2(A) Website

GQ. Explain the term Website. (2 Marks)

Definition : A website is made of number of web pages. Simply a website is a file which is accessible anywhere in the world through internet.

- Websites are usually created for commercial purpose. As well 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 some extensions depending upon the countries:

  .in – India

  .us – America

  .uk – United Kingdome

  .nz – New Zealand

- Now a day, Web development can be definitely considered as a happening career option. So if you think to become a web developer, then first you should understand the entire technology of web.
- In any website there are basically two important aspects – Web Designing and web development (scripting).

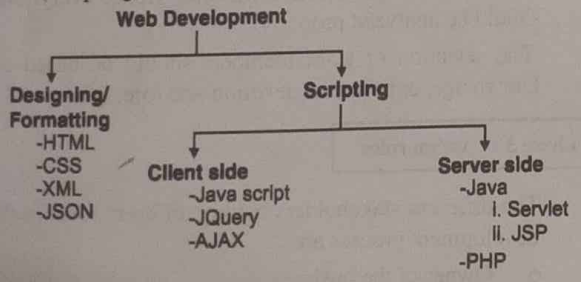


Fig. 1.1.1 : Web development

1.1.2(B) Web Designing

- As the basic intention of website is always commercial, it is necessary that the website should be user friendly and attractive which is completely depending upon the design of website.
- For designing purpose various Scripting languages like HTML, CSS and Software or tools like Photoshop, CorelDraw, Flash etc are used.

1.1.2(C) Web Development

- While developing software we write programs using programming languages. Such programs are also written in web development. These programs are known as scripts and the languages in which we write such scripts are known as Scripting Languages.
- Basically there are two types of scripts. To understand this we will take an example of email account form. To create an email account, we have to fill the email account form in which we have to give the username, password and some personal details.
- This email account form which contains different controls like textbox, checkbox, option buttons, submit buttons etc is designed using languages like **HTML, CSS** etc.
- Customized <tags> creation and data transfer is done by the languages like **XML, JSON** etc.
- You may remember that while filling such form it may be possible that we left some fields empty or may give some incorrect data. In such case the website gives error messages and forces us to fill or correct the data. It means that the website has written the code (script) to check the data given by us. It is known as validation checking.
- This script directly interacts with the client (website users) hence called as client side script. The language in which such script is written is known as Client Side Scripting Language. There are number of Client Side Scripting Languages like **JavaScript, JQuery, AJAX etc.**
- Now consider that you have given all the data correctly on email account form and submitted it. Now where this data goes? This data is stored in the databases like **Oracle, SQL-Server or MySQL** on the server.
- Now to store such data in the database, the website has to write the script to interact with database which is always there on the server.
- Hence such script is known as Server Side Script. The languages in which this script is written are known as Server Side Scripting languages.
- These languages are **Java(Servlet and JSP), PHP** etc.

1.2 WEBSITE PLANNING AND DESIGN ISSUES

1.2.1 Website Planning

UQ. Describe phases of web site development in brief. SPPU - Aug. 2014 - In Sem, 6 Marks

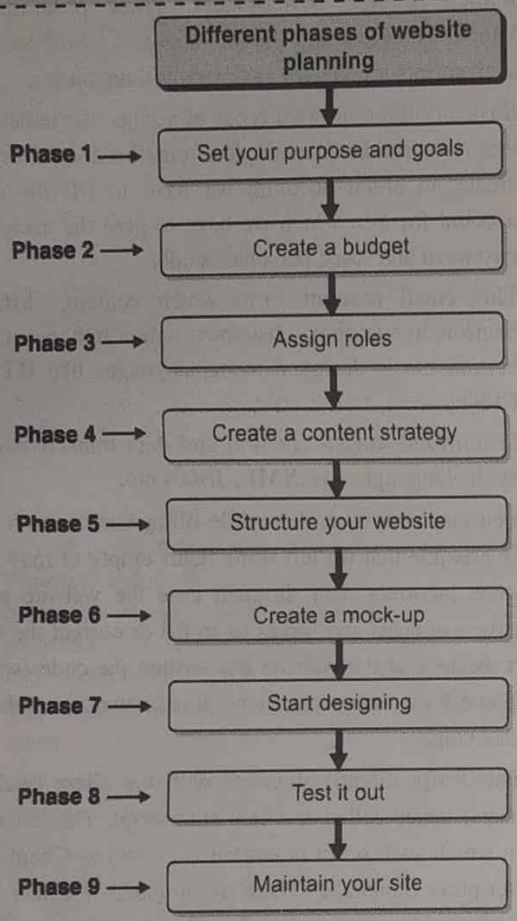


Fig. 1.2.1 : Different phases of website planning

- *"If you fail to plan, you plan to fail"*, this is an old adage which seem to be perfectly true. If we build any website without proper planning, then it will be same as of constructing a building without any blueprint.
- The lack of planning may lead to the things ends with wrong place, expected features may get missed and miscommunication between the web developer and clients may occur.
- The proper planning of website in time will definitely give a perfect direction as well as avoids missed deadlines and backtracking.
- The **different phases** of website planning :

Phase 1: Set your purpose and goals

- The purpose of website is the most important factor. There are various purposes like :
  Gain publicity for the business
  Sell the inventory
  Get support behind a particular cause
- It is necessary to identify purpose of website and the target audience. The goals must be refined. Some basic questions we have to revise like:
  - How many visitors we expect in a month for our website?
  - How many visitors should sign in for our news later?
  - How much sale we expect?
- We have to set specific, measurable goals for our website which should match with the marketing goals. There are some analytical tools available in market which can monitor the performance of our website over time.

Phase 2 : Create a budget

- Whatever the extent of our business, it may be mid-sized organization or a start up, it is very important to set a proper budget of our website expenses. The expenses may include the costs for web design, scripting and web hosting. Some more expenses may get added later. For this purpose we have to conduct a proper market survey as well as have some discussions with professionals.
- Sometimes avoiding expenses for some important things to decrease the cost may lead to big headaches or creation of non effective website. Hence everything should be analyzed properly.
- The selection of team members should be based on knowledge, experience, devotion and foresight.

Phase 3 : Assign roles

- The different stakeholders having different roles in the development process are :
  - Owner of the business
  - Marketing Manager
  - Web and graphic designer
  - HTML/CSS professional
  - Web developer
  - Content writer and/or editor
  - Database administrator

- We have to make sure that every person in the team is well aware of his/her role and expectations from him/her. They should also aware with the deadlines and new developments.

Phase 4 : Create a content strategy

- Content is nothing but any text on website which gives some information to visitors.
- The content of a website may include :
  - Blog posts
  - Pictures
  - Video
  - Slideshows
  - Embedded social media feeds like the Twitter stream or updates of Facebook page
  - Documents
- The content strategy helps to present the content in time. Consider we want to publish one blog post every week then we have to hire a professional web writer having experience of content writing.

Phase 5 : Structure your website

- It is important to decide the exact features of all the web pages.
- In general all the websites have "About" and "Contact Us" pages, but we have to decide the features of web pages in such a way that they should meet our business needs.

Phase 6 : Create a mock-up

- The mock-up which is also known as a wireframe is outline of our website. As it is the initial design, can be considered as first draft.
- Usually it is created in Photoshop or Fireworks. There is no need of putting in depth details in the mock-up. Placeholder text is used to fill the pages. This is just an idea for everyone that how the website will look like.
- If initially the design software like Photoshop is not available, then it can also be simply designed by using pen and paper.

Phase 7 : Start designing

Designing is the major aspect in web development. The website with good design will certainly attract the visitors.

The navigation (moving around the different web pages) of website should be user friendly.

While designing the website, we have to take care of some important issues :

- The navigation of website should be easy to find and understand. Mostly the users expect it vertical and centered at the top of web page.
- The font of the text should be easy to read rather than stylish. The text color and background color should be contrast.
- The size of web pages should fit in the screen. Responsive design is best option which makes the website fit for all screen sizes or resolutions.
- The website should be light-weight so that it can be loaded quickly.
- The logo and tag line of company should be prominent on the page.
- Consistency of style and colors should be maintained across the website.
- The important information and features must be located at prominent locations.

Phase 8 : Test it out

- Testing is necessary for finding out errors and retrieving missing details that we might have forgotten initially. Test the compatibility of website for various browsers like Chrome, Firefox, Internet Explorer, and mobile compatible browsers like Safari and Opera Mini. Test the website on mobile phones and tablets.
- The appearance of the website should be consistent.
- Make sure all the links work properly, images are in proper size.
- Make sure that all the forms and input fields are working properly.

Phase 9 : Maintain your site

- As the website is uploaded, it does not mean that everything is finished. Now the work of maintenance get starts as the website is ongoing entity which continuously represents the company.
- Monitor the behavior of website with end users with the help of analytics software.
- We have to keep an eye on important aspects like the number of unique visitors and which pages are frequently visited on the website.
- The maintenance work also includes the posting new contents, monitoring security etc. Getting feedback from end users is also very important as it helps us to improve the website.

1.2.2 Design Issues

UQ. List and explain different design issues in web design. **SPPU - Aug. 2015 (In sem), 4 Marks**

There are number of major technical issues which should be considered while developing a website.

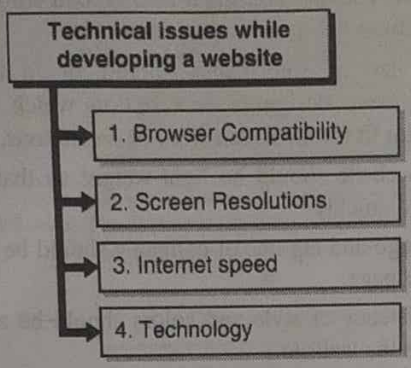


Fig. 1.2.2 : Technical issues while developing a website

1. Browser Compatibility

- The webpage should be properly displayed on different browsers like Chrome, Firefox, Internet Explorer, and mobile compatible browsers like Safari and Opera Mini.
- Hence while building the website, the web pages should be checked on all possible browsers and operating systems for the compatibility. It is important to test the pages on latest as well as older versions of browsers, because not all of the site visitors may have latest versions.

2. Screen Resolution

- Usually the common resolution of screen is 1024 x 768 pixels, but now there is growing trend towards higher resolutions. If the website is designed for higher resolution, it may be possible that some low -resolution screens may not be able to display all of the contents.
- Also it is very important to take care of mobile users. It is necessary to check the compatibility of our website for the mobiles, means how it look and feel on the mobile screens.

3. Internet speed

- It is not possible that all the users have high speed internet. The download speed may get affected by the webpage design.
- The study suggests that: Near about 50 percent of web users expect that the webpage should be loaded in 2 seconds or even less. At least 40 percent visitors abandon a website which requires more than 3 seconds to load.
- Use of more images or heavy media like animations or videos slow down the downloading process of web pages.
- This can result in a fact that customers leaving the site. The speed affects the ranking of our website in search engine. It indicates that we should try to keep the file and image sizes as minimum as possible. The total size of a webpage is expected not more than 40 to 60 kilobytes.

4. Technology

Some web technologies may also prevent end users from visiting our website or affect indexing of the website in search engines. These technologies include :

- HTML frames
- JavaScript
- Flash
- AJAX

These technologies have some potential risks to the usability and accessibility of our website.

1.3 INTRODUCTION TO BASIC INTERNET PROTOCOL

UQ. What do you mean by protocol? Explain the working of those protocols. **SPPU - Aug. 14 In-sem, 4 Marks**

UQ. Explain in detail HTTP protocol, purpose and operation. **SPPU - May 15, 5 Marks**

- **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.

❐ **Definition :** The Text which includes hyperlinks is known as Hypertext. Hyperlinks are the special text which links other nodes.

- It is an application-level stateless protocol which is used by information systems which are distributed, collaborative, and contain hypermedia.
- 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 Basic Features of HTTP protocol

Following are the 3 features of HTTP which helps HTTP protocol to become a simple, human readable & powerful protocol :

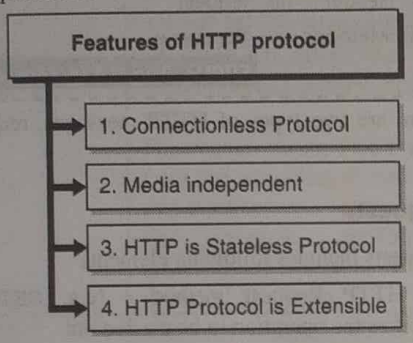


Fig. 1.3.1 : Features of HTTP protocol

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.
- Means that, the HTTP server and HTTP client are known to each other during a current HTTP request only and 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.

Where HTTP situated?

- Fig. 1.3.2 illustrates the architecture of a web application which indicates that where the HTTP protocol is situated :

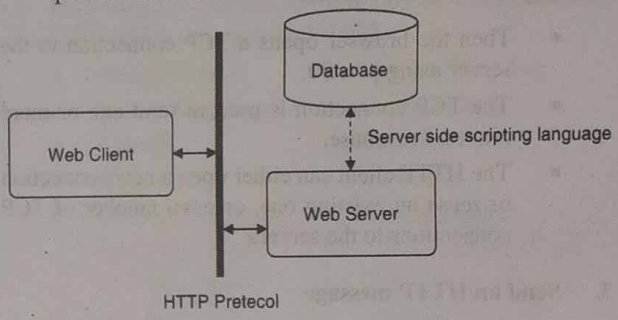


Fig. 1.3.2 : Architecture of web application

1. HTTP protocol

HTTP is also said to be the request/response protocol which depends on client-server architecture where the HTTP clients are : Web browsers and the HTTP Servers are : Web Servers.

2. Web Client

The HTTP client that is web browser sends a HTTP request to the HTTP server. The format of a HTTP request is as follows :

(i) An HTTP request method,

(ii) URI,

(iii) HTTP Protocol version,

(iv) A message as MIME including HTTP request modifiers, information about HTTP Client, and the body content about a TCP/IP connection.

3. Web Server

The HTTP server usually the Web server sends back the response. The format of a HTTP response is as follows :

(i) A status line
(ii) HTTP Protocol version of message
(iii) A status code i.e. success or error code,
(iv) A message as MIME including information about entity meta-information, and entity-body content.

☞ **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.
- e.g. following statement is used to request a web page index.html from www.PhoenixGlobe.com .
- GET www.PhoenixGlobe.com/index.html HTTP/1.0
- Before the introduction of HTTP 2, HTTP messages are human-readable.
- In HTTP 2, they are encapsulated in frames, to restrict them from direct reading, but the principle remains the same.

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.
- E.g. 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.

1.3.2 HTTP Messages

UQ. Define Following terms related to HTTP :
(i) Header (ii) Request
(iii) Methods (iv) Response.
SPPU - Aug. 14 In-sem, 4 Marks

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

1. Requests

- 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. Similar to responses.
- The first three elements together referred as request line. Let's see more information about 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 :

☞ **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.

- The Table 1.3.1 lists supported by HTTP

Sr. No.	Method
1.	GET
2.	HEAD
3.	POST
4.	PUT
5.	DELETE
6.	CONNECT
7.	OPTIONS
8	TRACE

☞ **Request-URI**

- The URI is short fo
The Request-URI i
- Following statemen
URI :

"*" | absoluteURI | abs

- Note that the abs
request is made to

3. Web Server

The HTTP server usually the Web server sends back the response. The format of a HTTP response is as follows :

(i) A status line

(ii) HTTP Protocol version of message

(iii) A status code i.e. success or error code,

(iv) A message as MIME including information about entity meta-information, and entity-body content.

☞ 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.
- e.g. following statement is used to request a web page index.html from www.PhoenixGlobe.com .
- GET www.PhoenixGlobe.com/index.html HTTP/1.0
- Before the introduction of HTTP 2, HTTP messages are human-readable.
- In HTTP 2, they are encapsulated in frames, to restrict them from direct reading, but the principle remains the same.

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.
- E.g. 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.

1.3.2 HTTP Messages

UQ. Define Following terms related to HTTP :
(i) Header (ii) Request
(iii) Methods (iv) Response.
SPPU - Aug. 14 In-sem, 4 Marks

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

1. Requests

- 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`. Similar to responses.
- The first three elements together referred as request line. Let's see more information about 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 :

☞ 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.

- The Table 1.3.1 lists all the Request-methods which are supported by HTTP/1.1.

Table 1.3.1

Sr. No.	Method	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.

☞ 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
```

☞ 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. Responses

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. Let's see more information about all the elements :

☞ 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

Let's see details about the parts of Status line :

(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 Table 1.3.2.

Table 1.3.2

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.

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 WEB CLIENTS

A web client is an application that communicates with a web server, using Hypertext Transfer Protocol (HTTP). Hypertext Transfer Protocol is the protocol behind the World Wide Web. With every web transaction, HTTP is invoked.

HTTP is behind every request for a web document or graphic, every click of a hypertext link, and every submission of a form. The Web is about distributing information over the Internet, and HTTP is the protocol used to do so.

1.5 WEB SERVERS

UQ. Explain working of Web Server.

SPPU - Dec. 17, 3 Marks

Definition : A **web server** is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web.

- The term can refer to the entire system, or specifically to the software that accepts and supervises the HTTP requests
- The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP).
- Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.
- A user agent, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so.
- The resource is typically a real file on the server's secondary storage, but this is not necessarily the case and depends on how the web server is implemented.
- 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 ("on-the-fly") 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.
- This usually means that no additional software has to be installed on the client computer, since only a web browser is required (which now is included with most operating systems).

1.5.1 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.

1.5.2 Features of web servers

UQ. What are the general features of web servers?
SPPU - Aug. 15 In-Sem, Dec. 17, 4 Marks

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.

1.5.3 Wamp Server

GQ. Explain Wamp Server in detail. **(4 Marks)**

- WAMP is a Windows OS based program that installs and configures Apache web server, MySQL database server, PHP scripting language, phpMyAdmin (to manage MySQL database), and SQLiteManager (to manage SQLite database).
- WAMP is designed to offer an easy way to install Apache, PHP and MySQL package with an easy to use installation program instead of having to install and configure everything yourself.
- WAMP is so easy as once it is installed it is ready to go. You don't have to do any additional configuring or tweaking of any configuration files to get it running. There are usually two reasons why someone chooses to install WAMP. They are looking to install WAMP for development purposes or to run their own server.

☞ Wamp Server Contains

1. **PHP Admin**
   - Allows you to change or add users and for making new databases phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the World Wide Web. phpMyAdmin supports a wide range of operations with MySQL.
   - The most frequently used operations are supported by the user interface (managing databases, tables, fields, relations, indexes, users, permissions, etc), while you still have the ability to directly execute any SQL statement.
2. **Apache**
   Apache Server deals with Server Side Includes, usually called simply SSI.
3. **SQL Server and Database System**
   - SQL Server is a relational database management system from Microsoft that's designed for the enterprise environment.
   - SQL Server runs on T-SQL (Transact -SQL), a set of programming extensions from Sybase and Microsoft that add several features to standard SQL, including transaction control, exception and error handling, row processing, and declared variables.
4. **Using WAMP as a Development Server**
   - People can use WAMP to develop and test websites locally on their own computer instead of having to get a web hosting account to develop with.
   - Most people are using WAMP for development purposes such as learning how to create websites with HTML, PHP, and MySQL.

1.5.4 Web Service

GQ. What is web service? (2 Marks)

The definition of web service according to W3C is given below :

❒ **Definition : A web service** is a software system designed to support interoperable machine-to-machine interaction over a network.

- The web service also can be defined as any service offered by an electronic device to another electronic device which communicates over the network.
- The web service uses XML messaging system that makes the web service as self describing service and it is available over the Internet.

1.5.5 Need of Web Service

GQ. What is the need for web service? (3 Marks)

The web services are used for following needs :

(a) Interoperability

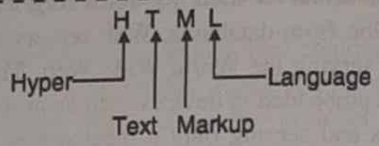
- Different developers can use different technologies for developing the applications so in distributed system there is a need of interoperability.
- The web services offer developers the freedom to work on whatever the technology they want to use for development.
- And also the web services permits to communicate different applications developed in different technologies with each other by data and service sharing between them.
- In other words we can say that, the Web services are used to make the applications platform and technology independent. Web services use set of various standard protocols to achieve interoperability.

(b) Reusability

- It is possible that the some sort of code may require to 10 applications.
- So with the help of web services the code which is necessary to 10 applications can be published on the network as a service so that it can be available to all of those applications.

1.6 HTML

GQ. What is HTML ? (2 Marks)



- **HTML** stands for **Hyper Text Markup Language**. Hypertext is nothing but the way in which the different web pages are linked with each other. Such links are called as Hypertext.
- As its name indicates, HTML is a Markup Language. That means we can use HTML to simply "mark-up" a text document with tags that instructs the Web browser like Chrome how to structure it to display.
- Basically the HTML is developed for the purpose of defining the structure of web document. It includes paragraphs, headings and lists.
- Now a day HTML is widely used for the purpose of formatting web pages by using various tags.

1.6.1 Structure of HTML Document

UQ. Explain the basic structure of HTML document.
SPPU - Oct. 2016 - In sem, 6 Marks

- All the elements are included in the main opening and closing <html> tags.

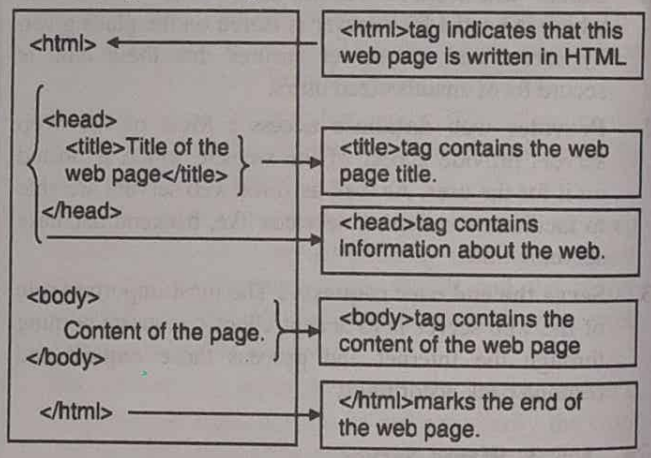


Fig. 1.6.1 : Structure of HTML Documents

- An HTML document contains two main parts:
  1. **Head :** The head element contains title and metadata of a web document. This section is used to declare variables and functions in scripting languages. These variables and functions are then accessible throughout the page.

2. **Body :** The body element contains the information which we like to display on a web page. All text to be displayed and control creation is done in this section.

1.7 HTML ELEMENTS / TAGS

UQ. Give the list with definition of HTML components.
SPPU - Dec. 2014, May 2015, 2.5 Marks, Oct. 2016 - In Sem, 5 Marks

UQ. List various tags in HTML with simple example for a web page. **SPPU - Dec. 2015, 5 Marks**

Definition : HTML tags are standard keywords in the web page which define how the web browser should format and display the content.

- Usually most of the tags have two parts, an opening and a closing part. For example, <html> is the opening tag while </html>is the closing tag.
- We have to remember that that the closing tag has the similar text as of the opening tag, but has an extra forward-slash (/) character.
- There are some tags which do not need closing tag. All the HTML files must have the necessary tags for it to be valid so that the web browser can understand and display it properly.
- **An HTML element** is generally defined with starting tag. If the element has some content then it ends with a closing tag.
- For example, is starting tag of a paragraph and is closing tag of the same paragraph.

  Phoenix InfoTech is a paragraph element.
- Some elements have attributes to which values are assigned to format the webpage.

```
<tag_name attribute=value>content</tag_name>
```

Example

```
<body bgcolor="skyblue">--</body>
```

1.7.1 Headings

GQ. Explain heading element in HTML. **(2 Marks)**

Use :

- 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.

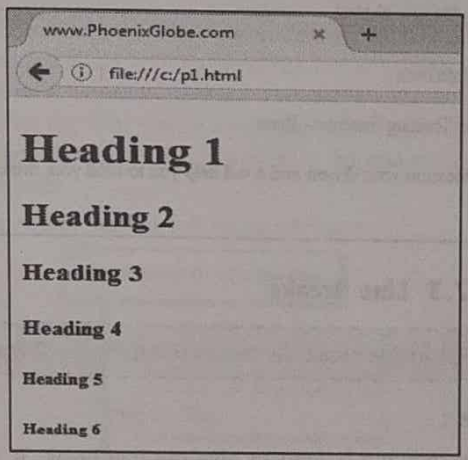
Program 1.7.1 : Write a program to display all heading tags from <h1> to <h6>.

Soln. :

Program to display various headings

```
<!DOCTYPE html>
<html>
 <head>
 <title>www.PhoenixGlobe.com</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



As we can observe that the contents of heading tags are by default bold and sizes are depending upon the particular heading tag.

1.7.2 Paragraph

GQ. Explain paragraph element in HTML. **(2 Marks)**

Use :

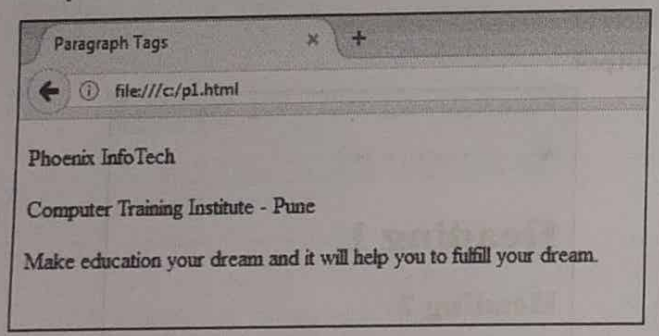
The tag is used to define paragraph in HTML. This tag structures our text in different paragraphs. The paragraph of text is enclosed in opening and closing tags.

Program 1.7.2 : Write a simple program to define the paragraph tag.

Soln. : Program with the use of paragraph tag

```
<!DOCTYPE html>
<html>
  <head>
    <title>Paragraph Tags</title>
  </head>
  <body>
    <p>Phoenix InfoTech</p>
    <p>Computer Training Institute - Pune</p>
    <p>Make education your dream and it will help you to fulfill your dream.</p>
  </body>
</html>
```

Output



1.7.3 Line Breaks

GQ. Explain line break element in HTML. **(2 Marks)**

Use :

The
 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.

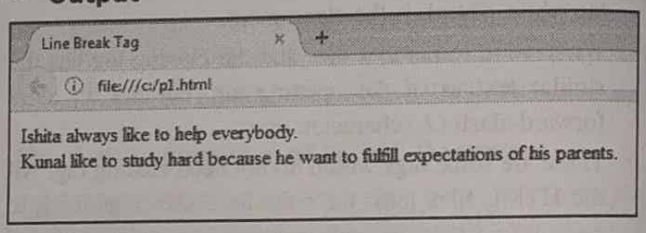
Program 1.7.3 : Write a simple code for line break example.

Soln. :

Program showing the line break example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Line Break Tag</title>
  </head>
  <body>
Ishita always like to help everybody.
<br>
Kunal like to study hard because he want to fulfill
expectations of his parents.
  </body>
</html>
```

Output



1.7.4 Colors and Fonts

1.7.4(A) Colors

GQ. Explain setting colors in HTML. **(4 Marks)**

- The look and feel of a website is dependent upon the colors used in a webpage for text and backgrounds.
- The colors can be specified at page level with the help of <body> tag or can be set for individual tags.

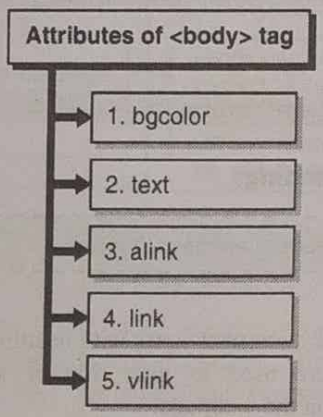


Fig. 1.7.1 : Attributes of <body> tag

The <body> tag has number of attributes which are used to set colors to different entities.

1. **bgcolor :** Used to set color to the background of the page.
2. **text :**Used to set color to the body text.
3. **alink :** Used to set color to the active links or selected links.
4. **link :**Used to set color to the linked text.
5. **vlink :** Used to set color to the visited links – that is, for linked text that you have already clicked on.

☞ Color Coding Methods in HTML

There are different methods to set colors in the web page as follows :

(i) **Color names** – We can directly specify color name like red, green, blue etc.

(ii) **Hex codes** – It is the six-digit code which represents the expected color.

(iii) **Color decimal or percentage values** – This is also called as RGB color scheme. Here we have to specify the amount of colors red, green and blue. The mixture of this gives expected color.

Now we will see these color methods in detail.

☞ HTML Colors - Color Names

- In this method the color names are directly assigned to set color for text or background. The W3C (World Wide Web Consortium) has listed 16 standard color names for HTML. But maximum of browsers supports more that 200 color names.
- The W3C Standard 16 Colors are as follows:

  Black, Yellow, Red, Maroon, Gray, Lime, Green, Olive, Silver, Aqua, Blue, Navy, white, Fuchsia, purple, Teal.

☞ HTML Colors - Hex Codes

- This is the six digit representation of a color. The initial two digits (RR) represent the red value, the next two digits represent green value (GG), and the last two digits represent the blue value (BB).
- There are various graphics software like MS Paint, Adobe Photoshop or Paintshop Pro where we can get the hexadecimal value.
- Each hexadecimal code is preceded with pound or hash sign (#).

☞ HTML Colors - RGB Values

- The RGB method takes three values as arguments for Red, Green and Blue colors. These values are in integer format which range from 0 to 255 depending upon the required intensity of color.
- Not all the browsers support this method, hence not suggestive.

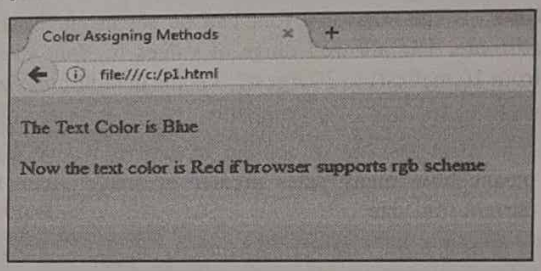
Program 1.7.4 : Write a simple code to display the text in various color format.

☑ Soln. :

Code to display the text in various color format

```
<!DOCTYPE html>
<html>
  <head>
    <title>Color Assigning Methods</title>
  </head>
  <body text = "blue" bgcolor = "#00FF00">
    <p>The Text Color is Blue</p>
    <font color = "rgb(255,0,0)">Now the text color is Red
if browser supports rgb scheme</font>
  </body>
</html>
```

Output



1.7.4(B) Font

GQ. Explain setting fonts in HTML. **(4 Marks)**

- User friendliness is the basic need of any website. This can be achieved by making it readable with the help of <font> tag. The text in a webpage can be formatted by setting the < font > ..... < /font > tag and various font attributes.
- The <font> tag has following attributes :

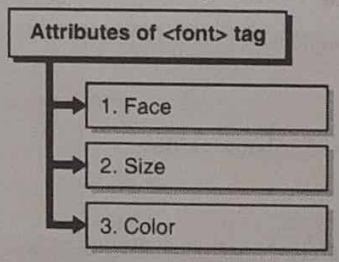


Fig. 1.7.2 : Attributes of <font> tag

1. Face

☞ **Use :** 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

☞ **Use :** 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

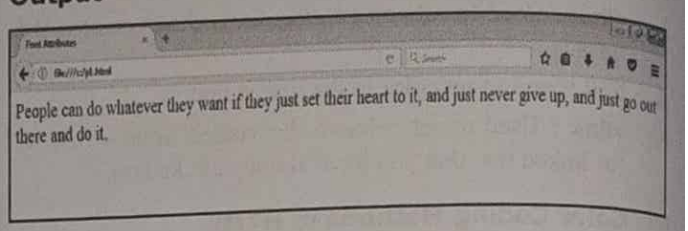
☞ **Use :** Used to set the color of font. This attribute can accept value as standard name of a color or color code.

Program 1.7.5 : Write a program to display font size, font face and color.

☑ Soln. : Program to display the font size, color and font face

```
<!DOCTYPE html>
<html>
 <head>
  <title>Font Attributes</title>
 </head>
 <body>
<font face = "Times New Roman,Sans serif" size = "5"
color="red">
People can do whatever they want if they just set their heart to
it, and just never give up, and just go out there and do it.
</font>
 </body>
</html>
```

Output



1.7.5 Links - Hyperlinks

UQ. Explain following HTML Tags with suitable example <a> **SPPU - Aug. 2014 - In sem, 2.5 Marks**

UQ. What are the Hyperlinks and Anchors in the HTML? Explain with suitable example(program). **SPPU - Oct. 2016 - In sem, 4 Marks**

In a website there are multiple web pages.

☞ **Use**

- **Hyperlinks** are used to navigate in the website. That means to move from one webpage to another. A webpage can have various hyperlinks which can take us directly to another pages and even specific parts of the current page. For hyper linking words, phrases or even images can be used.
- The **anchor tag <a>** is used to specify the hyperlink. The text written in opening <a> and closing tag is known as hypertext. When this hypertext is clicked, the target webpage get opened. The hypertext has default formatting. It is in blue color and has underline.

Program 1.7.6 **SPPU - Aug. 2014, Oct. 2016 - In sem**

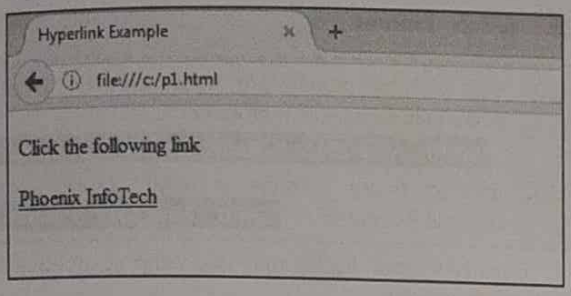
Write a program to display hyperlink example.

☑ Soln. : Program to display the hyperlink example

```
<!DOCTYPE html>
<html>
 <head>
  <title>Hyperlink Example</title>
 </head>
 <body>
  <p>Click the following link</p>
  <a href = "http://www.PhoenixGlobe.com">Phoenix
InfoTech</a>
 </body>
</html>
```

The **href** attribute is used to specify the address of webpage which we want to display.

Output



When this link is clicked, the home page of website will display.

1.7.5(A) The Target Attribute

Use

The target attribute is used to specify location where the linked webpage should be opened. Different values for target attribute are as follows :

Sr. No	Option	Description
1	_blank	The linked webpage is opened in new window or tab.
2	_self	The linked webpage is opened in the same frame.
3	_parent	The linked webpage is opened in the parent frame.
4	_top	The linked webpage is opened in the full body of the window.
5	targetframe	The linked webpage is opened in a named targetframe.

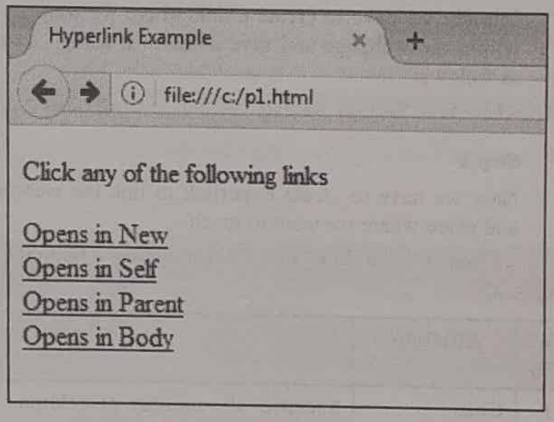
Program 1.7.7 : Write a program to display the hyperlinks using target attribute.

Soln. :

Program of hyperlinks using target attribute

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hyperlink Example</title>
  </head>
  <body>
    <p>Click any of the following links</p>
    <a href = "http://www.PhoenixGlobe.com"
    target = "_blank">Opens in New</a> <br>
    <a href = "http://www.PhoenixGlobe.com"
    target = "_self">Opens in Self</a> <br>
    <a href = "http://www.PhoenixGlobe.com"
    target = "_parent">Opens in Parent</a> <br>
    <a href = "http://www.PhoenixGlobe.com"
    target = "_top">Opens in Body</a>
  </body>
</html>
```

Output



Use of Base Path

When web pages related to same website are linked, then it is not necessary to specify the complete path of the webpage. This path is known as URL. In document header we can use <base> tag to set base path for all the links. The browser will concatenate the given relative path with this base path and a complete URL is generated.

Program 1.7.8 : Write a simple code for <base> tag.

Soln. :

Simple code for <base> tag

In this program we will use <base> tag to specify the base URL and afterwards we can use the relative path for all the links rather than giving full URL for every link.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Hyperlink Example</title>
  <base href = "http://www.PhoenixGlobe.com/">
 </head>
 <body>
  <p>Click following link</p>
  <a href = "/html/student_corner.html"
  target = "_blank">Student Corner</a>
 </body>
</html>
```

Linking to a Page Section

It is also possible to link specific portion of the webpage with the help of name attribute of anchor <a> tag. There are two steps to implement it.

Step 1

Initially we have to create a link where we want to go within the webpage and give name to it using <a> tag as follows:

```
<h1>Link Section <a name = "top"></a></h1>
```

Step 2

Now we have to create hyperlink to link the webpage and place where we want to reach –

```
<a href = "/html/html_text_links.htm#top">Go to the Top</a>
```

This will generate following link, where we can click to reach to the top of the webpage.

Go to the Top

1.7.6 Frames

UQ. Explain how frames are constructed in HTML document. Explain with program. **SPPU - May 2015, 5 Marks, Dec. 2015, 4 Marks**

UQ. How to create a frame in HTML? Explain with example (Program). **SPPU - May 2017, 5 Marks**

Usually we can display only one webpage at a time in the browser window.

Use :

- If we wish to display multiple pages at the same time on the browse window, then we can use the frames to divide the browser window into different sections and in each section we can display a separate webpage. A group of frames in the browser window is called as a frameset.
- While dividing the window, it is considered as a table and organized in rows and columns.
- In the frameset application, basic tag of html like <body> is not used. Instead of it, <frameset> tag is used. This tag defines how to divide the window into different sections. There are two main attributes of frameset tag.
  1. **Rows** – Defines the horizontal frames.
  2. **Cols** – Defines the vertical frames
- The <frame> is sub-tag of <frameset> tag. The <frame> tag has attribute src (source) to which we have to specify the URL of webpage which we want to display in particular frame.

Attributes of the <frameset> tag

Following are the important attributes of tag <frameset>:

Sr. No	Attribute	Description
1.	Cols	Specifies the number of columns contained in the frameset and also the size of all the columns. Width of each column can be specified in four ways: (1) Absolute values can be given in pixels. For example, to generate three vertical frames, we can set cols = "200, 400, 100". (2) A percentage amount of the current browser window. For example, to generate three vertical frames, use cols = "20%, 70%, 10%". (3) Using the special wildcard symbol. For example, to generate three vertical frames, use cols = "20%, *, 20%". Here the wildcard takes remaining percentage of the window. (4) As relative widths of current browser window. For example, to generate three vertical frames, use cols = "3*, 2*, 1*". This is substitute of percentages. The relative widths of the browser

Sr. No	Attribute	Description
		window can be used. This divides the window as: first column occupies half of the window, the second occupies one third, and the third occupies one sixth.
2.	rows	This attribute is same as of the cols attribute and accepts the same values, but it is used to mention the rows in the frameset. For example, to generate two horizontal frames, use rows = "20%, 80%". The height of each row can be specified in the same way as we have seen for columns.
3.	border	The width of the border of each frame can be specified with border attribute. For example, border = "3". A value of zero indicates frame without border.
4.	frameborder	This attribute mentions whether to display a three-dimensional border between frames or not. It takes value either 1 (true) or 0 (false). For example frameborder = "1" specifies frame with three-dimensional border.
5.	Framespacing	The amount of space between different frames in a frameset can be specified with the help of this attribute. It accepts any integer value. For example framespacing = "5" indicates that there should be 5 pixels spacing between the frames.

☞ The <frame> Tag Attributes

Following are the important attributes of the sub-tag <frame> :

Sr. No	Attribute	Attribute & Description
1.	Src	This attribute specifies the URL of web page that should be loaded in the frame. **Example :** src = "/html/index.htm".
2.	Name	This attribute specifies the name to a frame. This is usually important when we want to establish links in one frame that load web pages into another frame. In this case the second frame must have a name to identify itself as the target for the link.
3.	Frameborder	This attribute is used to set whether or not the borders of the frame should be displayed; it overwrites the value specified in the frameborder attribute of the <frameset> tag if it is specified. It takes value either 1 (true) or 0 (false).
4.	Marginwidth	This attribute specifies the width of the space between the left and right borders of the frame and the content of the frame. The value is specified in pixels. **Example :** marginwidth = "5".
5.	Marginheight	This attribute specifies the height of the space between the top and bottom borders of the frame and its contents. The value is specified in pixels. For example marginheight = "5".
6.	Noresize	The borders of the frames can be resized by default. If we want to restrict user from resizing the frame then we can set : noresize = "noresize".
7.	Scrolling	This attribute specifies the behavior of the scrollbars which appear on the frame. It accepts value either "yes", "no" or "auto". **Example :** scrolling = "yes" indicates that the frame should have scroll bars.
8.	longdesc	This attribute specifies link to another page which contains a long description about the contents of the frame. **Example :** longdesc = "details.htm"

☞ Browser Support for Frames

- If any user has old browser or such a browser which does not have support for frames then the <noframes> element is shown on the browser.
- In such case we have to place the <body> element inside the <noframes> element since the <frameset> element should be replaced by the <body> element.

Program 1.7.9 SPPU - May 2015, Dec. 2015, May 2017
Write a code to create horizontal frames.

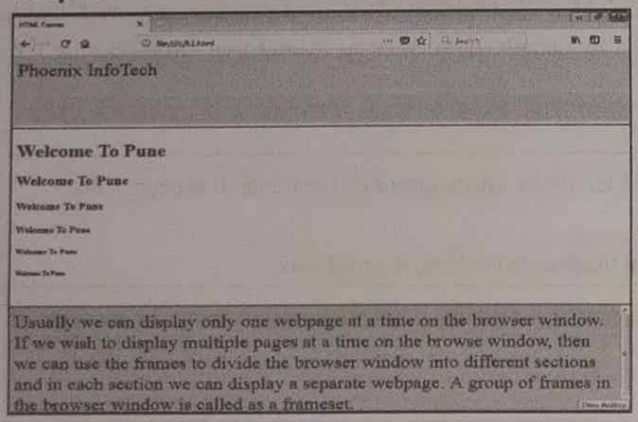
☑ **Soln. :**

Code to create the horizontal frames

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Frames</title>
  </head>
  <frameset rows = "20%,50%,30%">
    <frame name = "top" src = "First.html" />
    <frame name = "main" src = "Second.html" />
    <frame name = "bottom" src = "Third.html" />
    <noframes>
     <body>Your browser does not support frames.</body>
    </noframes>
  </frameset>
</html>
```

➠ **Note** : Here it is considered that pages First.html,Second.html and Third.html are already created.

Output



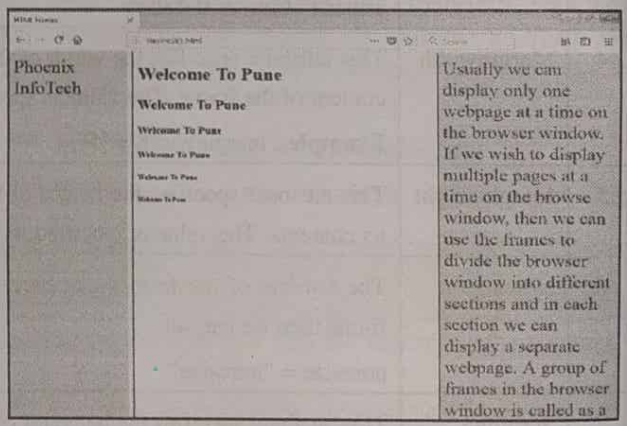
Program 1.7.10 SPPU - Aug. 2014 - In sem, 2.5 Marks
Write a code to create the vertical frames.

☑ **Soln. :**

Code to create the vertical frames

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Frames</title>
  </head>
  <frameset cols = "20%,50%,30%" border=10
  framespacing=200>
    <frame name = "left" src = "First.html" />
    <frame name = "center" src = "Second.html" />
    <frame name = "right" src = "Third.html"
    scrolling="no"/>
    <noframes>
      <body>Your browser does not support
      frames.</body>
    </noframes>
  </frameset>
</html>
```

Output



☞ Frame's name and target attributes

One of the best uses of frame is to place navigation bars in one frame and then load the link web pages into a separate frame.

Program 1.7.11 : Write a code to divide the browser window into two frames.

☑ Soln. : Code to divide the browser window into two frames

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Target Frames</title>
  </head>
  <frameset cols = "200, *">
    <frame src = "menu.html" name = "menu_page" />
    <frame src = "main.html" name = "main_page" />
  </frameset>
</html>
```

Here the browser window is divided into two frames. The first frame is small and will contain the only the navigation menu options of menu.html page. The second column occupies near about 80% of size and contain the main part of the page and it is implemented by main.htm file. In the links of the menu we will specify the target frame as main_page. Whenever any link in the menu is clicked, the related page will be opened in main page.

Following is the content of menu.html file

```
<!DOCTYPE html>
<html>
  <body>
    <a href = "http://www.TechMaxBooks.com"
    target = "main_page">Tech-Max</a>
    <br />
    <br />
    <a href = "http://www.PhoenixGlobe.com"
    target = "main_page">Phoenix InfoTech</a>
    <br />
    <br />
    <a href = "http://www.edudecision.com"
    target = "main_page">Edudecision</a>
  </body>
</html>
```

Following is the content of main.html file –

```
<!DOCTYPE html>
<html>
  <body bgcolor = "skyblue">
    <h3>Welcome To Our Web</h3>
    <p>Click any link and page will be displayed here.</p>
  </body>
</html>
```

Output

1.7.6(A) iframe

GQ. Explain the concept of iframe with suitable example **(2 Marks)**

iframe stands for inline frame. It is defined using tag <iframe>.

☞ Use :

The <iframe> tag defines a region with rectangular shape inside the main page where the browser should display another webpage with scrollbars and borders.

The URL of webpage which occupies the inline frame is specified by the src attribute.

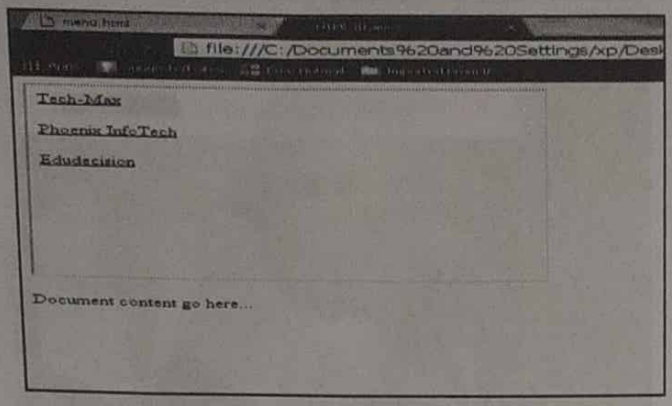
Program 1.7.12 : Write a code to define <iframe> tag.

☑ Soln. :

Simple code to define <iframe> tag

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Iframes</title>
  </head>
  <body>
    <iframe src = "menu.html" width = "400"
    height = "250">
    </iframe>
    <p>Document content go here...</p>
  </body>
</html>
```

Output



1.7.7 Lists

GQ. Explain the Lists in HTML. **(4 Marks)**

☞ **Use :** As the name suggests, these elements are used to give list of items. 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.

☞ **Types of HTML list**

HTML provides three ways to give the lists :

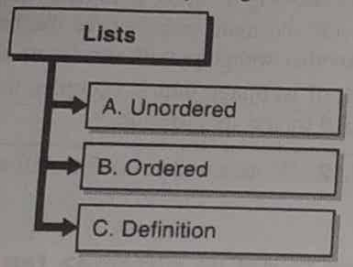


Fig. 1.7.3 : Types of HTML list

- **A. – An unordered list. Plain bullets are used for the list items.**
- **B. – An ordered list. Different schemes of numbers are used for the list items.**
- **C. <dl> – A definition list. This is used to give list of definitions just like dictionaries.**

I. HTML Unordered Lists

☞ **Use :** 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.

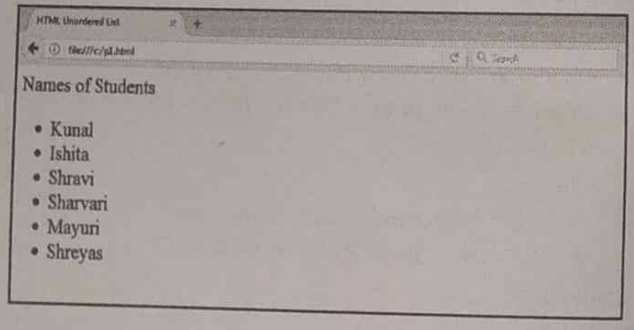
The <li> which is the sub-tag of <ul> is used for individual list elements.

Program 1.7.13 : Write a simple code demonstrating unordered list.

☑ **Soln. : Code to display an unordered list**

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Unordered List</title>
  </head>
  <body>
<font size=5>
Names of Students
    <ul>
      <li>Kunal</li>
      <li>Ishita</li>
      <li>Shravi</li>
      <li>Sharvari</li>
      <li>Mayuri</li>
      <li>Shreyas</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 are following options for the attribute *TYPE* of <ul> tag.

  (i) Square (ii) Circle (iii) Disc

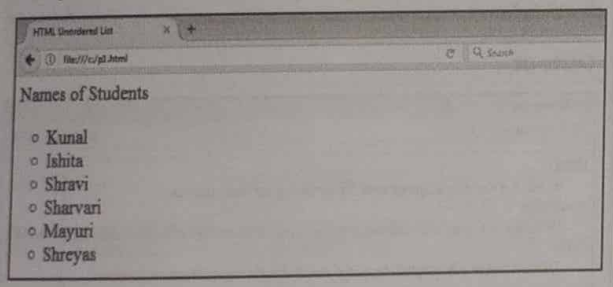
See the following example which illustrates the use of attribute *TYPE*.

Program 1.7.14 : Write a code illustrating the use of attribute TYPE.

☑ Soln. : Code illustrating the use of TYPE attribute

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Unordered List</title>
  </head>
  <body>
<font size=5>
Names of Students
    <ul type="circle">
      <li>Kunal</li>
      <li>Ishita</li>
      <li>Shravi</li>
      <li>Sharvari</li>
      <li>Mayuri</li>
      <li>Shreyas</li>
    </ul>
</font>
  </body>
</html>
```

Output



II. HTML Ordered Lists

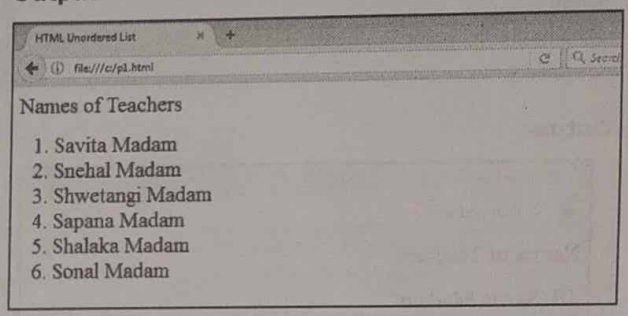
☞ **Use :** Ordered lists are generally used when we want to specify numbers instead of bullets. The <ol> tag is used for such list. The number starts from 1 and also incremented by one for each successive list element in the ordered list.

Program 1.7.15 : Write a program to display the ordered list.

☑ Soln. : Program to display the ordered list

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML ordered List</title>
  </head>
  <body>
<font size=5>
Names of Teachers
    <ol>
      <li>Savita Madam</li>
      <li>Snehal Madam</li>
      <li>Shwetangi Madam</li>
      <li>Sapana Madam</li>
      <li>Shalaka Madam</li>
      <li>Sonal Madam</li>
    </ul>
</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 are following options for the attribute *TYPE* of <ul> tag.

  1 - Number

  I – Upper Roman

  i – Lower Roman

  A – Upper Alpha

  a – Lower Alpha

See the Program 1.7.16 which illustrates the use of attribute *TYPE*.

☞ **The start Attribute**

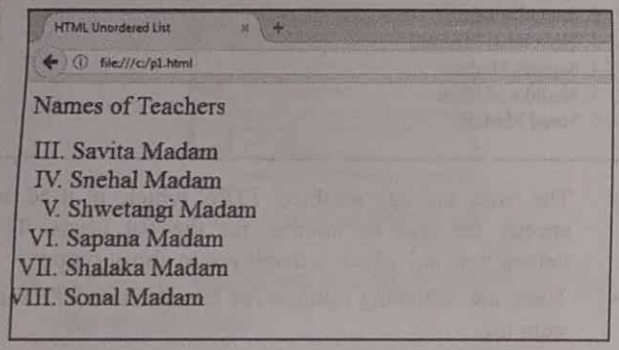
This attribute of <ol> tag is used to specify the starting point of numbering.

Program 1.7.16 : Write a program to illustrate the use of TYPE and start attribute in ordered list.

☑ **Soln. :**Program to illustrate the use of TYPE and start attribute in ordered list

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML ordered List</title>
  </head>
  <body>
<font size=5>
Names of Teachers
    <ol type="I" start="3">
      <li>Savita Madam</li>
      <li>Snehal Madam</li>
      <li>Shwetangi Madam</li>
      <li>Sapana Madam</li>
      <li>Shalaka Madam</li>
      <li>Sonal Madam</li>
    </ul>
</font>
  </body>
</html>
```

Output



III. HTML Definition Lists

☞ **Use :** 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.

(i) **<dl>** – Defines the beginning of the list

(ii) **<dt>** – Specify a term

(iii) **<dd>** – Specify term definition

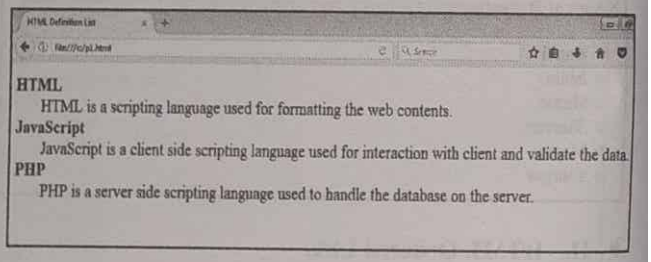
(iv) **</dl>** – Defines the end of the list

Program 1.7.17 : Write a simple code for definition list with the help of various tags in definition list.

☑ **Soln. : Code for definition list using various tags in <dl>**

```
<!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



1.7.8 Tables

UQ. How to create Tables on the web pages using HTML? **SPPU - Aug. 2014 - In sem, 4 Marks**

☞ **Use :**

- 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.
- Table 1.7.1 shows the list of table related tags.

Table 1.7.1

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.

1.7.8(A) Difference between <tr> and <td>

UQ. What is the difference between <tr> and <td>? **SPPU - May 2015-In Sem, 2 Marks**

Sr. No.	Parameters	<tr>	<td>
1.	Longform	<tr> stands for table row	<td> stands for table data
2.	Use	Used to create row in a table	Used to create cell in a row
3.	Hierarchy	<tr> is parent tag of <td>	<td> is sub tag of <tr>

Attributes of <table> tag

The <table> tag has following attributes

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
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 1.7.18 **SPPU - May 2015 - In sem, 2 Marks**

Write HTML code which includes table.

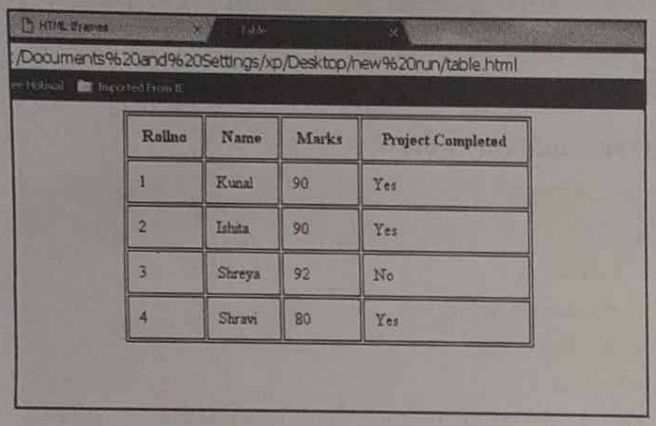
Soln. :

HTML code for table

```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
  <body>
<table border=1 bordercolor="red" width=40%
height = 40% cellspacing=2 cellpadding=10
bgcolor="gray" align="center">
<tr>
<th>Rollno</th>
<th>Name</th>
<th>Marks</th>
<th>Project Completed</th>
</tr>
<tr>
<td>1</td>
<td>Kunal</td>
<td>90</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Ishita</td>
<td>90</td>
<td>Yes</td>
```

```
</tr>
<tr>
<td>3</td>
<td>Shreya</td>
<td>92</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Shravi</td>
<td>80</td>
<td>Yes</td>
</tr>
</table>
</body>
</html>
```

Output

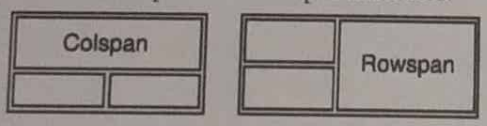


Rollno	Name	Marks	Project Completed
1	Kunal	90	Yes
2	Ishita	90	Yes
3	Shreya	92	No
4	Shravi	80	Yes

1.7.8(B) Colspan and Rowspan Attributes

GQ. Explain Colspan and Rowspan attributes of table tag with suitable examples (programs). **(4 Marks)**

- 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.



(a) (b)

Fig. 1.7.4 : Colspan and Rowspan Attributes

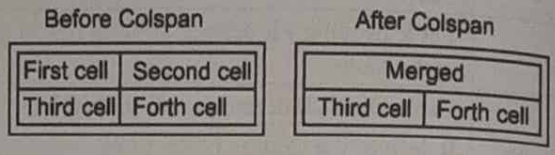
1. Colspan

Use :

- 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.



(a) (b)

Fig. 1.7.5 : Colspan

In the Fig. 1.7.5 there are two tables. In the first Fig. 1.7.5(a), 2 rows and 2 columns in each row are present. In the second Fig. 1.7.5(b), 2 rows and 1 column in first row and 2 columns in second row are present. Now In the second Fig. 1.7.5(b) first two Cells are merged horizontally with the help of Colspan attribute.

Code for Fig. 1.7.5(b)

```
<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.

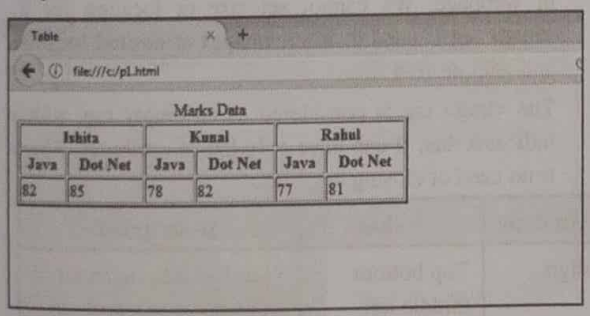
Program 1.7.19 : Write a program demonstrating the use of colspan attribute in table.

Soln. : Program using colspan attribute

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

```
  <title>Table</title>
 </head>
 <body>
<table border=2 width="40%">
<caption>Marks Data</caption>
<tr>
<th colspan="2">Ishita</th>
<th colspan="2">Kunal</th>
<th colspan="2">Rahul</th>
</tr>
<tr>
<th>Java</th>
<th>Dot Net</th>
<th>Java</th>
<th>Dot Net</th>
<th>Java</th>
<th>Dot Net</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

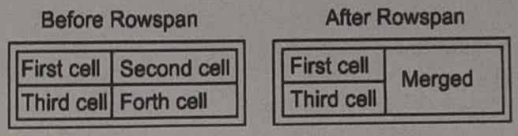


2. Rowspan

☞ **Use :** 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 give code will merge two Cells into one Cell vertically.



(a) (b)

Fig. 1.7.6 : Rowspan

In the above Fig. 1.7.6 we can see there are two tables. In the first Fig. 1.7.6(a) there are 2 rows and each row has 2 columns. In the second Fig. 1.7.6(b) 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.

☞ **Code for Fig. 1.7.6(b)**

```
<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.

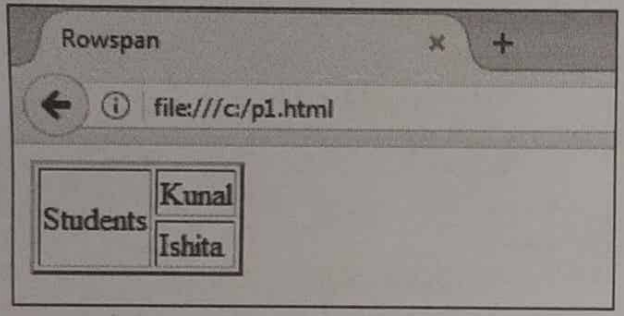
Program 1.7.20 : Write a program demonstrating the use of rowspan attribute in table.

☑ **Soln. : Program using rowspan attribute**

```
<html>
<head>
<title>
Rowspan
</title>
</head>
```

```
<body>
<table border=2>
  <tr>
    <td rowspan="2">
    Students
    </td>
    <td>
   Kunal
    </td>
  </tr>
  <tr>
    <td>
    Ishita
    </td>
  </tr>
</table>
</body>
</html>
```

Output



1.7.9 Images and Forms

1.7.9(A) Images

UQ. Explain how the images can be inserted in HTML document with program.

SPPU - Aug. 2016 - In sem, 5 Marks

- Images are important in the designing of our webpage. Images are used to describe some complex concepts in simple pictorial format on the web page.
- In HTML there are two options to **insert image** in our webpage.

(1) Using <body> tag

The body tag has background attribute to set image as a background to the webpage.

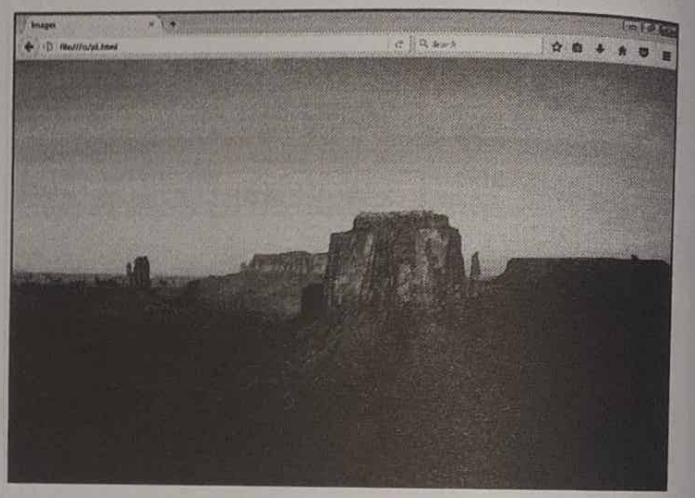
Program 1.7.21 **SPPU - Aug. 2016 - In sem**

Write a simple code for setting the image at the background.

☑ **Soln. :Code for setting the image as background**

```
<html>
<head>
<title>
Images
</title>
</head>
<body background = "Desert.jpg">
</body>
</html>
```

Output



(2) Using <img> tag

- The background image occupies the entire background of webpage. We cannot set size or location for it. <img> tag is used to insert images at desired location and with desired size.
- The <img> tag is considered as an empty tag, which indicates that, it can have only list of attributes. There is no need of closing tag.

Attribute	Value	Description
align	Top bottom middle left right	Specifies the alignment of an image with respect to the surrounding elements
alt	text	Defines an alternate text for the image
border	pixels	Defines the width of the border around an image

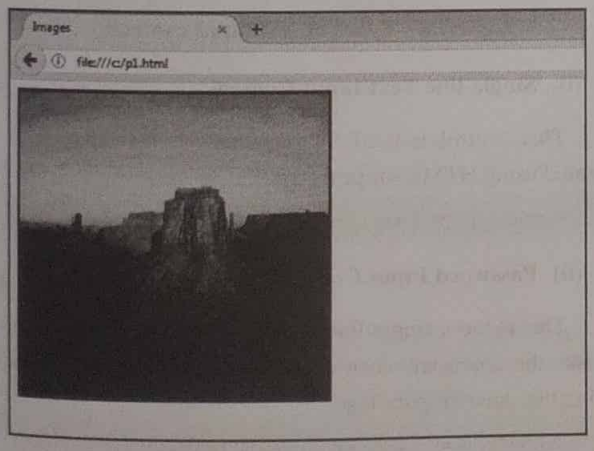
Attribute	Value	Description
height	pixels	Defines the height of an image
hspace	pixels	Defines the whitespace on both of the left and right side of an image
src	URL	Defines the URL of an image
usemap	#mapname	Defines an image as a client-side image-map
vspace	pixels	Defines the whitespace on top and bottom of an image
width	pixels	Specifies the width of an image

Program 1.7.22 : Write a program to display the image.

Soln. : Program to display the image

```
<html>
<head>
<title>
Images
</title>
</head>
<body>
<img src="Desert.jpg" width= 300 height=300
alt="Phoenix InfoTech" border=3 >
</body>
</html>
```

Output



1.7.9(B) Image Maps

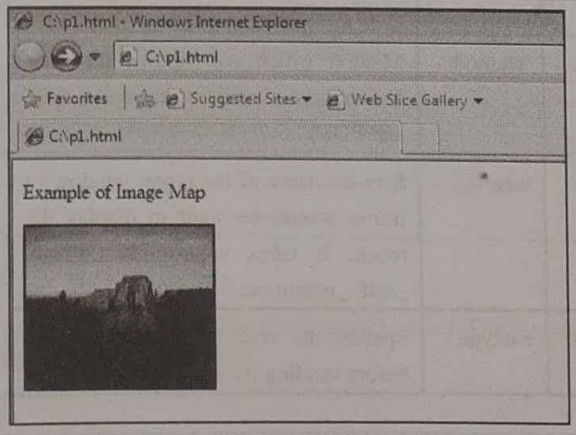
UQ. Explain Image Map with example (program). **SPPU - May 2015, Dec. 2015, 5 Marks**

- HTML provides the <map> tag to define a client-side image-map. It is an image with clickable areas.
- The <map> element has name attribute which is associated with the attribute use map and links the image with the map.
- The <map> element has sub-tag <area> , which defines the clickable areas in the image map.

Program 1.7.23 **SPPU - May 2015, Dec. 2015**
Write a program to demonstrate use of <map> tag.

Soln. : Program to demonstrate use of <map> tag

```
<!DOCTYPE html>
<html>
<body>
<p>Example of Image Map</p>
<img src="desert.jpg" width="145" height="126"
usemap="#smap">
<map name="smap">
 <area shape="rect" coords="0,0,82,126" alt="Jellyfish"
href="Jellyfish.htm">
 <area shape="circle" coords="90,58,3" alt="Koala"
href="Koala.htm">
 <area shape="circle" coords="124,58,8" alt="Penguins"
href="Penguins.htm">
</map>
</body>
</html>
```



1.7.9(C) Forms

GQ. Explain Form element in HTML. **(8 Marks)**

- 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.

Use

- 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>
```

Tag 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

HTML Form Controls

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

GQ. Explain Different Form elements / Form Controls. **(4 Marks)**

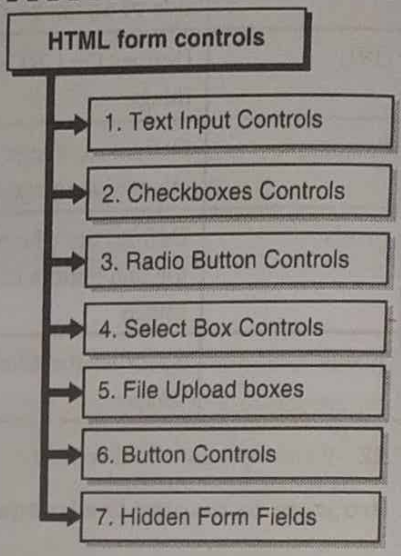


Fig. 1.7.7 : HTML form controls

1. Text Input Controls

Three types of text input controls are used on forms :

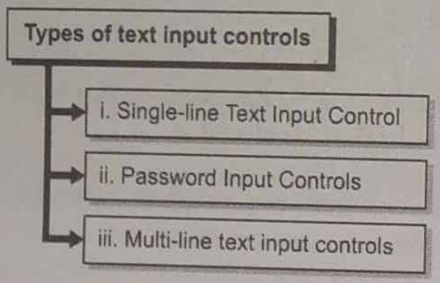


Fig. 1.7.8 : Types of text input controls

(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.

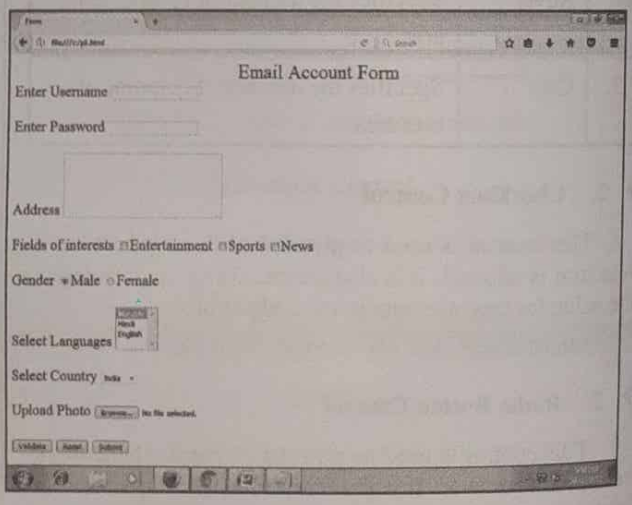
Program 1.7.24 : Write a program to display the form containing all controls.

☑ **Soln. :**

Program displaying the form containing all records

```
<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



1.8 DIFFERENCE BETWEEN HTML AND HTML5

UQ. Differentiate between HTML and HTML5

SPPU - Aug. 2014 - In sem, 5 Marks

Sr. No	Parameter	HTML	HTML5
1.	Audio and Video	Audio and Video tags are not provided in HTML4	Audio and Videos are provided in HTML5 e.g. <audio> and <video>.
2.	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
3.	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)
4.	Cookies	Html use cookies.	It supports local storage instead of cookies.
5.	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.
6.	Browser Support	Supported by old browsers	Supported by all new browsesr.
7.	Syntax	Doctype declaration in Html is very long <!DOCTYPE HTML PUBLIC "-//PhoenixGlobe//DTD HTML 4.01//EN" "http://www.PhoenixGlobe.com/TR/html4/strict.dtd">	DOCTYPE declaration in Html5 is very simple "<!DOCTYPE html>"
8	Character Encoding	Character encoding in Html is very long <!DOCTYPE HTML PUBLIC "-//PhoenixGlobe//DTD HTML 4.0 Transitional//EN">	Character encoding is very simple <meta charset="UTF-8">

1.9 CSS

1.9.1 Introduction to Style Sheet

- 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.

1.9.2 Features of CSS

GQ. State the features 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.

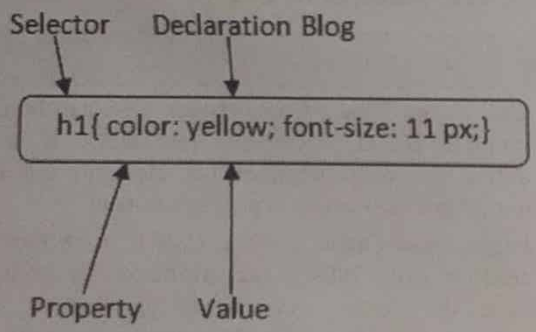
1.9.3 CSS Syntax

- A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.
- A CSS rule set contains a selector and a declaration block.
- A style rule is made of three parts:

1. **Selector :** A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
2. **Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.
3. **Value**: Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule Syntax as follows:

```
Selector{ property: value } OR
Selector{Property1: value1; Property2: value2; .....;}
```



Example

```
table {border :1px solid #C00;}
```

- Here table is a selector and border are a property and given value 1px solid #C00 is the value of that property.

1.9.4 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

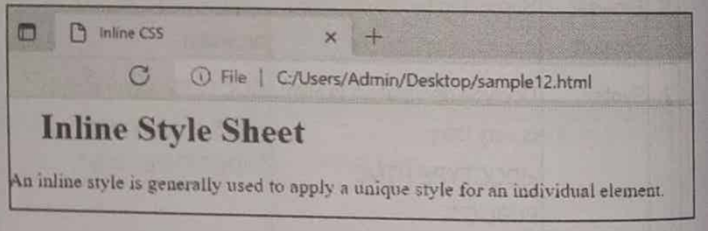
1. Inline Style Sheet

- 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 1.9.1

```
<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



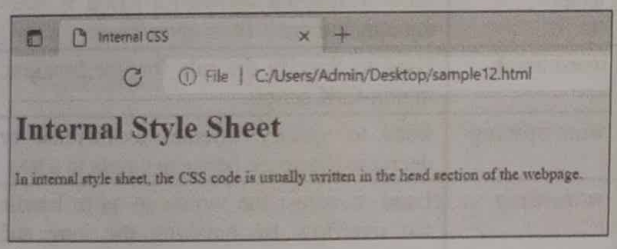
2. Internal Style Sheet

- 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 1.9.2

```
<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 1.9.3

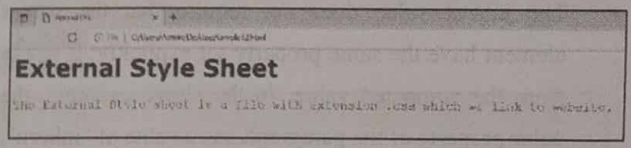
CSS file: style.css

```
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



1.10 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.

Program 1.10.1

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.

1.11 TEXT PROPERTIES

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 :

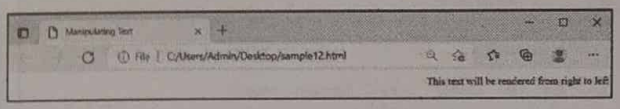
```
text-shadow:4px 4px 8px blue
```

- 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 1.11.1 : 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

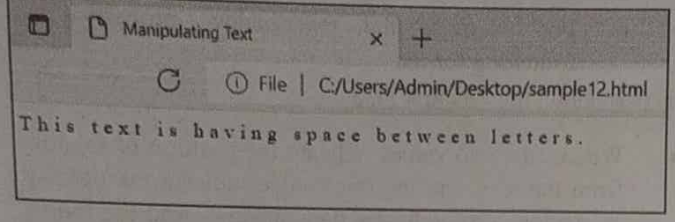


Program 1.11.2 : Write a code to demonstrate the use of letter-spacing property.

```
<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>
```

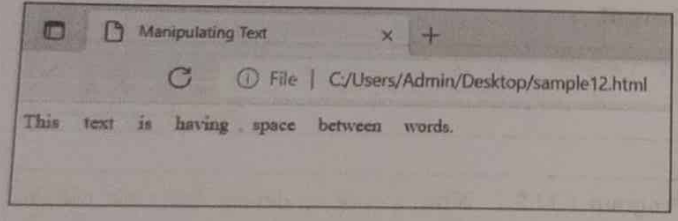
Output



Program 1.11.3: 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>
```

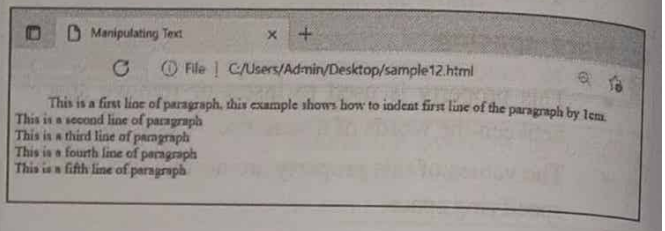
Output



Program 1.11.4 : 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>
```

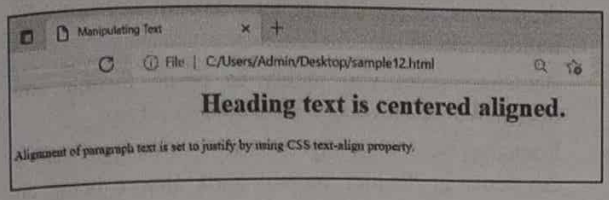
Output



Program 1.11.5 : 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>
```

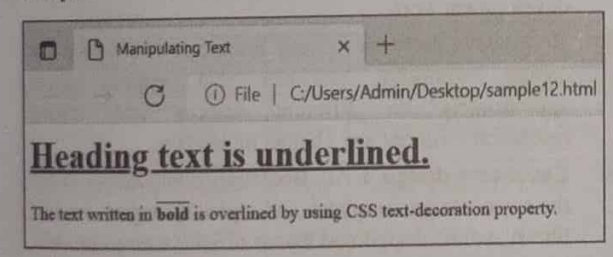
Output



Program 1.11.6 : 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>
```

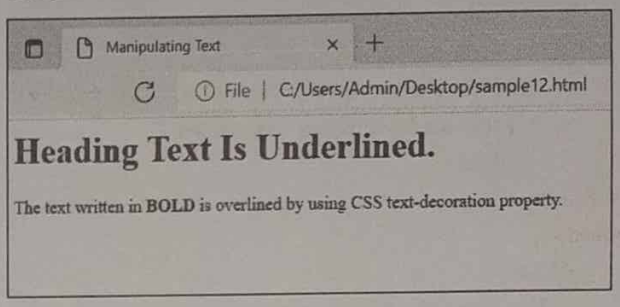
Output



Program 1.11.7 : 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>
```

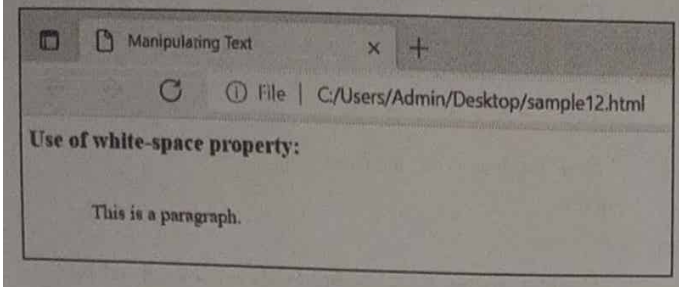
Output



Program 1.11.8 : 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>
```

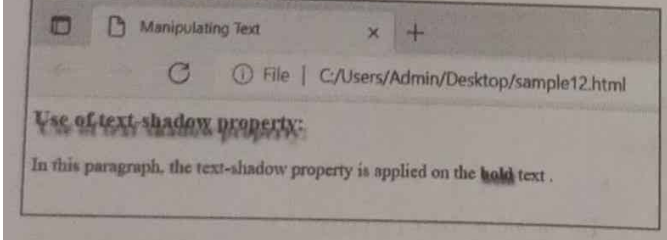
Output



Program 1.11.9 : 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



1.12 BOOTSTRAP

GQ. What You Can Do with 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.
- 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.

1.12.1 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.

1.12.2 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 1.12.1

```
<!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>
```

1.12.3 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

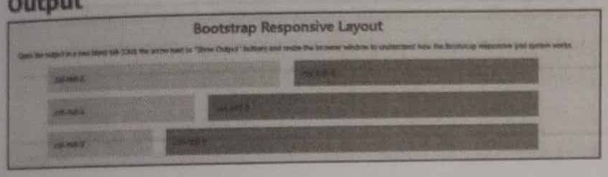
The following example will show you how to create two column layouts for medium, large and extra-large devices like tables, laptops and desktops etc. However, on mobile phones (screen width less than 768px), the columns will automatically become horizontal (2 rows, 1 column).

Program 1.12.2

```
<!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/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>
  /* 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

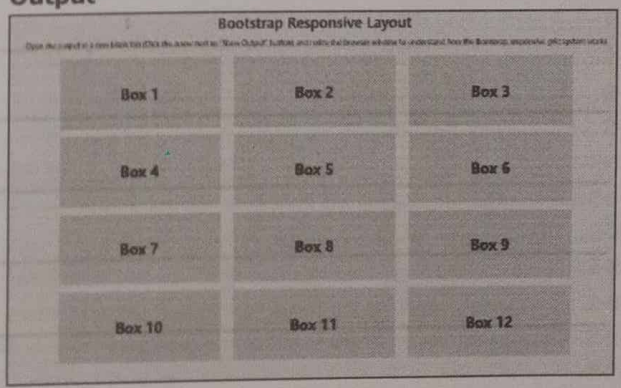


☞ Creating Multi-Column Layouts with Bootstrap

Program 1.12.3

```
<!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/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>
  /* 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



❑❑❑

UNIT II

CHAPTER 2

Client Side Technologies: JavaScript and DOM

Syllabus

JavaScript : Introduction to JavaScript, JavaScript in perspective, basic syntax, variables and data types, statements, operators, literals, functions, objects, arrays, built in objects, JavaScript debuggers. DOM: Introduction to Document Object Model, DOM history and levels, intrinsic event handling, modifying element style, the document tree, DOM event handling, jQuery, Overview of Angular JS.

2.1 INTRODUCTIONS TO JAVASCRIPT

Definition : JavaScript is an open source probably the most popular client side scripting language which is supported by all the browsers.

Use

- JavaScript is an extremely powerful **client-side scripting language**. This language is used mostly for increasing the interaction of an end user with the webpage.
- 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 supports dynamic scripting. It is lightweight and mostly used as an integral part of web pages where there is need of client side script and dynamic pages. JavaScript supports object oriented concepts.
- JavaScript was developed by Mr. Brendan Eich in 1995 who was working in Netscape.
- JavaScript was initially called as LiveScript and later on the name is changed to JavaScript. The syntax of JavaScript is typically influenced by the C programming language.

Client-side JavaScript

- This is the most common form of the language. The script (code) of JavaScript is embedded into HTML or referenced by an HTML document so as to be interpreted by the browser.
- It proves that the web page not necessarily is a static HTML, but it can include scripts that interact with the end user, controls the browser, and create HTML content dynamically.
- The client-side features of JavaScript have various benefits over the previous traditional CGI (Common Gateway Interface) server-side scripts. For example, JavaScript can be used to verify whether the user has given a valid e-mail id in a form field or not.
- When user submits the form, the JavaScript validates the data, if all the given data is valid then only it is submitted to the Web Server.
- JavaScript helps to handle events fired by user like button clicks, navigation through links, and other actions which are taken by user explicitly or implicitly.

2.1.1 Characteristics or Advantages of JavaScript

GQ. Explain characteristics of JavaScript. **(5 Marks)**

1. JavaScript is a **lightweight, interpreted** client side scripting language.
2. Designed for developing network-based applications.
3. JavaScript is complementary to Java.
4. JavaScript is complementary to and integrated with HTML.
5. It is Open source and cross-platform.
6. 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.
7. There is no need for the user to wait to see if something have been forgotten to enter.
8. Interactive interfaces can be created which can give responses to end user actions like mouse or keyboard activities.
9. 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 ? **(5 Marks)**

Advantages

1. Immediate response to user's actions which enables more interactivity
2. No need to go to server hence execution is fast.
3. Improve the usability of Web sites for users whose browsers support scripts.
4. Developer get more control over the look and behavior of their Web widgets.
5. Possible to substitute by HTML if users 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 there may occur errors if no alternatives have been provided.

Unit II In Sem.

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 to development if the scripts are not already available through other resources.
4. Sometimes the web widget looks like a standard control but their behavior 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? **(5 Marks)**

Parameter	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 JavaScript Development Tools

GQ. Explain JavaScript editing tools. **(5 Marks)**

One of main advantage of JavaScript is that there is no need of development tools which are expensive. A simple text editor like Notepad can be used to write the scripts. Inside the context of a web browser, JavaScript is an interpreted language; hence there is no need to buy a compiler.

JavaScript Editing Tools

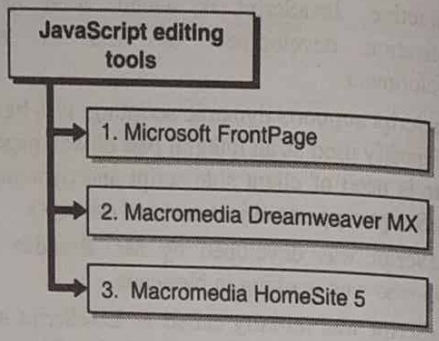


Fig. 2.1.1 : JavaScript editing tools

1. Microsoft FrontPage

This is popular product of Microsoft. It provides various JavaScript tools to web developers for assistance to create interactive websites.

2. Macromedia Dreamweaver MX

It is very popular HTML and JavaScript tool used for web development professionally. It provides various JavaScript components to handle databases, and supports new standards like XHTML and XML.

3. Macromedia HomeSite 5

This is a product of Macromedia which is well-liked HTML and JavaScript editor. It helps to effectively manage personal websites.

2.1.5 Difference between Java and JavaScript

GQ. Write the difference between java and JavaScript. **(5 Marks)**

Parameter	Java	JavaScript
Execution	Java creates application which can be executed on virtual machine or browser.	JavaScript creates application which can be executed on browser only.
Features	Java code allows programmer full functionality.	JavaScript code contains limited number of commands and features.
Naming	The first name of Java was OAK and was developed by James Gosling, Sun MicroSystems.	JavaScript was earlier known as LiveScript and was developed by Brendan Eich, Netscape.
Type Safety	Java is high-level, compiled and strongly typed language.	JavaScript is text based and weakly typed language.
Variables	Variables are created using the data type names like int, char double etc.	Variables are created using the var keyword.
Extension	Java program has file extension ".Java" and after compilation it creates ".class" file.	JavaScript file has file extension ".js" or ".html".
Objects	Objects of Java are class based.	Objects of JavaScript are prototype based.
Scope	Java has block based scope.	JavaScript has function based scope and object based context.

Unit II In Sem.

2.1.6 Using JS in an HTML (Embedded, External)

- 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>
```

2.1.7 Embedded JavaScript

GQ. What is Embedded JavaScript ? Explain with suitable example. **(10 Marks)**

Program 2.1.1 : Write an embedded JavaScript code displaying welcome message.

☑ Soln. : Embedded JavaScript code displaying welcome message

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

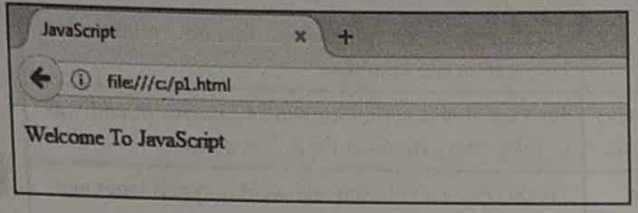
```
<!DOCTYPE html>
<html>
<head>
<title>
JavaScript
</title>
<script language="JavaScript">
document.write("Welcome To JavaScript");
</script>
</head>
```

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

document.write :

This method writes a string into HTML document.

Output



Explanation

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.

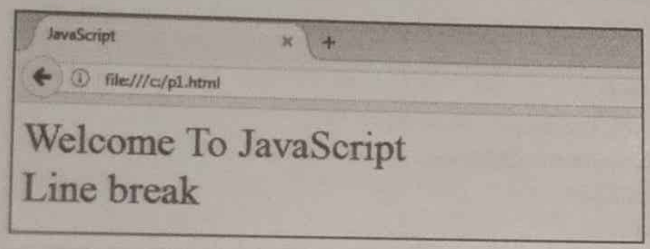
Program 2.1.2 : Write a program to demonstrate line break example.

☑ **Soln. :**

Program to demonstrate the line break

```
<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



2.1.8 External JavaScript

GQ. Write note on External JavaScript with example (program). **(10 Marks)**

- 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".

Program 2.1.3 : Write an external JavaScript code demonstrating welcome message.

☑ **Soln. :**

External JavaScript code demonstrating welcome message

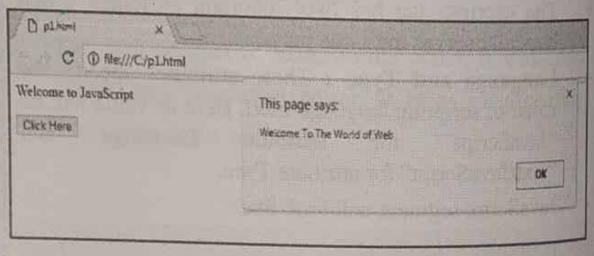
MyFile.js

```
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>
</html>
```

Output



2.2 JAVASCRIPT IN PERSPECTIVE

- JavaScript is a fully-fledged Object Oriented Programming (OOP) language. While HTML and CSS are used to annotate a documents, JavaScript employs several set of instructions and variables to produce an output.
- JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements.
- Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects; for example :
  (1) Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
  (2) Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.
- JavaScript is standardized at ECMA International — the European association for standardizing information and communication systems (ECMA was formerly an acronym for the European Computer Manufacturers Association) to deliver a standardized, international programming language based on JavaScript.
- This standardized version of JavaScript, called ECMAScript, behaves the same way in all applications that support the standard.
- Companies can use the open standard language to develop their implementation of JavaScript. The ECMAScript standard is documented in the ECMA-262 specification.

2.3 DATA TYPES AND VARIABLES

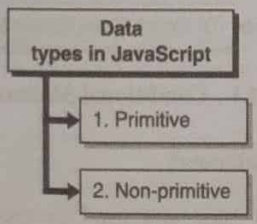


Fig. 2.3.1 : Data types in JavaScript

- Variable is a name given to memory location where we can store some value. The value depends upon the **data type** of 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. "Ishita"
2.	Number	Represents numeric values e.g. 101
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="Kunal";        //holding string
```

Unit II In Sem.

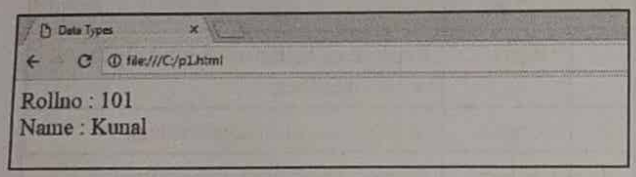
Program 2.3.1 : Write a program to display roll no. and name of student using "var" keyword.

☑ **Soln. :**

Program 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 = 101;
var sname = "Kunal";
document.write("Rollno : "+rno);
document.write("<br>Name : "+sname);
document.write("</font>");
</script>
</body>
</html>
```

Output



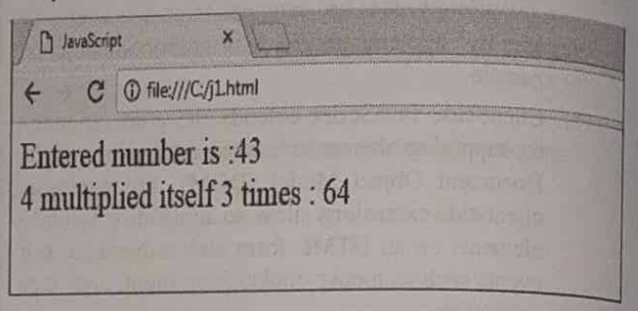
Program 2.3.2 : Write a JavaScript to take 2 digit number and then separate these 2 digits, then multiply first digit by itself for second digit times. (for example, 23 should be separated as 2 and 3. 2 should multiply with itself 3 times.

☑ **Soln. :**

JavaScript to multiply two numbers

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

<body>
<font size=5>
<script type="text/javascript">
var num=parseInt(prompt(("enter no")));
var rem=num%10;
document.write("Entered number is :" + num +"<br/>")
var quat=parseInt(num/10);
var result=Math.pow(quat, rem);
document.write(quat+" multiplied itself "+rem+" times : "+ result);
</script>
</font>
</body>
</html>
```

Output



2.4 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.

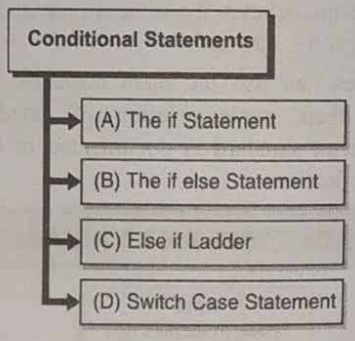


Fig. 2.4.1 : Conditional Statements

(A) The if Statement

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

Syntax

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

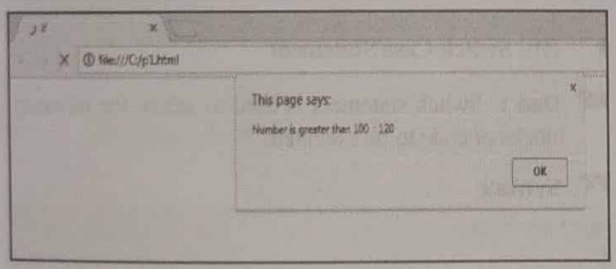
Program 2.4.1 : Write a program to accept a number and check whether it is greater than 100 or not.

Soln. :

Program to accept a number to 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



(B) The if else Statement

Use

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

Syntax

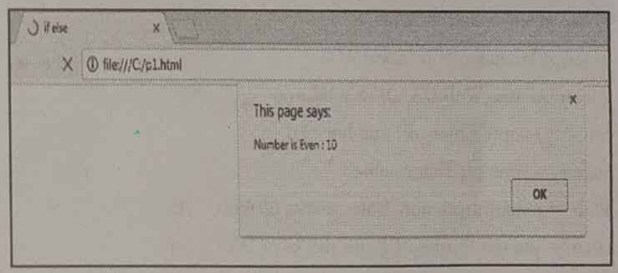
```
if (condition)
{
    statements;
}
else
{
    statements;
}
```

Program 2.4.2 : Write a program to accept a number from user and check whether it is even or odd.

Soln. : Program to check number 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



(C) else if Ladder

Use

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;
}
```

Program 2.4.3 : 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...)

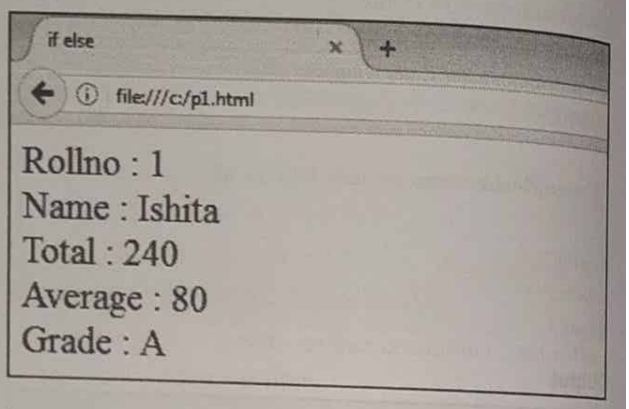
(**Note :** The student should get the grade only if he/she is pass in all the subjects.)

☑ **Soln. :**

Program to calculate the average and display the grades of students

```
<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 Web : "));
DAA = parseInt(prompt("Enter marks of DAA : "));
SYS = parseInt(prompt("Enter marks of SysPro and OS : "));
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



(D) Switch Case Statement

☞ **Use :** 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;
}
```

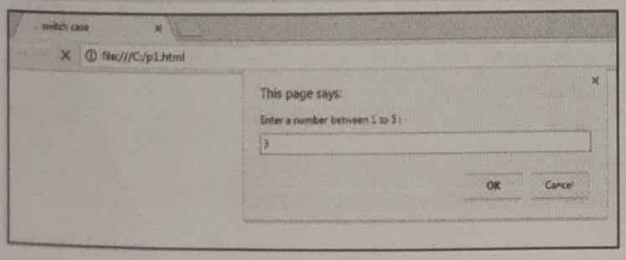
Working

- 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.

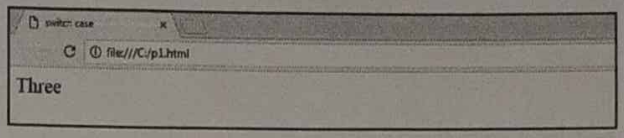
Program 2.4.4 : Write a program to print the numbers between 1 to 5.

Soln. : 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



Program 2.4.5 : Write a program to display the name of weekday using switch case.

Hint : The getDay() method returns the number of weekday in the range from 0 and 6.

(For example Sunday = 0, Monday = 1, Tuesday = 2)

Soln. :

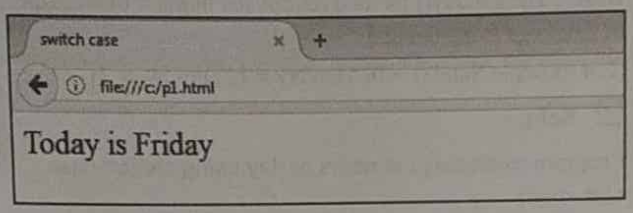
Program to display the name of day using switch case statement

```
<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";
}
```

Unit II In Sem.

```
document.write("Today is "+day);
</script>
</font>

</body>
</html>
```

Output



2.4.1 Loop Statements

GQ. Explain Loop statements in JavaScript. **(10 Marks)**

It contains while, do while and for loop.

(A) while Loop

Syntax

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

Working

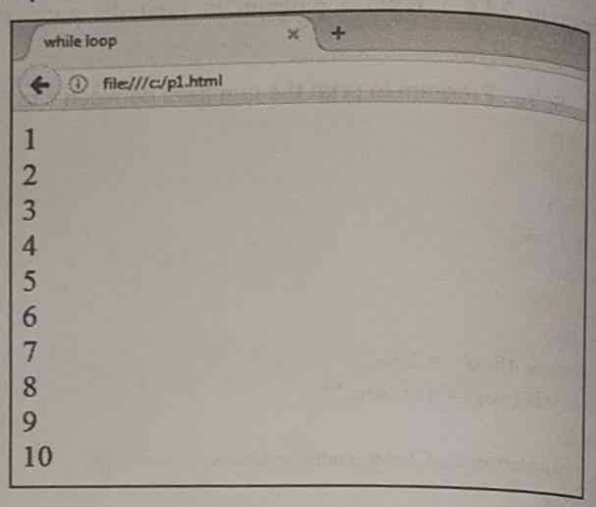
The while is an entry controlled loop. That means if the given condition is not satisfied then the loop statements will never get execute.

Program 2.4.6 : Write a program to print 1 to 10 numbers using while loop.

Soln. : 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

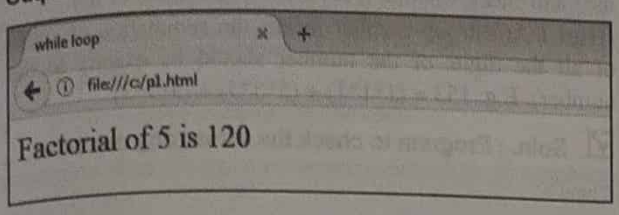


Program 2.4.7 : Write a program to accept a number from user and print factorial of it.

Soln. : Program to print factorial of a number

```
<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

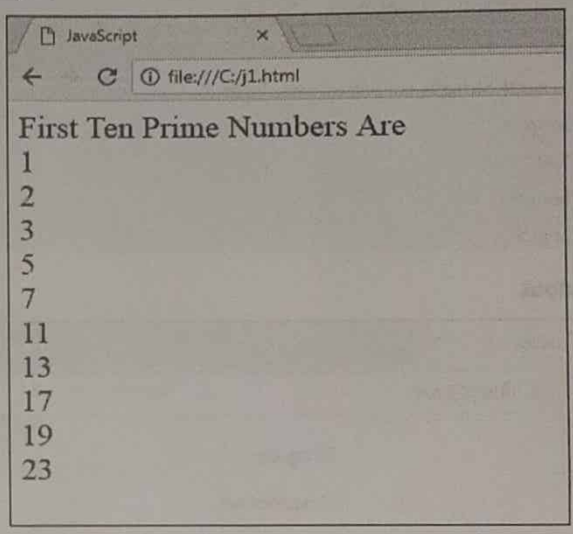


Program 2.4.8 : Write a JavaScript to find first 10 prime numbers.

☑ **Soln. : JavaScript to find first 10 prime numbers**

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

<body>
<font size=5>
<script language="JavaScript">
  document.write("First Ten Prime Numbers Are<br>");
var n,flag,i,cnt;
cnt = 1;
n = 1;

while(1)
{
    i = 2;
    flag = 0;

while(i<n)
{
    if(n%i==0)
{
    flag = 1;
    break;
}
i++;
}
if(flag==0)
{
    document.write(n + "<br>")
    cnt++;
}
if(cnt>10)
break;
n++;
}
</script>
</font>
</body>
</html>
```

Output



Program 2.4.9 : Write a JavaScript to check input string is palindrome or not.

☑ **Soln. :**

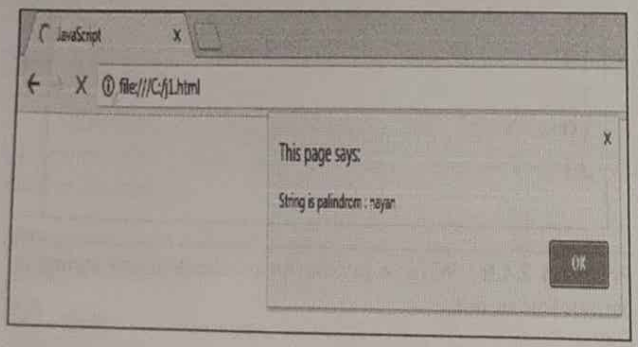
JavaScript to check string is palindrome or not

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

<body>
<font size=5>

<script language="JavaScript">
```

Unit II In Sem.

```
var str = prompt("Enter a string : ");
var i,flag;
flag = 0;

i = 0;
while(i<str.length/2)
{
    if(str.charAt(i) != str.charAt(str.length-i-1))
    {
        flag = 1;
        break;
    }
i++;
}
if(flag==0)
    alert("String is palindrome : "+str);
else
    alert("String is not palindrome: "+str);
</script>
</font>
</body>
</html>
```

Output



(B) do while Loop

Syntax

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

Working

The do while is an exit controlled loop. That means the first execution of loop statement is done without checking any condition. From second execution the condition get checked. Hence even if the condition is not satisfying then also the loop statements get executed once.

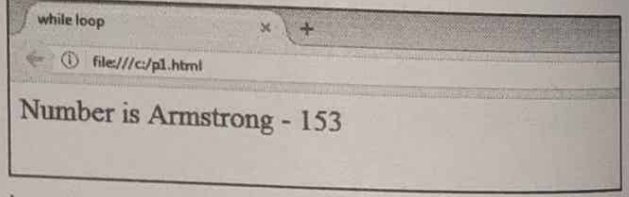
Program 2.4.10 : Write a program to accept a number from user and check whether it is Armstrong number or not. (**Hint :** Armstrong number means the summation of cubes of all the digits of the number should be exactly to the number). E.g. 153 = (1*1*1) + (5*5*5) + (3*3*3)

☑ Soln. : Program to check the Armstrong number

```
<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>
```

Note : Math.floor() This method returns the integer less that the number.

Output



(C) for Loop

Syntax

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

Explanation

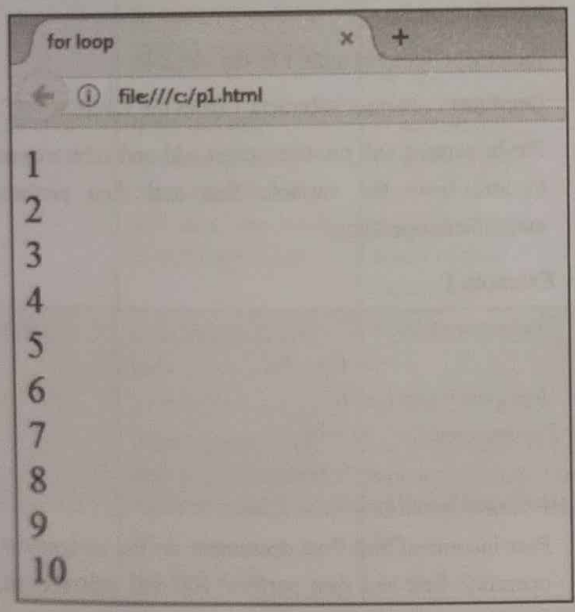
- In for loop the initialization, condition and increment or decrement of loop variable is done in a single statement.
- For loop helps to minimize the code.

Program 2.4.11 : Write a program to print 1 to 10 numbers using for loop.

Soln. : 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



(D) Nested for loop

Program 2.4.12 : Write a program to print the following pattern.

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

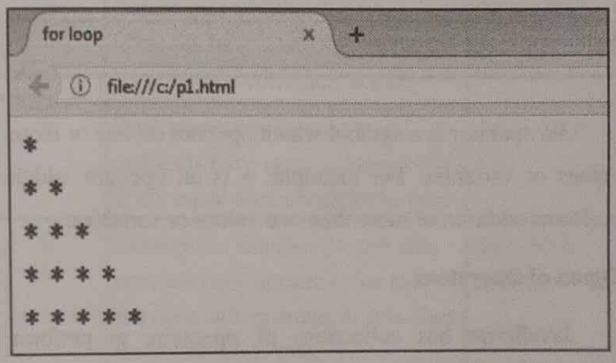
Soln. :

Unit II In Sem.

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



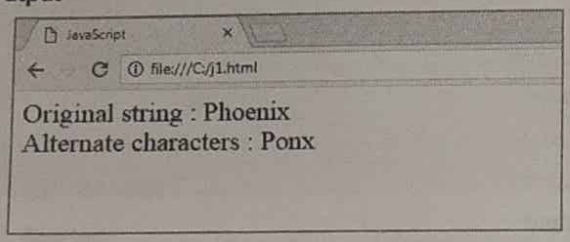
Program 2.4.13 : Write a JavaScript to print characters of a string at odd positions. (For example : for the string India, I, d and a should get printed).

☑ **Soln. :**

JavaScript to print characters of a string at odd positions

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var str = prompt('Enter String ');
var mychar, value;
document.write("Original string : "+str);
document.write("<br>Alternate characters : ");
for (var i=1; i<=str.length; i++)
{
 if (i % 2 == 1)
    {
        document.write(str.charAt(i-1));
    }
}

</script>
</font>
</body>
</html>
```

Output



2.5 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.

☞ **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.

Unit II In Sem.

Operator	Description	Example
<	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.	5&1 // returns 1 0101&0001 // returns 1
|	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.	5|1 // returns 5 0101 | 0001 // returns 5 in decimal or 65 in octal
^	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.	5^1 // returns 4 0101^0001 // returns 4 in decimal or 64 in octal
~	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.	~5 // output 6 ~0101 // output 66 in octal
<<	Binary Left Shift Operator: The bits of the left operand are moved left by the number of bits specified by the right operand.	5<<1 // returns 10 0101<<1 // returns 10 in decimal and 130 in octal
>>	Binary Right Shift operator: The bits of the left operand are moved right by the number of bits specified by the right operand.	5>>1 // returns 2 0101>>1 // returns 2
>>>	Unsigned Right Shift operator: Shifts right by pushing zeros in from the left, and let rightmost bit falls off	5>>>1 = 2

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's consider two variables X=10 and Y=20.

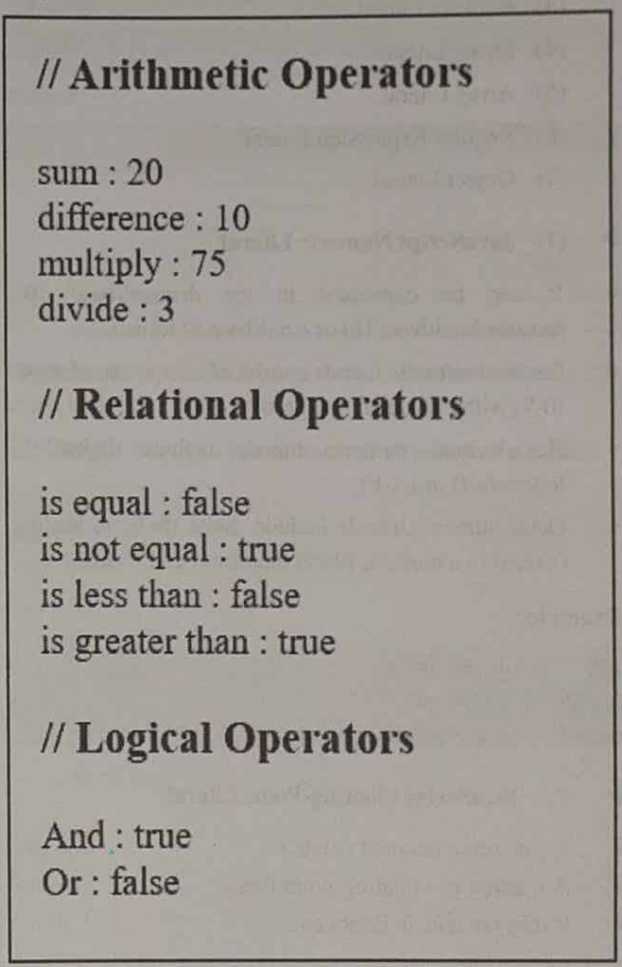
MAX = (X>Y) ?X : Y

- 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.

Program 2.5.1 : JavaScript Operators

```
<!doctype html>
    <head>
        <title>JavaScript Operators</title>
    </head>
    <body>
<script>
    document.write("<h3>
// Arithmetic Operators</h3>")
    var x = 15 + 5;
    document.write(" sum : "+x)

    x = 15 - 5;
    document.write("<br>difference : "+x)

    x = 15 * 5;
    document.write("<br>multiply : "+x)

    x = 15 / 5;
    document.write("<br>divide : "+x)

document.write("<h3>// Relational Operators</h3>")
    var x = 15 == 5;
    document.write(" is equal : "+x)

    x = 15 != 5;
    document.write("<br>is not equal : "+x)

    x = 15 < 5;
    document.write("<br>is less than : "+x)

    x = 15 > 5;
    document.write("<br> is greater than : "+x)

    document.write("<h3>// Logical Operators</h3>")
    var x = 15 > 5 && 5 < 15 ;
    document.write(" And : "+x)

    x = 15 ==5 || 5 == 15;
    document.write("<br>Or : "+x)

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

Output



Unit II In Sem.

2.6 LITERALS

- JavaScript Literals are the fixed value that cannot be changed, you do not need to specify any type of keyword to write literals. Literals are often used to initialize variables in programming, names of variables are string literals.
- A JavaScript Literal can be a numeric, string, floating-point value, a boolean value or even an object. In simple words, any value is literal, if you write a string "SJCEM" is a literal, any number like 220590 is a literal, etc.
- JavaScript supports various types of literals which are listed below :
  (1) Numeric Literal
  (2) Floating-Point Literal
  (3) Boolean Literal
  (4) String Literal
  (5) Array Literal
  (6) Regular Expression Literal
  (7) Object Literal

(1) JavaScript Numeric Literal

- It can be expressed in the decimal(base 10), hexadecimal(base 16) or octal(base 8) format.
- Decimal numeric literals consist of a sequence of digits (0-9) without a leading 0(zero).
- Hexadecimal numeric literals include digits(0-9), letters (a-f) or (A-F).
- Octal numeric literals include digits (0-7). A leading 0(zero) in a numeric literal indicates octal format.

Example :

```
120       // decimal literal
021434    // octal literal
0x4567    // hexadecimal literal
```

(2) JavaScript Floating-Point Literal

- It contains a decimal point(.)
- A fraction is a floating-point literal
- It may contain an Exponent.

Example :

```
6.99689       // floating-point literal
-167.39894    // negative floating-point literal
```

3. JavaScript Boolean Literal

Boolean literal supports two values only either true or false.

Example :

```
true      // Boolean literal
false     // Boolean literal
```

4. JavaScript String Literal

A string literal is a combination of zero or more characters enclosed within a single(') or double quotation marks (").

Example :

```
"Shraddha"      // String literal
'Sayali'        // String literal
```

String literals can have some special characters too which are listed below :

String Special Characters :

- \b It represents a backspace.
- \f It represents a Form Feed.
- \n It represents a new line.
- \r It represents a carriage return.
- \t It represents a tab.
- \v It represents a vertical tab.
- \' It represents an apostrophe or single quote.
- \" It represents a double quote.
- \\ It represents a backslash character.
- \uXXXX It represents a Unicode character specified by a four-digit hexadecimal number.

5. JavaScript Array Literal

- An array literal is a list of zero or more expressions representing array elements that are enclosed in a square bracket([]).
- Whenever you create an array using an array literal, it is initialized with the elements specified in the square bracket.

Example :

```
var emp = ["Sumita","Sudha","Manisha"];  // Array literal
```

6. JavaScript Regular Expression Literal

- Regular Expression is a pattern, used to match a character or string in some text.
- It is created by enclosing the regular expression string between forward slashes.

Example :

```
var myregexp = /ab+c/; // Regular Expression literal
var myregexp = new RegExp("abc"); // Regular Expression literal
```

7. JavaScript Object Literal

- It is a collection of key-value pairs enclosed in curly braces({}).
- The key-value pair is separated by a comma.

Example :

```
var games = {cricket :11, chess :2, carom: 4}  // Object literal
```

2.7 FUNCTIONS

GQ. Explain user defined function in JavaScript with suitable example. **(10 Marks)**

- **Definition : function** is a block of statements related to such a task which we want to execute repeatedly in our program.
- 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.
- In previous sections we have seen the built in functions like alert() and write().
- We can create our own functions known as user defined functions.
- **Definition :** Before using a function, we have to define it. Usually 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 2.7.1 : Write a simple function calling program.

Soln. :

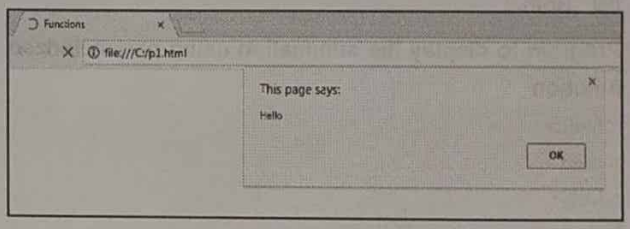
Program to display hello message using function call

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

Function Definition

Function Call

Output



Program 2.7.2 : Write a program to call a function on click event of button.

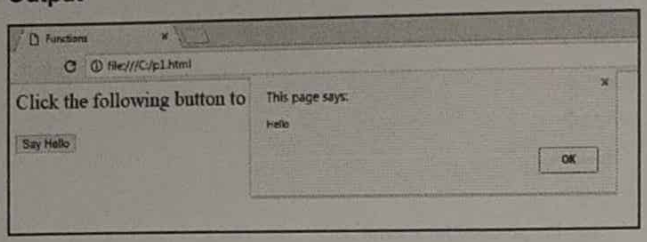
Soln. : Calling a function on click event of a button

```
<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">
```

Unit II In Sem.

```
</form>
</font>
</body>
</html>
```

Output



A) Parameterized Function

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.

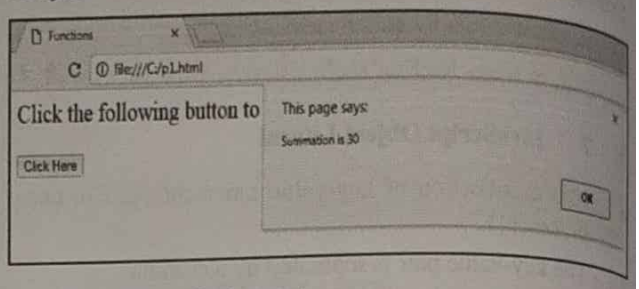
Program 2.7.3 : Write a program to display the summation of two values using parameterized function.

☑ **Soln. :**

Program to display the summation 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



B) The Return Statement

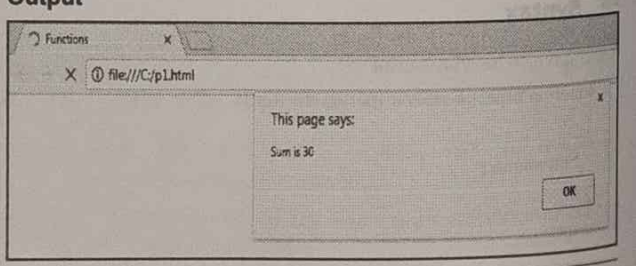
- In JavaScript function we can have **return** statement which is optional. This helps to return a value from a function.
- This statement is written at the last in a function.
- The returned value goes to the location where the function is called.

Program 2.7.4 : Write a program which will accept two numbers as arguments and returns their summation.

☑ **Soln. : Program to display the summation accepting two numbers as arguments**

```
<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



2.8 OBJECTS IN JAVASCRIPT

GQ. What are JavaScript objects? List the important built-in objects. How can you write your own object? **(10 Marks)**

Definition : An **object** is nothing but an entity having its own state and behavior (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.

Examples of objects

- Following are some of the examples of objects in JavaScript.
  (i) Booleans (when defined with the new keyword)
  (ii) Numbers (when defined with the new keyword)
  (iii) Strings (when defined with the new keyword)
  (iv) Dates
  (v) Regular expressions
  (vi) Arrays
  (vii) Functions
- We can create our own user defined objects in JavaScript. JavaScript is basically a template based scripting language not class based. Hence, we directly create the object without class.

2.8.1 Creating Objects in JavaScript

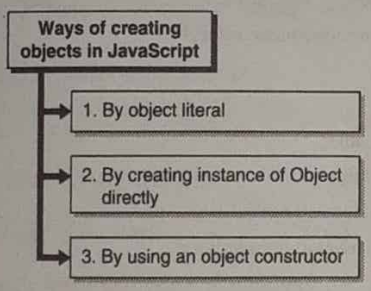


Fig. 2.8.1 : Ways of creating objects

1. JavaScript Object by Object Literal

Syntax

Following is the syntax of creating object using object literal :

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

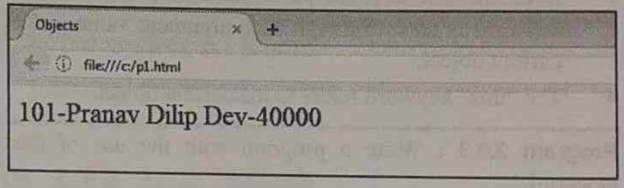
As we can observe, the property and value is separated by the separator : (colon).

Program 2.8.1 : Write a sample program of creating object using object literal.

Soln. : Program of creating object using object literal

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
employee = {id:101,name:"Pranav Dilip Dev",salary:40000}
document.write(employee.id+"-"+employee.name+"-" +employee
.salary);
</script>
</font>
</body>
</html>
```

Output



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 2.8.2 : Write a simple program of creating the instance of object.

Soln. :

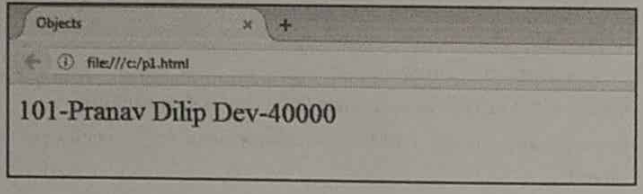
Program of creating the instance of object

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
```

Unit II In Sem.

```
var employee=new Object();
employee.id=101;
employee.name="Pranav Dilip Dev";
employee.salary=40000;
document.write(employee.id+"-"+employee.name+"-"
+employee.salary);
</script>
</font>
</body>
</html>
```

Output



3. By using an Object constructor

GQ. How a constructor can be used to populate data in the object ? **(5 Marks)**

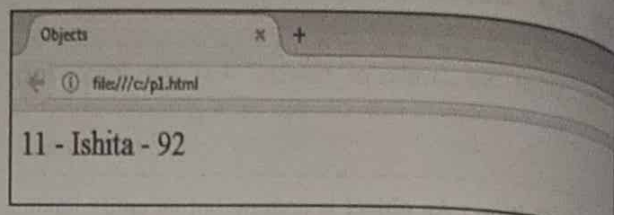
- Here, we have to create parameterized function. **"this"** keyword is used to assign each argument value in the current object.
- The "this" keyword refers to the current object.

Program 2.8.3 : Write a program with the use of this keyword.

Soln. : 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(11,"Ishita",92);
document.write(s.id+" - "+s.sname+" - "+s.marks);
</script>
</font>
</body>
</html>
```

Output



2.8.2 Defining Method in JavaScript Object

GQ. How to define method in JavaScript Object ? **(5 Marks)**

It is possible to define method in JavaScript object. For this purpose we have to add property in the method with same name as of the method name.

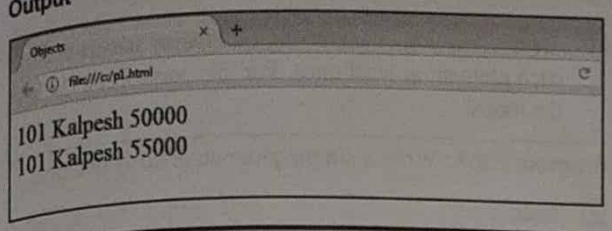
Program 2.8.4 : Write a program to define the method in JavaScript.

Soln. :

Program to define the method in JavaScript

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
function employee(id,sname,salary)
{
this.id=id;
this.sname=sname;
this.salary=salary;
this.changeSalary=changeSalary;
 function changeSalary(otherSalary)
  {
  this.salary=otherSalary;
  }
}
e = new employee(101,"Kalpesh",50000);
document.write(e.id+" "+e.sname+" "+e.salary);
e.changeSalary(55000);
document.write("<br>"+e.id+" "+e.sname+" "+e.salary);
</script>
</font>
</body>
</html>
```

Output



2.9 ARRAYS

UQ. How to create arrays in JavaScript?
SPPU - May 15, 8 Marks

Definition : An array is a group of elements of same data type. All the elements in array have index numbers which starts from zero. These index numbers are used to access the specific array element.

Syntax

```
var students = new Array( "Kunal", "Ishita", "Shravi" , "Shrey");
```

- The array is created using new keyword. The maximum size allowed for an array is 4,294,967,295.
- You can create array by simply assigning values as follows –

```
var students = ["Kunal", "Ishita", "Shreya", "Shravi"];
```

The index numbers are used to access the array elements

```
students[0] is the first student – Kunal
students[1] is the second student – Ishita
```

2.9.1 Different Ways to Create an Array

1. Empty array without elements

```
var empty = [];
```

2. Array with 2 string elements

```
var days = ["Sunday", "Monday"];
```

3. Array with different types of elements

```
var mixed = [true, 100, "Hello"];
```

4. Two dimensional array with object literals

```
var arr = [[1,{x:10, y:20}], [2, {x:30, y:40}]];
```

5. The 3rd element is undefined

```
var colors = ["Red", "Blue", undefined];
```

6. No value in the 1st position, it is undefined

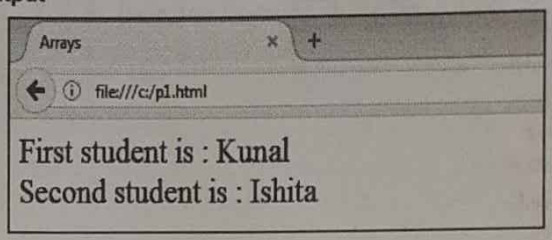
```
var hobbies = [,"Sports"];
```

Program 2.9.1 : Write a simple program of array displaying student name.

Soln. : Program of array displaying student name

```
<html>
<head>
<title>
Arrays
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var students = new Array( "Kunal", "Ishita", "Shravi" , "Shrey");
document.write("First student is : "+students[0]);
document.write("<br>Second student is : "+students[1]);
</script>
</font>
</body>
</html>
```

Output



2.9.2 Array Methods

Method	Description
Length	Returns length of an array (It is a property of array)
concat()	Concatenates (merge) multiple arrays.
every()	If all the elements in the array satisfy the given condition of testing function, it returns true.
filter()	Returns array elements which satisfy the filter criteria of given function.
forEach()	Calls a specific function for all the elements in the array.
indexOf()	Return the index of first occurrence of given element. Returns -1 if element not found.
join()	Joins all the elements of an array and converts into a string.
lastIndexOf()	Return the index of last occurrence of given element. It returns 1, if the specified element not found.
pop()	Removes and returns the last element of an array.

Unit II In Sem.

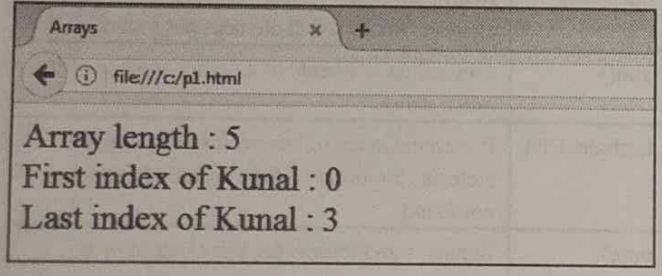
Method	Description
push()	Adds one or more elements at the end of an array.
reverse()	Reverses the sequence of the elements of an array.
shift()	Removes and returns the last element of an array.
slice()	Extracts a portion of an array and returns it.
sort()	Sorts the elements of an array
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representation of the given array
unshift()	Adds one or more elements at the beginning of an array and returns it.

Program 2.9.2 : Write a program to display the length of array and index of array element.

☑ **Soln. : Program to display the length of array and index of array element**

```
<html>
<head>
<title>
Arrays
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var students = ["Kunal", "Ishita", "Shravi" , "Kunal","Shrey"];
document.write("Array length : "+students.length);
document.write("<br>First index of Kunal :
"+students.indexOf("Kunal"));
document.write("<br>Last index of Kunal :
"+students.lastIndexOf("Kunal"));
</script>
</font>
</body>
</html>
```

Output



2.9.3 Iterating Through an Array

☞ **Use :** Iterating or traversing of an array means visiting each element at least once. For this purpose we can use the loops.

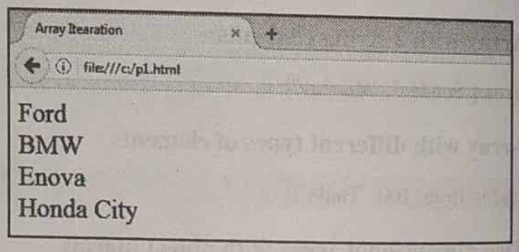
Program 2.9.3 : Write a simple program of array iteration.

☑ **Soln. :**

Program of array iteration

```
<html>
<head>
<title>
Array Iteration
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var cars = [];
cars[0] = "Ford";
cars[1] = "BMW";
cars[2] = "Enova";
cars[3] = "Honda City";
for (var i = 0; i < cars.length; i++)
{
    document.write(cars[i] + "<br>");
}
</script>
</font>
</body>
</html>
```

Output



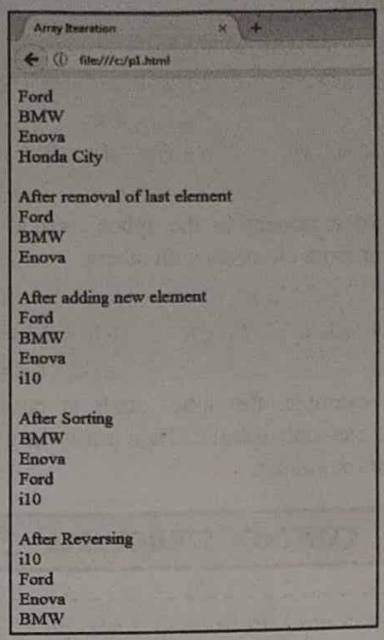
Program 2.9.4 : Write a program to perform various methods like adding a new element, sorting, reversing, removal of last element etc.

☑ **Soln.: Program to perform various methods on array**

```
<html>
<head>
```

```
<title>
Array Iteration
</title>
</head>
<body>
<font size=4>
<script language="JavaScript">
var cars = [];
cars[0] = "Ford";
cars[1] = "BMW";
cars[2] = "Enova";
cars[3] = "Honda City";
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.pop();
document.write("<br>After removal of last element<br>");
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.push("i10");
document.write("<br>After adding new element<br>");
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.sort();
document.write("<br>After Sorting<br>");
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.reverse();
document.write("<br>After Reversing<br>");
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
</script>
</font>
</body>
</html>
```

Output



Unit II In Sem.

2.9.4 Deleting Element from an Array

GQ. Explain how to delete an element from array in JavaScript. **(2 Marks)**

☞ **Use :** The delete operator is used to remove an element from an array. Deleting an element from an array does not affect the length property and the array becomes sparse. Also the elements which are at the right of the deleted element do not get shifted to left to fill in the gap.

☞ **Example**

```
var days = ["Sunday", "Monday", "Tuesday", "Wednesday"];
delete days[1];          // delete the element "Monday"
```

2.9.5 Array Method : Splice()

GQ. Explain array method splice() with suitable examples in JavaScript. **(2 Marks)**

☞ **splice() method**

- **Use :** The splice() method is used to insert new, delete existing, and replace existing elements by new elements in the array.
- It moves the elements to higher or lower positions as per the requirement to avoid any gap.
- The first argument of splice() indicates the starting position and second argument indicates the number of elements to delete.

Examples

```
var letters = ["a","b","c","d","e","f","g"];
alert(letters.splice(5, 2));        // f, g (deleted elements)
alert(letters);                     // a, b, c, d, e
alert(letters.splice(2, 1));        // c  (the deleted element)
alert(letters);                     // a, b, d, e
```

The third argument in the splice method is used to replace one or more elements with others.

```
var letters = ["a","b","c","d"];
alert(letters.splice(1, 2, "e", "f", "g"));      // b, c (deleted ones)
alert(letters);                                  // a, e, f, g, d
```

In this example, the splice starts at position 1 and removes two elements b and c. Then it fills the gap with the three elements e, f and g.

2.10 CONTROL STRUCTURES

GQ. Explain control structures in JavaScript. **(8 Marks)**

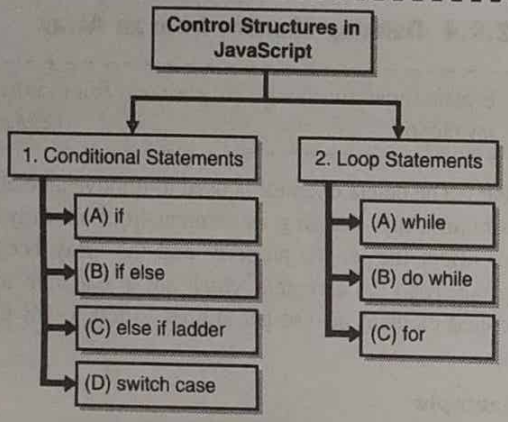


Fig. 2.10.1 : Control Structures in JavaScript

2.10.1 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

(A) The if Statement

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

Syntax

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

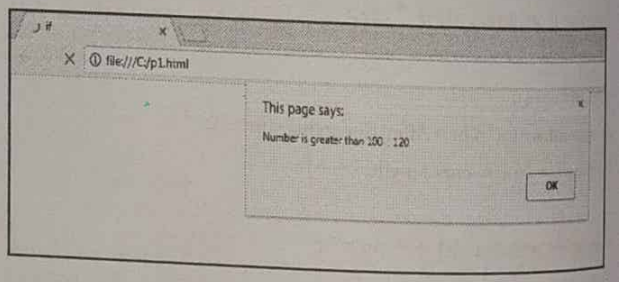
Program 2.10.1 : Write a program to accept a number and check whether it is greater than 100 or not.

Soln. :

Program to accept a number to 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



(B) The else Statement

Use : The 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;
}
```

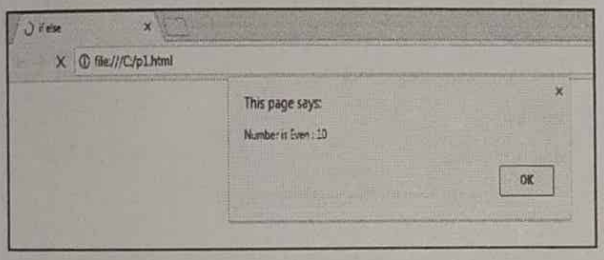
Program 2.10.2 : Write a program to accept a number from user and check whether it is even or odd.

Soln. :

Program to check number 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



(C) else if Ladder

Use : 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;
}
```

Program 2.10.3 : 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.)

Soln. :

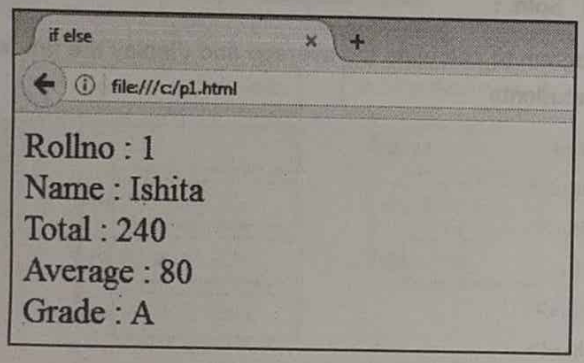
Program to calculate the average and display the grades of students

```
<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 Web : "));
DAA = parseInt(prompt("Enter marks of DAA : "));
SYS = parseInt(prompt("Enter marks of SysPro and OS : "));
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);
```

Unit II In Sem.

```
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



(D) Switch Case Statement

☞ **Use :** 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;
}
```

☞ **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.

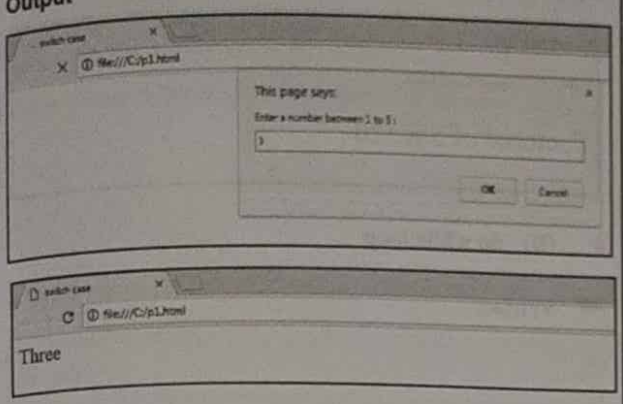
Program 2.10.4 : Write a program to print the numbers between 1 to 5.

☑ **Soln. :**

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



Program 2.10.5 : Write a program to display the name of weekday using switch case.

Hint : The getDay() method returns the number of weekday in the range from 0 and 6.

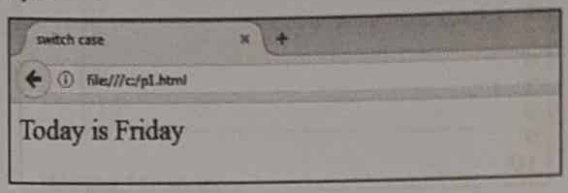
(For example Sunday = 0, Monday = 1, Tuesday = 2 ..)

☑ **Soln. :**

Program to display the name of day using switch case statement

```
<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



2.10.2 Loop Statements

It contains while, do while and for loop.

(A) while loop

Syntax

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

Working

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

Program 2.10.6 : Write a program to print 1 to 10 numbers using while loop.

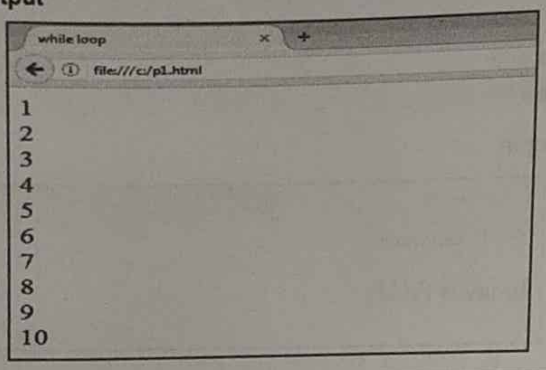
☑ **Soln.:**

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

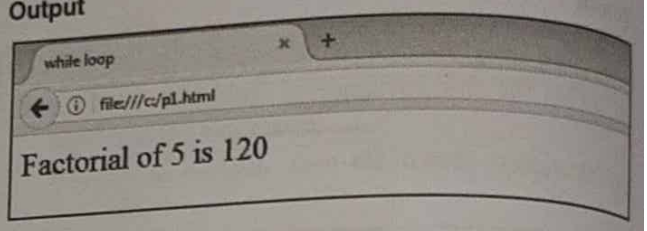


Program 2.10.7 : Write a program to accept a number from user and print factorial of it.

☑ **Soln. : Program to print factorial of a number**

```
<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



(B) do while loop

☞ **Syntax**

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

☞ **Working**

The do while is an exit controlled loop. That means the first execution of loop statement is done without checking any condition. From second execution the condition get checked. Hence even if the condition is not satisfying then also the loop statements get executed once.

Program 2.10.8 : Write a program to accept a number from user and check whether it is Armstrong number or not.

(Hint : Armstrong number means the summation of cubes of all the digits of the number should be exactly to the number)

E.g. 153 = (1*1*1) + (5*5*5) + (3*3*3)

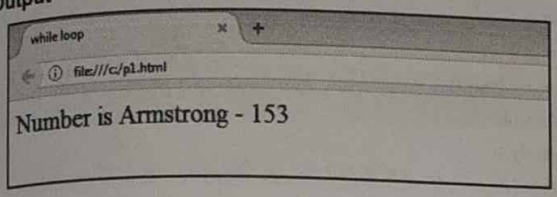
☑ **Soln. : Program to check the Armstrong number**

```
<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>
```

Note : Math.floor() This method returns the integer less that the number.

Output



(C) for loop

Syntax

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

Explanation

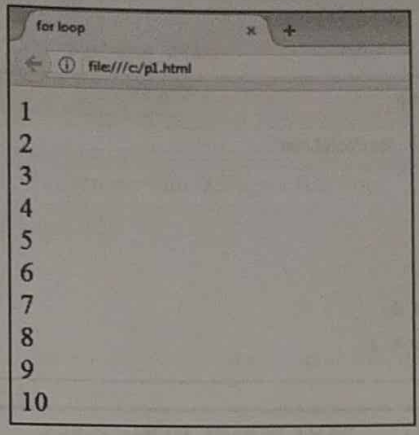
In for loop the initialization, condition and increment or decrement of loop variable is done in a single statement. For loop helps to minimize the code.

Program 2.10.9 : Write a program to print 1 to 10 numbers using for loop.

Soln. : 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

Program 2.10.10 : Write a program to print the following pattern.

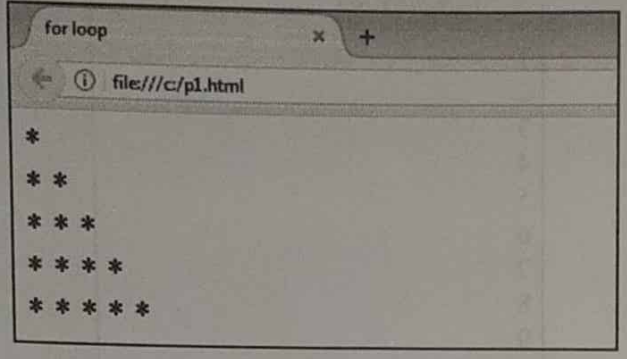
```
*
* *
* * *
* * * *
* * * * *
```

Soln. : 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



2.11 JAVASCRIPT DEBUGGERS

- It is common to have errors while writing codes and the errors can be due to syntax or logical. These errors create a lot of ambiguity in the logic and understanding of both users and programmers.
- There can also be errors in the code which can remain invisible to the programmer's eye and can create havoc. To identify these errors we need Debuggers that can go through the entire code or program, identify the errors and also fix them.
- Generally, we can open the built-in JavaScript debugging with F12 Key to display the JavaScript values by using "console".

By using console.log () Method

We can use the console.log() method for displaying the JavaScript values only if the browser supports debugging.

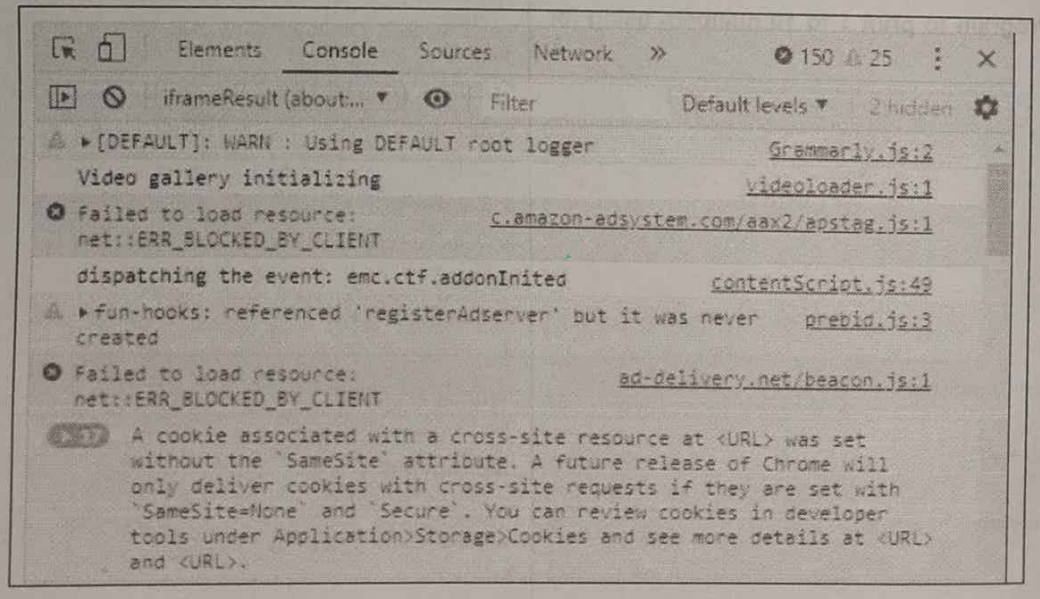
Example

```
<!DOCTYPE html>
<html>
<body>
<h1>This is my first web page</h1>
<script>
x = 5;
y = 6;
z = x + y;
console.log(c);
</script>
</body>
</html>
```

By using the Debugger Keyword

- This debugger keyword is used in browsers like Chrome, Firefox, Internet Explorer, etc.
- Wherever we predict errors occur, there put a debugger.
- While executing the code, instead of executing the entire code, the compiler pauses the execution at the debugger.
- With this, we can step by step analyze where the error occurs exactly.

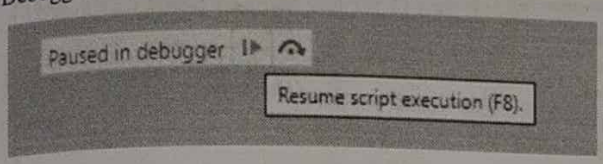
Console Tab :



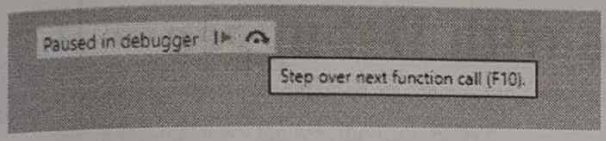
☞ **Syntax :**

```
var a=10;
debugger;
for(let i=0;i<=10;i++)
{
//code
}
```

Debugger mode looks like below :



- When we used the debugger keyword in our code, if we press the F12 button and Run the code, we will get debugger mode as above.
- The first blue color button is used to Resume (come out of the debug) the code.

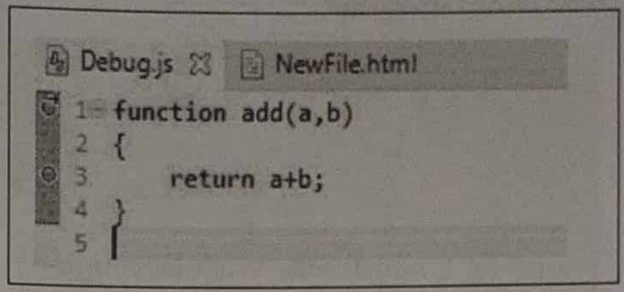


- The second black color button is used to move step by step line execution of code.

☞ **By using Breakpoints**

- These breakpoints are used in IDE's instead of using the debugger keyword.
- Wherever we predict errors occurs there put breakpoint.
- While executing the code, instead of executing the entire code, the compiler pauses the execution at the debugger.
- With this, we can step by step analyze where the error occurs exactly.

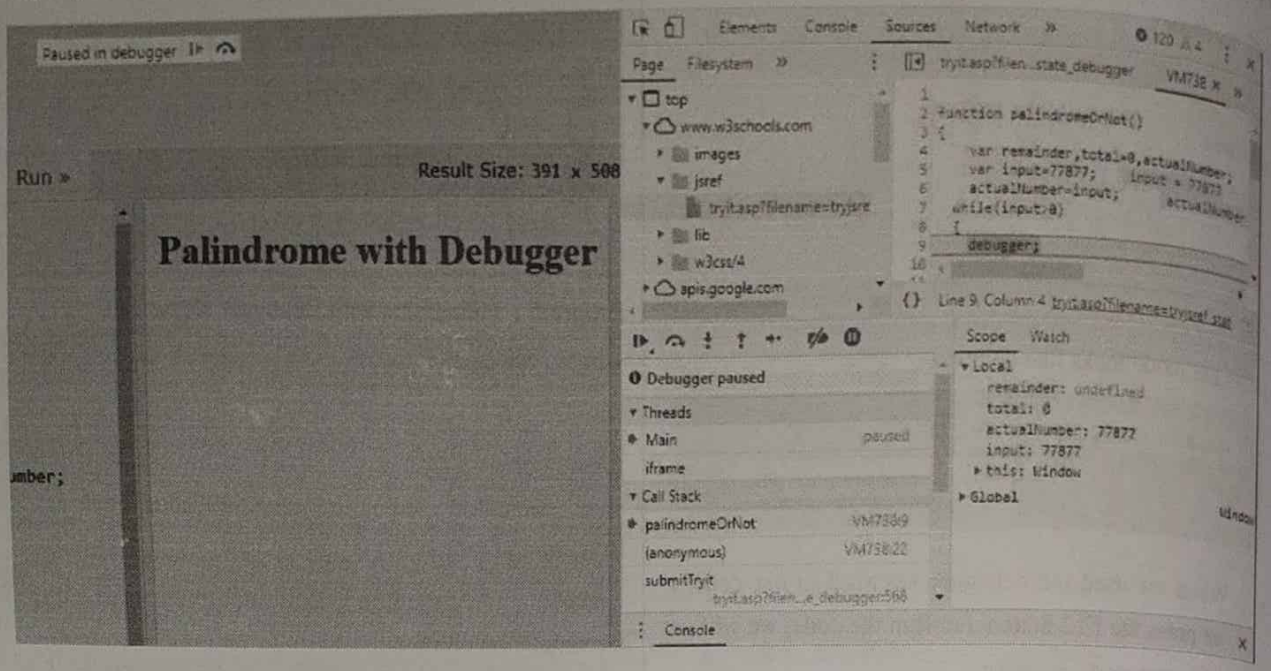
Example : In Eclipse IDE breakpoints look like below:



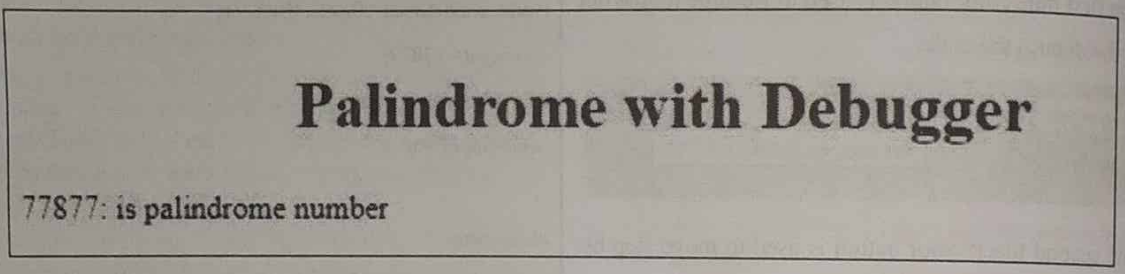
Program : Palindrome with Debugger Keyword

```
<!DOCTYPE html>
<html>
<body>
<font color="blue"><h1 align="center">Palindrome with
Debugger</h1></font>
<script>
functionpalindromeOrNot()
{
varremainder,total=0,actualNumber;
var input=77877;
actualNumber=input;
while(input>0)
{
debugger;
remainder=input%10;
total=(total*10)+remainder;
input=parseInt(input/10);
}
if(actualNumber===total){
document.write(actualNumber+": is palindrome number ");
}
else {
document.write(actualNumber+": is not palindrome");
}
}
palindromeOrNot();
</script>
</body>
</html>
```

Output paused at debugger



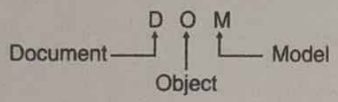
Final Output



- In the above code, when we press F12 and run the code then code moved to debugger mode as above.
- Once we click on step over the next function call or F10 then code moved from debugger to next line, keep on pressing until we figure out any errors.
- If no errors encountered, then code automatically comes out of the debug mode prints the output.

2.12 DOM

GQ. Explain DOM in JavaScript. **(8 Marks)**



☞ **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**

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.

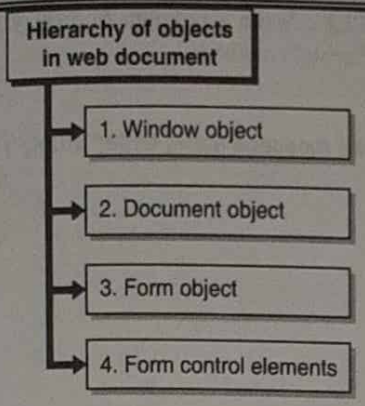


Fig. 2.12.1 : Hierarchy of objects in web document

1. Window object

It resides at top of the hierarchy. It is the topmost element of the object hierarchy.

2. Document object

All the HTML documents which get loaded into browser are considered as document objects. The contents of the page are stored in the document object.

3. Form object

Everything which is contained in the opening <form> and closing</form> tag sets the form object.

4. Form control elements

The form object has all the elements which are defined for the objects like text fields, buttons, checkboxes, select box, radio buttons etc.

2.12.1 DOM Levels

The DOM provides all the **features** to JavaScript to create 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.

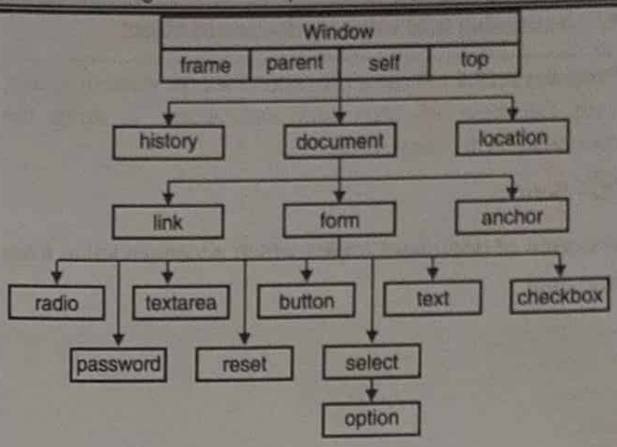


Fig. 2.12.2 : DOM levels

2.12.2 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.

Unit II In Sem.

1. Accessing field value by document object

Program 2.12.1 : Write a program to accept value from user with the help of text field and access it using the document.form1.name.value.

☑ **Soln. :**

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**

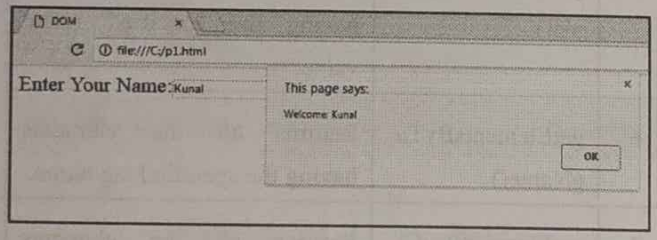
Here, **document** represents the html document. It is the root element.

form1 - name of the form.

name - 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.

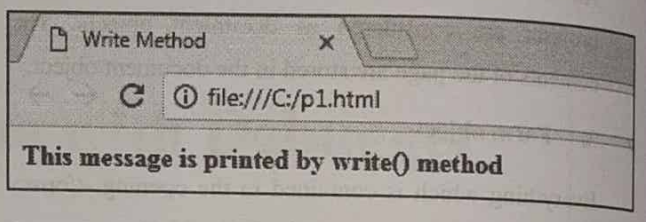
Program 2.12.2 : Write a program to display the message using write("string") method.

☑ **Soln. :**

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 2.12.3 : Write a program to demonstrate the use of writeln("String") method.

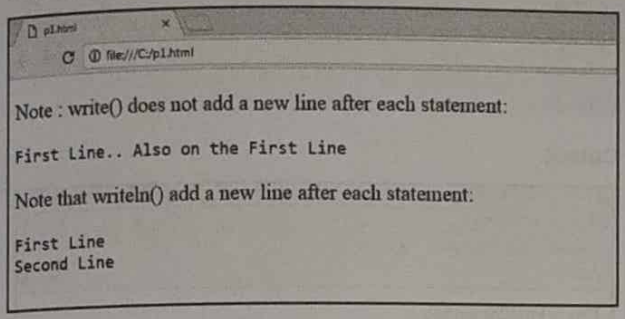
☑ **Soln. :**

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.

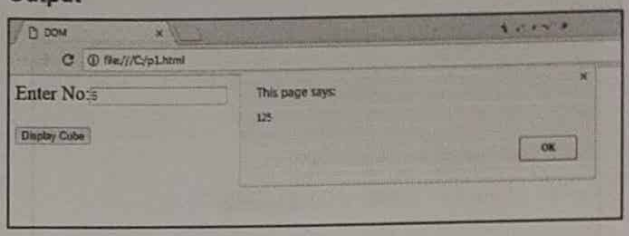
For the same purpose now we will use Document.getElementById() method. Here we have to define id for the input text field.

Program 2.12.4 : Display the cube of number using Document.getElementById() method.

☑ Soln. : 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



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 2.12.5 : Write a program to display number of languages available using radio buttons. Display the count of languages on click of a button.

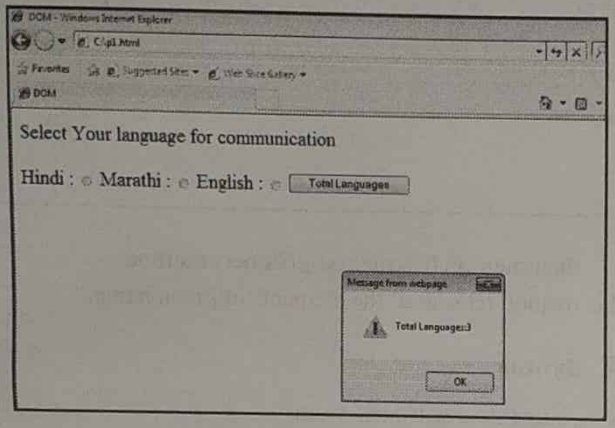
☑ Soln. : Program to display number of languages and their count on click event of button

```
<html>
<head>
<title>
DOM
</title>
<script type="text/javascript">
function languages()
{
var lan = document.getElementsByName("lang");
alert("Total Languages:"+lan.length);
}
</script>
</head>
```

Unit II In Sem.

```
<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")
```

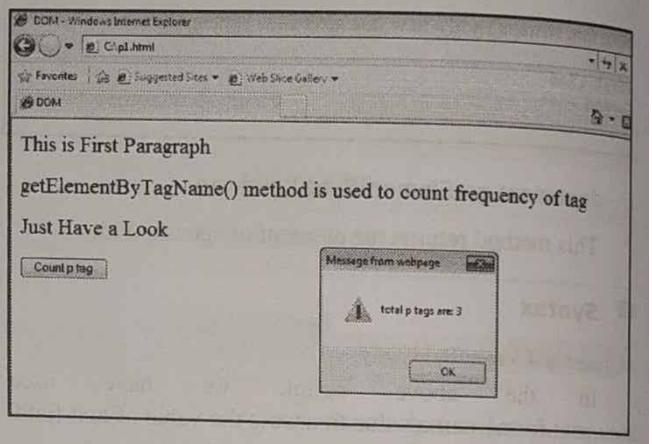
Program 2.12.6 : Write a program to count total number of tags used by document.getElementsByTagName() method.

Soln. :

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 2.12.7 : Write a program to demonstrate the use of getElementsByClassName() method.

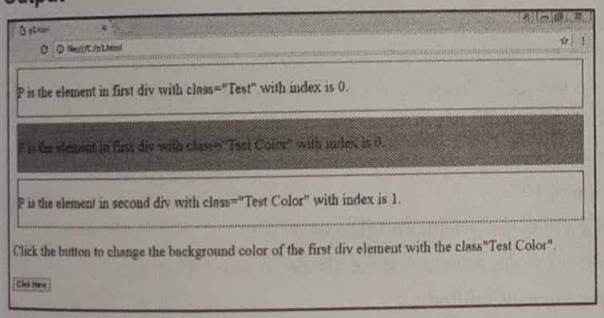
Soln. :

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



2.12.3 JavaScript - innerHTML

GQ. Explain innerHTML in JavaScript with the help of example (program). **(2 Marks)**

The dynamic html contents can be written in html document with the help of **innerHTML** property.

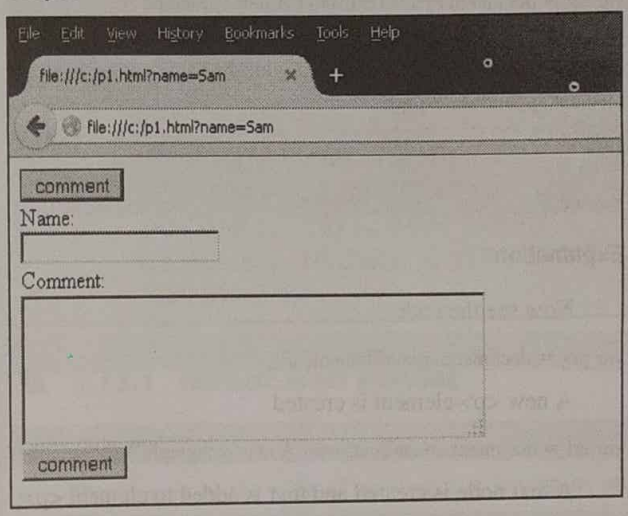
It is generally used in the html documents to generate the dynamic contents like registration form, comment form, links etc.

Program 2.12.8 : Write a program to generate comment form using innerHTML property.

☑ Soln. : Program of comment form using innerHTML property

```
<html>
<body>
<script type="text/javascript" >
function disp() {
var data="Name:<br><input type='text'
name='name'><br>Comment:<br><textarea rows='5'
cols='40'></textarea><br><input type='submit'
value='comment'>";
document.getElementById('myloc').innerHTML=data;
}
</script>
<form name="myForm">
<input type="button" value="comment" onclick="disp()">
<div id="myloc"></div>
</form>
</body>
</html>
```

Output



2.12.4 Manipulating DOM

GQ. Write a note on DOM manipulation. **(4 Marks)**

Using JavaScript we can add or remove nodes (HTML elements) in the document.

1. Creating New HTML Elements (Nodes)

Initially we have to create the element (element node) which can be then appended to the existing element.

Program 2.12.9 : Write a program to append the new element to an existing element node.

☑ **Soln. :**

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>
<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 :

Now 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>

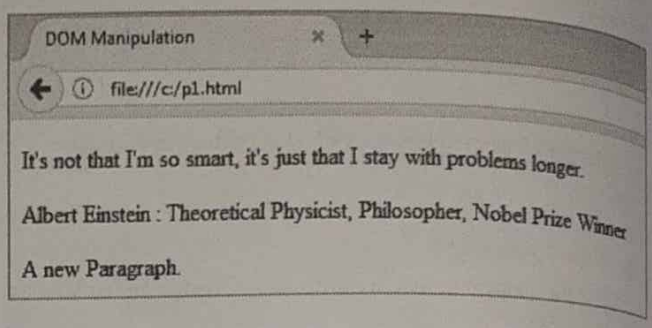
```
var element = document.getElementById("div1");
```

This code finds an existing element:

```
element.appendChild(prg);
```

This code appends the new element to an existing element.

Output



2. Creating new HTML Elements - insertBefore()

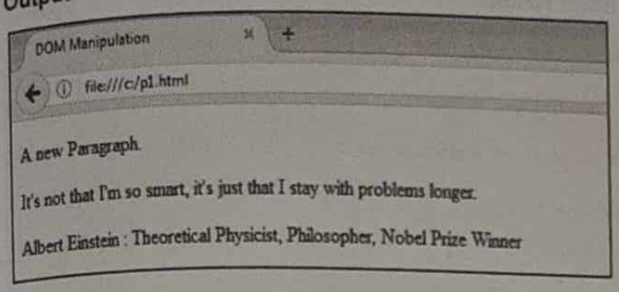
- In the previous program, we have seen the appendChild() method which appends the element at the end in parent.
- the insertBefore() method is used to add the new element at the beginning.

Program 2.12.10 : Write a program to add new element at the beginning using insertBefore() method.

☑ **Soln. : 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



3. Removing Existing HTML Elements

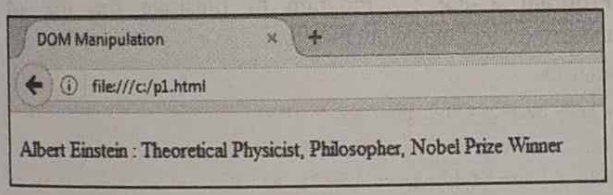
To remove an HTML element, we should have the information about the parent element. The removeChild() method is used to remove the HTML element.

Program 2.12.11 : Write a program to remove an existing element using removeChild() method.

☑ **Soln. : 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



4. Replacing HTML Elements

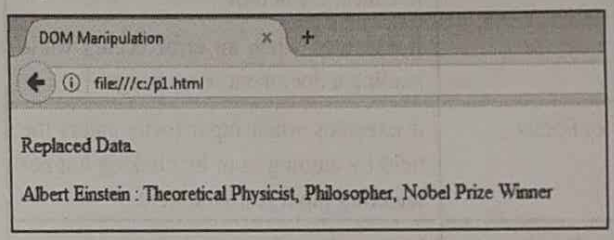
It is also possible to replace an existing element by new one. The replaceChild() method is used for this purpose.

Program 2.12.12 : Write a program to replace an existing element by new element using replaceChild() method.

☑ **Soln. : 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



Unit II In Sem.

2.13 INTRINSIC EVENT HANDLING, MODIFYING ELEMENT STYLE, THE DOCUMENT TREE, DOM EVENT HANDLING, JQUERY, OVERVIEW OF ANGULAR JS

2.13.1 Intrinsic event handling

Definition of Event

- The change in the state of an object is known as an Event.
- Events are the actions performed by the end users while browsing the website.

- For example mouse moves, mouse clicks, pressing a particular key of keyboard represent the events. Such events are called an intrinsic events.

Definition of Event Handler

- Event handler is a script that gets executed in response to these events.
- This event handler enables the web document to respond the user activities through the browser window.

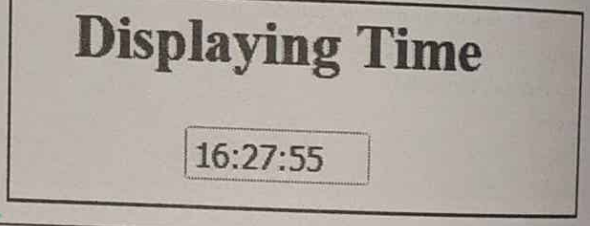
Event Handlers

Event Handler	Description
onAbort	It executes when the user aborts loading an image.
onBlur	It executes when the input focus leaves the field of a text, textarea or a select option.
onChange	It executes when the input focus exits the field after the user modifies its text.
onClick	In this, a function is called when an object in a button is clicked, a link is pushed, a checkbox is checked or an image map is selected. It can return false to cancel the action.
onError	It executes when an error occurs while loading a document or an image.
onFocus	It executes when input focus enters the field by tabbing in or by clicking but not selecting input from the field.
onLoad	It executes when a window or image finishes loading.
onMouseOver	The JavaScript code is called when the mouse is placed over a specific link or an object.
onMouseOut	The JavaScript code is called when the mouse leaves a specific link or an object.
onReset	It executes when the user resets a form by clicking on the reset button.
onSelect	It executes when the user selects some of the text within a text or textarea field.
onSubmit	It calls when the form is submitted.

Program 2.13.1 : Program to illustrate the use ofonload Event handler

```
<html>
<head>
<script type="text/javascript">
function time()
  {
var d = new Date();
varty = d.getHours() + ":"+d.getMinutes()+":"+d.getSeconds();
document.frmty.timetxt.value=ty;
setInterval("time()",1000)
  }
</script>
</head>
<body onload="time()">
<center><h2>Displaying Time</h2>
<form name="frmty">
<input type=text name=timetxt size="8">
</form>
</center>
</body>
</html>
```

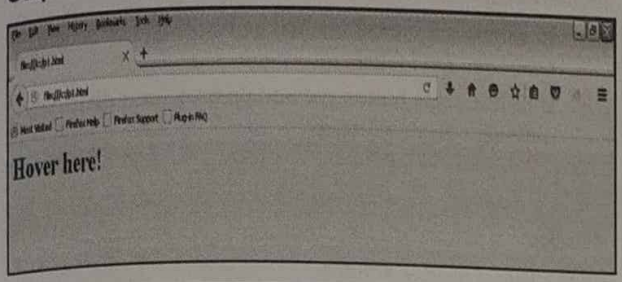
Output :



Program 2.13.2 : 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>
```

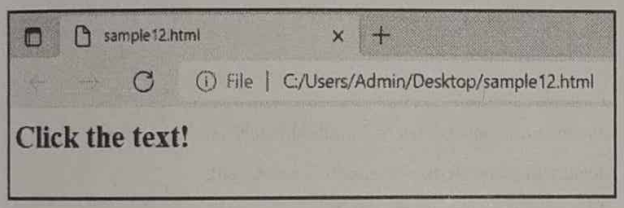
Output



Program 2.13.3 : 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



Program 2.13.4 : Program illustrating onsubmit() &onfocus() Event handler

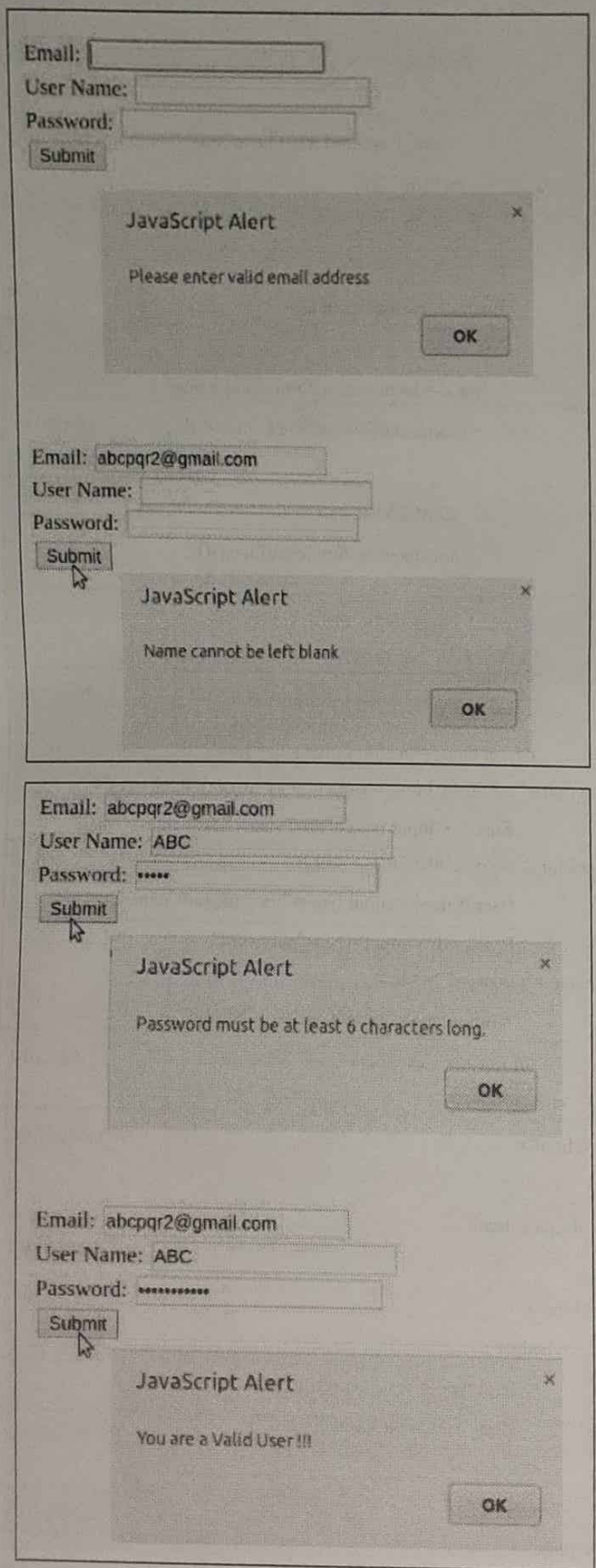
```
<html>
  <body>
    <script>
      function validateform()
      {
        varuname=document.myform.name.value;
        varupassword=document.myform.password.value;
        if (uname==null || uname=="")
        {
          alert("Name cannot be left blank");
          return false;
        }
        else if(upassword.length<6)
        {
          alert("Password must be at least 6 characters long.");
          return false;
        }
      }
      function emailvalidation()
      {
        var a=document.myform.email.value
        if (a.indexOf("@")==-1)
        {
          alert("Please enter valid email address")
          document.myform.email.focus()
        }
      }
    </script>
  <body>
    <form name="myform" method="post"
action="validpage.html" onsubmit="return validateform()">
      Email: <input type="text" size="20" name="email"
onblur="emailvalidation()"><br>
      User Name: <input type="text" name="name"><br>
      Password: <input type="password"
name="password"><br>
      <input type="submit" value="Submit" >
    </form>
  </body>
</html>
```

validpage.html

```
<html>
  <body>
    <script type="text/javascript">
      alert("You are a Valid User !!!");
    </script>
  </body>
</html>
```

Unit II In Sem.

Output



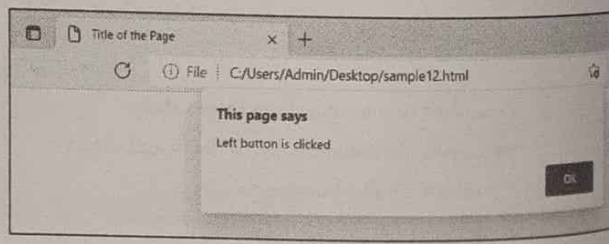
Program 2.13.5 : Write JavaScript that handles following mouse event.

(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".

```
<html>
<head>
<title>Title of the Page</title>
</head>
<body>
<script language="JavaScript">
varsTestEventType='mousedown';
functionhandleMouseEvent(e)
{
    varevt = (e==null ? event:e);
    varclickType = '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>
```

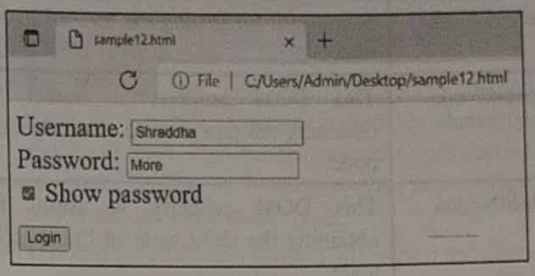
Output



Program 2.13.6 : Design HTML form which include two fields username and password. Write JavaScript code to show and hide password.

```
<html>
<body>
<Script>
(function() {

    varPasswordToggler = 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);

    vartoggler = new PasswordToggler( checkbox, pwd );
    });
})();
</Script>
<form action="" method="post" id="login">
    <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



Unit II In Sem.

2.13.2 Modifying Element Style

We can access or change the contents of document by using various methods. Some commonly used properties and methods of DOM are as follows:

Methods

Method	Description
getElementById ()	Returns the element with the specified ID.
getElementsByTagName ()	Returns a list of nodes (set / node array) containing all elements with the specified tag name.
getElementsByClassName ()	Returns a list of all of the elements with the specified node contains the name of the class.
appendChild ()	To add a new child node to the specified node.
removeChild ()	Removes the child node.
replaceChild ()	Replace child nodes.
insertBefore ()	Insert a new child node in front of the specified child node.
createAttribute ()	Create attribute node.
createElement ()	Create an element node.
createTextNode ()	Create a text node.
getAttribute ()	Returns the specified attribute values.
setAttribute ()	To set or modify the specified value of the specified property.

Properties

Property	Description
innerHTML	It is useful for getting the text value of a node.
parentNode	This DOM property is useful for obtaining the parent node of the specific node.
childNodes	This DOM property is useful for obtaining the child node of the specific node.
attributes	This property is used to get the attribute node of the specific node.

2.13.3 Accessing Elements using DOM

- There are several ways by which we can access the element of the web document.
- To understand these methods of accessing we will consider one simple web document as follows :

```
<html>
<head>
<title> This is my web page</title>
</head>
<body>
<form name= "form1">
<input type= "text" name= "myinput"/>
</form>
</body>
</html>
```

Method 1

- Every XHTML document element is associated with some address. This is called DOM address.
- The document has the collection of forms and elements. Hence we can refer the text box element as

  VarDom_obj=document.forms[0].elements[0];
- But this is not the suitable method of addressing the elements. Because if we change the above script as

```
.....
<form name= "form1">
<input type= "button" name= "mybutton"/>
<input type= "text" name= "myinput"/>
</form>
.....
```

- Then index reference gets changed. Hence another approach of accessing the element is developed.

Method 2

- We can access the desired element from the web document using JavaScript method getElementById.
- The element access can be made as follows:

```
VarDom_Obj=document.getElementById("myinput");
```

- But if the element is in particular group, that means if there are certain elements on the form such as radio buttons or check boxes then they normally appear in the groups.
- Hence to access these elements we make use of its index. Consider the following code sample:

```
<form id="Food">
<input type= "checkbox" name= "vegetables" value=
"Spinach"/> Spinach
<input type= "checkbox" name= "vegetables" value=
"Fenugreek"/>Fenugreek
<input type= "checkbox" name= "vegetables" value=
"Cabbage"/>Cabbage
</form>
```

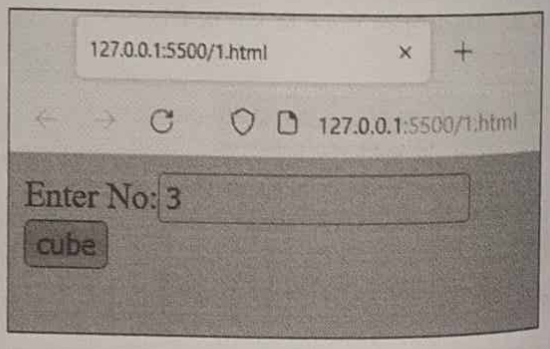
For getting the values of these checkboxes we can write following code:

```
VarDom_obj=document.getElementById("Food");
For (i=0; i<Dom_obj.vegetables.length; i++)
    document.write(Dom_obj.vegetables[i]+ "<br/>");
```

Program 2.13.7 : Example of document.getElementById() method that prints cube of the given number.

```
<script type="text/javascript">
functiongetcube(){
var number=document.getElementById("number").value;
alert(number*number*number);
}
</script>
<form>
Enter No:<input type="text" id="number"
name="number"/><br/>
<input type="button" value="cube" onclick="getcube()"/>
</form>
```

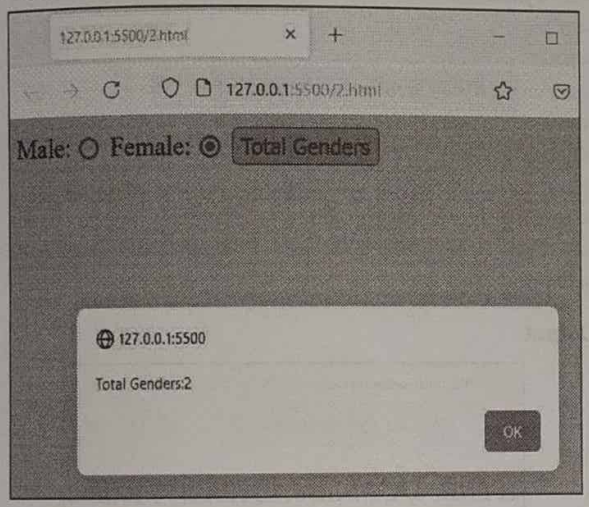
Output



Program 2.13.8 : Example of document.getElementsByName() method to get all the genders.

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

Output



2.13.4 Modifying Elements using DOM

Appending an element using DOM

- The appendChild() method of the node interface, is used to create a text node as the last child of the node. This method is also used to move an element from one element to another element.
- It is used for creating a new element with some text then first create the text as the text node and then append it to the element, then append the element to the document.

Syntax

```
node.appendChild(node);
```

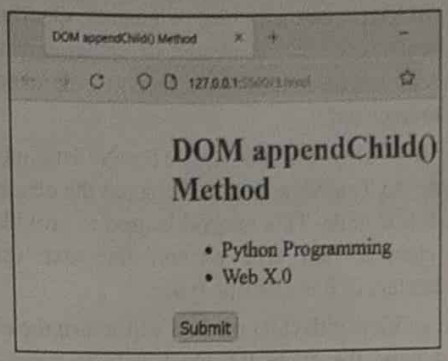
- This method accepts a single parameter node which is mandatory and used to specify the node object which needs to be appended.
- document.createElement() method that is used to create the HTML element ie., element specified with elementName will be created or an unknown HTML element will be created if the specified elementName is not recognized.
- We will also use the createTextNode() method to create the TextNode that will contain the element node and a text node. This method is used to provide text to an element that will contain the text values as parameters & it is of string type.
- The getElementById() method will return the elements containing the given ID which will be passed to the function.

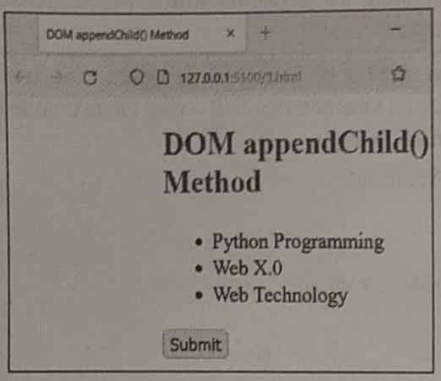
Unit II In Sem.

Program 2.13.9 : Example describes the use of the appendChild() method that will create the text node as the last child of the node.

```
<!DOCTYPE html>
<html>
<head>
<title>DOM appendChild() Method</title>
<style>
                    h1,
                    h2 {
font-weight: bold;
color: green;
                    }

body {
margin-left: 130px;
                    }
</style>
</head>

<body>
<h2>DOM appendChild() Method</h2>
<ul id="demo">
<li>Python Programming</li>
<li>Web X.0</li>
</ul>
<button onclick="geeks()">Submit</button>
<script>
function geeks() {
var node = document.createElement("LI");
vartextnode = document.createTextNode("Web Technology");
node.appendChild(textnode);
document.getElementById("demo").appendChild(node);
                    }
</script>
</body>

</html>
```

Output





Inserting an element using DOM

The insertBefore() method in HTML DOM is used to insert a new node before the existing node as specified by the user.

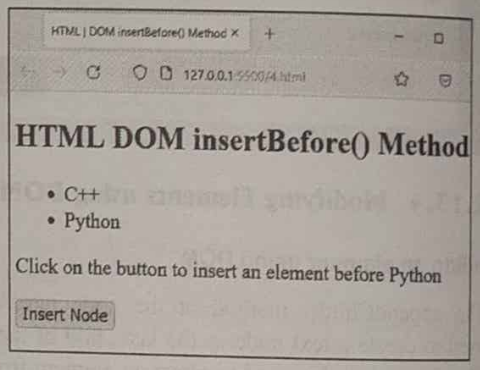
Syntax

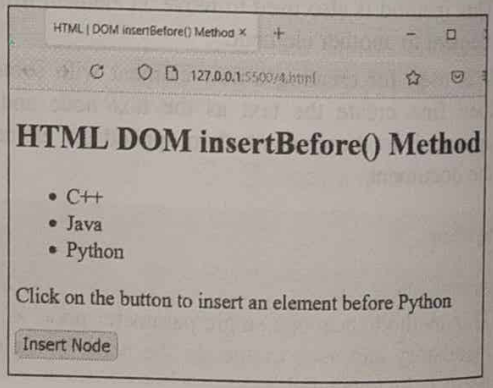
- node.insertBefore(newnode, existingnode)
- **Parameters :** This method accepts two parameters as mentioned above and described below:
- **Newnode :** It is the required parameter. This parameter contains the new node object which need to insert.
- **existingnode :** It is the required parameter. It describes the position where new node insert before this node. If it set to null then insertBefore method will insert the new node at the end.

Program 2.13.10 : Insert a list element before the second node of the list.

```
<!DOCTYPE html>
<html>

<head>
<title>
                    HTML | DOM insertBefore() Method

</title>
<!--Script to insert a new node before existing node-->
<script>
functionmyGeeks() {
varnewItem = document.createElement("li");
vartextnode = document.createTextNode("Java");
newItem.appendChild(textnode);

var list = document.getElementById("subjects");
list.insertBefore(newItem, list.childNodes[2]);
                    }
</script>
</head>
<body>

<h2>
                    HTML DOM insertBefore() Method
</h2>
<ul id="subjects">
<li>C++</li>
<li>Python</li>
</ul>
<p>
Click on the button to insert an element before Python
</p>
<button onclick="myGeeks()">
                    Insert Node
</button>
</body>
</html>
```

Output





Removing an element using DOM

- The removeChild() method in HTML DOM is used to remove a specified child node of the given element.
- It returns the removed node as a node object or null if the node doesn't exist.

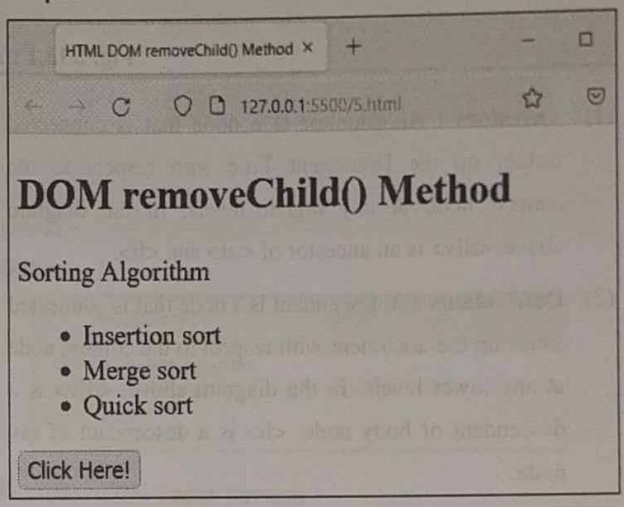
Syntax

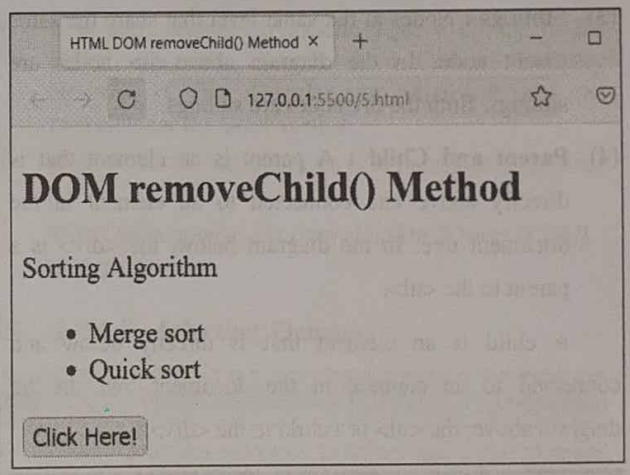
- node.removeChild(child)
- **Parameters :** This method accepts single parameter child which is mandatory. It represents the node which need to be remove.
- **Return Value :** It returns a node object which represents the removed node, or null if the node doesn't exist.

Program 2.13.11 : DOM removeChild() Method

```
<!DOCTYPE html>
<html>
<head>
<title>
                HTML DOM removeChild() Method
</title>
</head>
<body>
<h2>
                DOM removeChild() Method
</h2>

<p>Sorting Algorithm</p>

<ul id="listitem">
<li>Insertion sort</li>
<li>Merge sort</li>
<li>Quick sort</li>
</ul>

<button onclick="sample()">
                Click Here!
</button>

<script>
function sample() {
var doc = document.getElementById("listitem");
doc.removeChild(doc.childNodes[0]);
                }
</script>

</body>
</html>
```

Output





2.13.5 The Document Tree

Each HTML document can be represented in the tree format, where the elements can be described in a family-like hierarchy having ancestors, descendants, parents, children, and siblings.

Unit II In Sem.

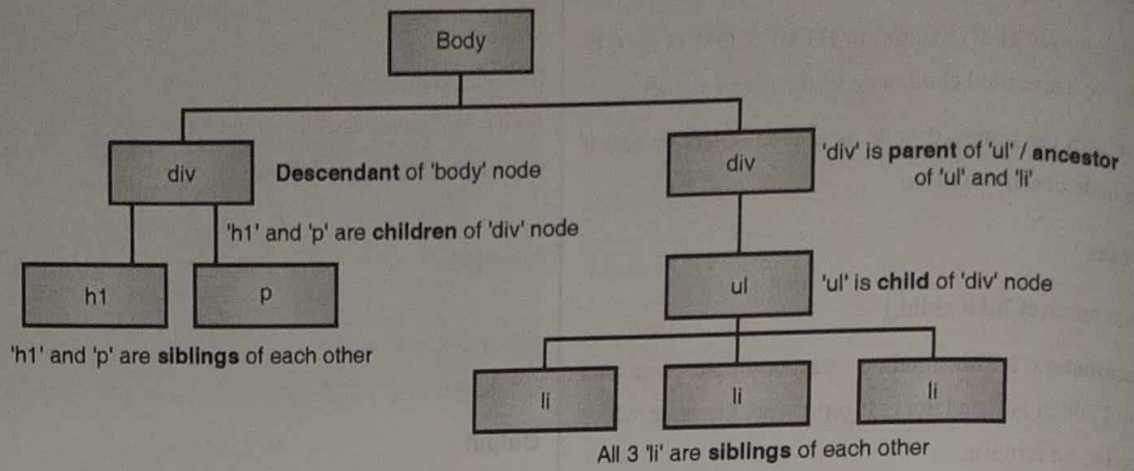


Fig. 2.13.1 : Document Tree

(1) **Ancestors :** An ancestor is a node that is connected further up the Document Tree with respect to the context node at any higher levels. In the diagram above, is an ancestor of <ul> and <li>.

(2) **Descendants :** A descendant is a node that is connected lower up the document with respect to the context node at any lower levels. In the diagram above, is a descendant of body node, <li> is a descendant of div node.

(3) **Siblings :** Nodes at the same level that share the same parent node. In the diagram above, nodes are siblings. Both the div nodes are siblings.

(4) **Parent and Child :** A parent is an element that is directly above and connected to an element in the document tree. In the diagram below, the is a parent to the <ul>.

A child is an element that is directly below and connected to an element in the document tree. In the diagram above, the <ul> is a child to the .

2.14 INTRODUCTION TO JQUERY

- jQuery is basically a JavaScript library which is cross-platform and used to support the client side scripting of HTML. jQuery is open source and free application available on internet.
- jQuery's provides various functionalities like navigation in web site, selecting DOM elements, managing events, creating different types of animations etc. jQuery provides a modular approach which simplifies the creation of dynamic content.
- jQuery was initially released by John Resig at BarCamp NYC in January 2006. Now a day jQuery is maintained by Timmy Willison and his team.

2.14.1 Features of jQuery

GQ. Write features of jQuery. **(4 Marks)**

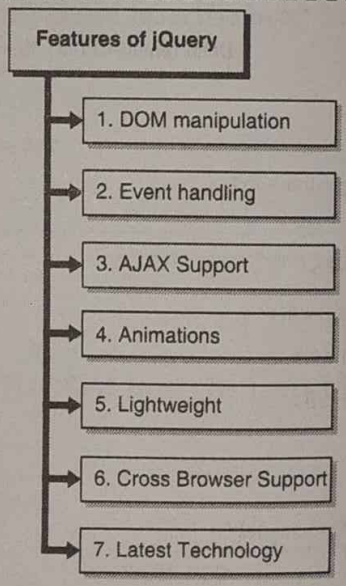


Fig. 2.14.1 : Features of jQuery

1. DOM manipulation

The DOM manipulation which includes selection of elements, traversing and modification in contents is easy with jQuery. jQuery make it easy with the help of cross-browser open source selector engine called **Sizzle**.

2. Event handling

Handling a wide variety of events like link click, mouse over etc. are effectively handled in the jQuery using event handlers.

3. AJAX Support

AJAX technology can be used with jQuery to develop responsive websites having a rich set of advanced features.

4. Animations

Number of built-in animation effects are provided by the jQuery.

5. Lightweight

The jQuery is a lightweight library which takes minimum time to load. It is about 19KB in size only.

6. Cross Browser Support

jQuery is supported by various latest browsers like Chrome, Mozila, IE, Opera, Safari erc.

7. Latest Technology

The advanced technologies like CSS3 selectors and basic XPath syntax are supported by jQuery.

2.14.2 Loading jQuery

(1) **Local Installation :** The jQuery library can be loaded on our local machine. This library can be used in the HTML document.

(2) **CDN Based Version :** It is also possible to add jQuery library in the HTML document directly from Content Delivery Network (CDN).

(A) Local Installation

URL to download jQuery :

https://jQuery.com/download/

- Now save this downloaded **jQuery-3.2.1.min.js** file in a directory of your website, e.g. /jQuery.
- Now you can include *jQuery* library in your HTML file as follows. As the webpage and *jQuery* library are in the same directory, there is no need to specify whole path in "src" attribute.

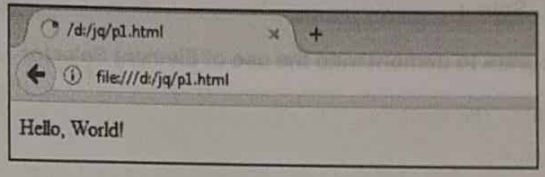
Program 2.14.1 : Write a simple program displaying hello world message using jQuery library

Soln. :

Program displaying hello world message using jQuery library

```
<html>
  <head>
    <title>jQuery</title>
    <script type = "text/javascript" src = "jQuery-
3.2.1.min.js"></script>
    <script type = "text/javascript">
     $(document).ready(function()
     {
       document.write("Hello, World!");
     });
    </script>
  </head>
  <body>
  </body>
</html>
```

Output



(B) CDN Based Version

- It is also possible to include jQuery library in the HTML document directly from Content Delivery Network (CDN). Google and Microsoft are the providers for the latest version.
- Here we have to set the value of *src* attribute as follows

  src :

  https://ajax.googleapis.com/ajax/libs/jQuery/3.2.1/jQuery.min.js

2.14.3 Selecting Elements

GQ. Explain different selectors in jQuery. **(4 Marks)**

- jQuery provides selectors to select and make changes in HTML elements.
- jQuery selectors use the name, id, classes, types, attributes, values of attributes etc to select HTML elements.
- In jQuery, the selectors start with the symbol $ (dollar sign) and parentheses: $()
- Now we will discuss different type of selectors available in jQuery.

Unit II In Sem.

☞ Types of selectors

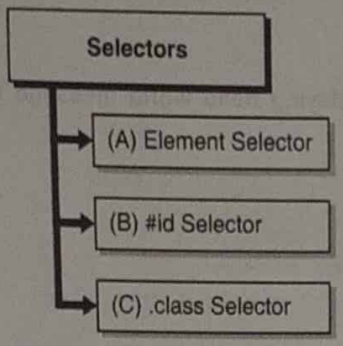


Fig. 2.14.2 : Different types of selectors

▶ (A) The Element Selector

The jQuery element selector selects the HTML elements depending upon the element name.

<p> elements can be selected as follows :

```
$("p")
```

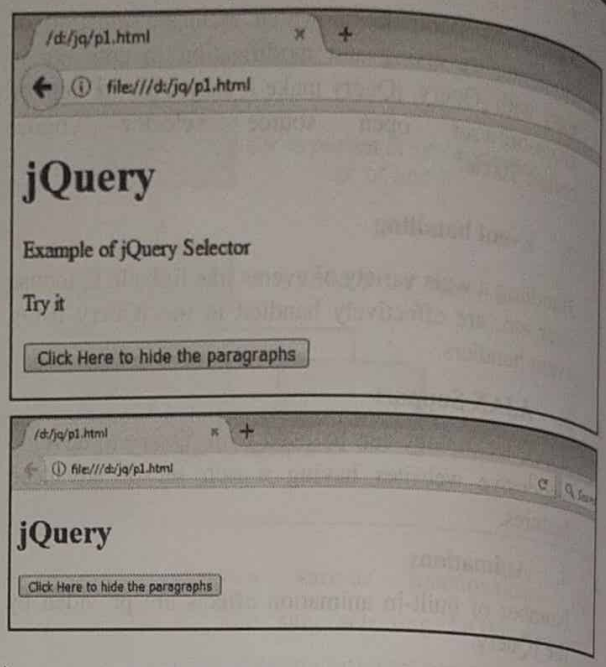
Program 2.14.2 : Write a program to demonstrate the use of Element Selector.

☑ **Soln. :**

Program to demonstrate the use of Element Selector

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>
<body>
<h1>jQuery</h1>
<p>Example of jQuery Selector</p>
<p>Try it</p>
<button>Click Here to hide the paragraphs</button>
</body>
</html>
```

Output



▶ (B) The #id Selector

- The jQuery #id selector selects the HTML id attribute of the HTML tag to find the specific element. All the elements should have unique id.
- To search an element with a given id, hash character is written which is followed by the HTML element id.

```
$("#id1")
```

Program 2.14.3 : Write a program to demonstrate the use of #id Selector.

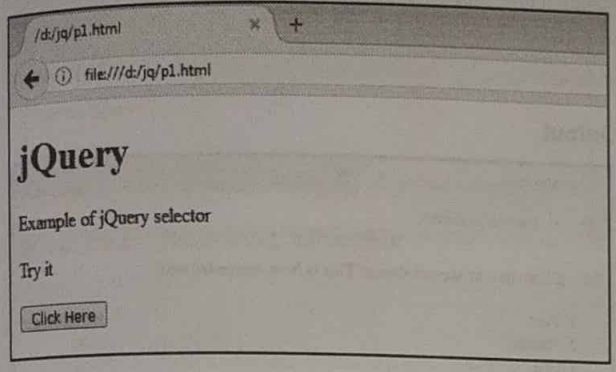
☑ **Soln. :**

Program to demonstrate the use of #id Selector

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#id1").hide();
  });
});
</script>
</head>
<body>
<h1>jQuery</h2>
<p>Example of jQuery selector</p>
<p id="id1">Try it</p>
```

```
<button>Click Here</button>
</body>
</html>
```

Output



(C) The .class Selector

The jQuery class selector finds elements with a specific class. To search an element with a specific class, period character is written which is followed by the name of the class.

```
$(".myclass")
```

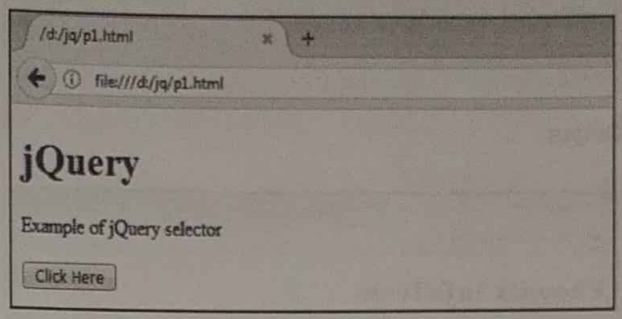
Program 2.14.4 : Write a program to demonstrate the use of .class Selector.

Soln. :

Program to demonstrate the use of .class Selector

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $(".myclass").hide();
  });
});
</script>
</head>
<body>
<h1 class="myclass">jQuery</h2>
<p class="myclass">Example of jQuery selector</p>
<button>Click Here</button>
</body>
</html>
```

Output



Unit II In Sem.

2.14.4 Changing Styles

GQ. How to change style in jQuery? Give example. **(4 Marks)**

jQuery css() Method is used to set or return one or more style properties for the selected elements.

Syntax

To set a specified CSS property, use the following syntax :

```
css("propertyname","value");
```

Example

```
$("p").css("background-color", "skyblue");
```

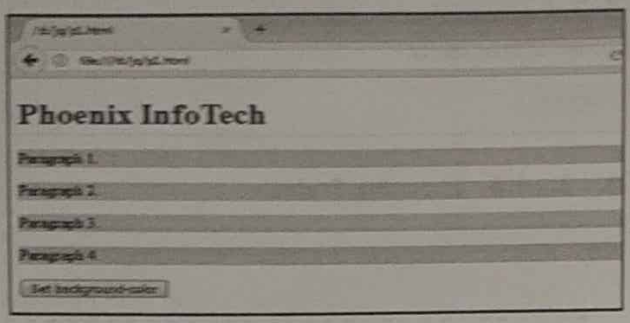
Program 2.14.5 : Write a program to change style in jQuery.

Soln. : Program to change style in jQuery

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"
></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").css("background-color", "skyblue");
  });
});
</script>
</head>
<body>
<h1>Phoenix InfoTech</h1>
<p style="background-color:red">Paragraph 1.</p>
<p style="background-color:green">Paragraph 2.</p>
<p style="background-color:blue">Paragraph 3.</p>
```

```
<p>Paragraph 4.</p>
<button>Set background-color</button>
</body>
</html>
```

Output



2.14.5 Creating and Appending Elements

GQ. Write an example (program) to create and append HTML element in jQuery. **(4 Marks)**

jQuery provides **append()** method to add an element.

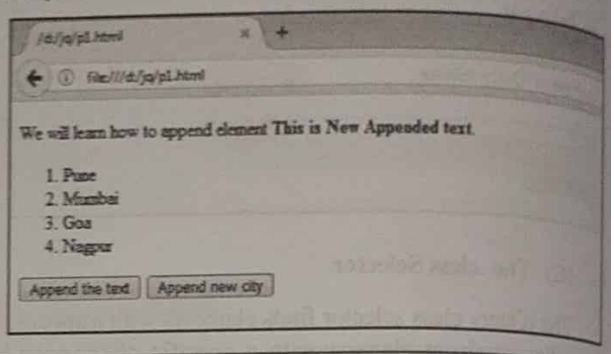
Program 2.14.6 : Write a program to create and append HTML element in jQuery.

☑ **Soln. :**

Creating and appending HTML element in jQuery

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"
></script>
<script>
$(document).ready(function(){
  $("#btnA").click(function(){
    $("p").append(" <b>This is New Appended text</b>.");
  });
  $("#btnB").click(function(){
    $("ol").append("<li>Nagpur</li>");
  });
});
</script>
</head>
<body>
<p>We will learn how to append element</p>
<ol>
  <li>Pune</li>
  <li>Mumbai</li>
  <li>Goa</li>
</ol>
<button id="btnA">Append the text</button>
<button id="btnB">Append new city</button>
</body>
</html>
```

Output



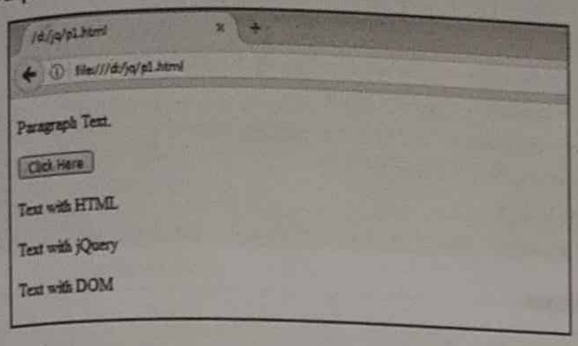
Program 2.14.7 : Write a program to add multiple elements at a time in jQuery.

☑ **Soln. :**

Program to add multiple elements at a time in jQuery

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"
></script>
<script>
function appText()
{
  var t1 = "<p> Text with HTML</p>";
  var t2 = $("<p></p>").text("Text with jQuery");
  var t3 = document.createElement("p");
  t3.innerHTML = "Text with DOM";
  $("body").append(t1, t2, t3);    // Append new elements
}
</script>
</head>
<body>
<p>Paragraph Text.</p>
<button onclick="appText()">Click Here</button>
</body>
</html>
```

Output



2.14.6 Removing Elements

GQ. Write an example (program) to remove HTML element in jQuery. **(4 Marks)**

Removing existing HTML elements is very easy in jQuery.

To remove elements and its content, jQuery provides two methods :

(i) **remove() :** Removes the selected element with its child elements.

(ii) **empty() :** Removes the child elements from the selected element.

☞ jQuery remove() Method

This method removes the selected element with its child elements.

☞ Syntax

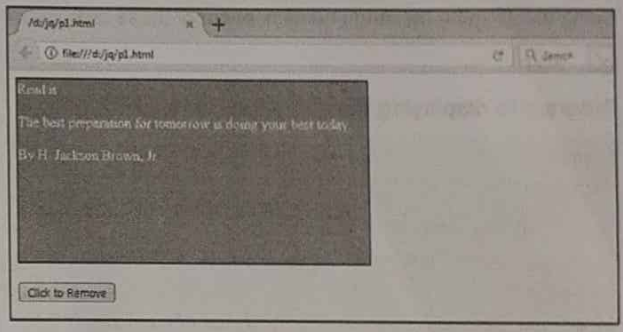
```
$("#div1").remove();
```

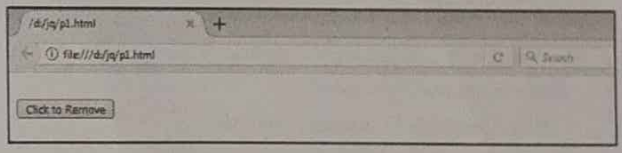
Program 2.14.8 : Write a program to demonstrate the use of remove method.

☑ Soln. : Program to demonstrate the use of remove method

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"
></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#mydiv").remove();
  });
});
</script>
</head>
<body>
<div id="mydiv"
style="height:200px;width:400px;color:white;border:2px solid
black;background-
color:gray;">
Read it
<p>The best preparation for tomorrow is doing your best
today.</p>
<p>By H. Jackson Brown, Jr.</p>
</div>
<br>

<button>Click to Remove</button>

</body>
</html>
```

Output





2.14.7 Handling Events

GQ. Explain Event Handling in jQuery. **(4 Marks)**

Events are the actions which are taken by the end user while browsing the websites. We can execute specific scripts on fire of different events.

jQuery provides various types of events.

- Mouse click
- Page load
- Mouse over
- Submitting an HTML form
- Key Events

When these events get fired we can call custom functions to execute scripts. Such custom functions are called as **Event Handlers**.

(A) Binding event handlers

The events handlers can be established using bind() method on DOM elements in jQuery Event Model.

Syntax

The syntax of the bind() method is as follows

```
selector.bind( eventType[, eventData], handler)
```

- **eventType** – It is the string which contains the JavaScript event types like click or submit
- **eventData** – This is a map of data that will be passed to the event handler. It is optional.
- **handler** – It is a function which will be executed when the event is triggered.

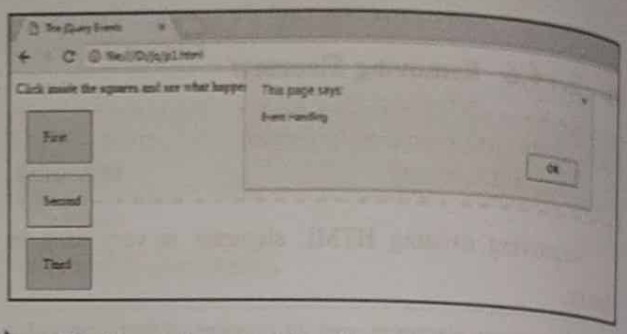
Program 2.14.9 : Write a program to draw three sections using div. Display message on click event of these divs.

Soln. :

Program to displaying three sections on click event

```
<html>
  <head>
    <title>The jQuery Events</title>
    <script type = "text/javascript"
     src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></
script>
    <script type = "text/javascript" language = "javascript">
     $(document).ready(function() {
       $('div').bind('click', function( event ){
         alert('Event Handling');
       });
     });
    </script>
    <style>
     .div{ margin:13px;padding:20px; border:1px solid #666;
width:40px;}
    </style>
  </head>
  <body>
    <p>Click inside the squares and see what happens.</p>
    <div class = "div" style = "background-
color:pink;">First</div>
    <div class = "div" style = "background-
color:yellow;">Second</div>
    <div class = "div" style = "background-
color:skyblue;">Third</div>
  </body>
</html>
```

Output



(B) Removing event handlers

Sometimes we may want to remove the event handler for a particular reason which we have already created. jQuery provides unbind() method to remove the event handler.

Syntax

```
selector.unbind(eventType, handler)
```

or

```
selector.unbind(eventType)
```

(C) Event Types

There are various types of events in jQuery. Some of them are as follows :

Sr. No.	Event Types & Description
1.	**Blur :** Fires when the element lost the focus.
2.	**Change :** Fires when the element changes.
3.	**Click :** Fires when element get clicked.
4.	**Dblclick :** Fires when element get double clicked.
5.	**error :** Fires when error occurs in loading or unloading etc.
6.	**focus :** Fires when the element gets focus on it.
7.	**Keydown :** Fires when key is pressed.
8.	**Keypress :** Fires when key is pressed and released.

Sr. No.	Event Types & Description
9.	**Keyup :** Fires when key is released.
10.	**Load :** Fires when document is loaded.
11.	**mousedown :** Fires when mouse button is pressed.
12.	**mouseenter :** Fires when mouse pointer moves on the element.
13.	**Mouseleave :** Fires when mouse pointer leaves the element.
14.	**Mousemove :** Fires when mouse pointer moves.
15.	**Mouseup :** fires when mouse button is released.
16.	**resize :** Fires when window is resized.
17.	**Scroll :** Fires when window is scrolled.
18.	**select :** Fires when a text is selected.
19.	**submit :** Fires when form is submitted.
20.	**unload** Fires when documents are unloaded.

(D) The Event Attributes : There are number of attributes for an event in jQuery. These attributes provides different types of information about the particular event object.

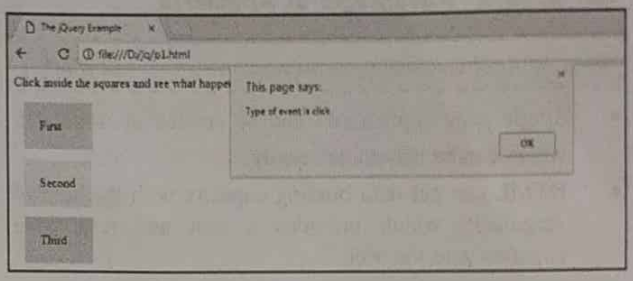
Program 2.14.10 : Write a program to draw three sections using div, display information about event type, X and Y co-ordinates, and the section which get clicked.

☑ **Soln. :**

Program to draw the three sections and displaying the information about event type, X, Y co-ordinates and section which get clicked

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></
script>
    <script type = "text/javascript" type = "text/javascript">
      $(document).ready(function() {
        $('div').bind('click', function( event ){
          alert('Type of event is ' + event.type);
          alert('X co-ordinate : ' + event.pageX);
          alert('Yco-ordinate : ' + event.pageY);
          alert('The Section click is  : ' + event.target.innerHTML);
        });
      });
    </script>
    <style>
      .div{ margin:13px;padding:20px;color:white border:3px solid
#666; width:50px;}
    </style>
  </head>
  <body>
      <p>Click inside the squares and see what happens:</p>
    <div class = "div" style = "background-
color:pink;">First</div>
    <div class = "div" style = "background-
color:yellow;">Second</div>
    <div class = "div" style = "background-
color:skyblue;">Third</div>
  </body>
</html>
```

Output



2.15 INTRODUCTION TO ANGULAR JS

GQ. What is AngularJS ? **(2 Marks)**

❒ **Definition : AngularJS** is JavaScript MVC framework which runs at client side and used to execute dynamic web applications.

- **Misko Hevery** and **Adam Abrons** developed AngularJS in the year of 2009. Later on it is maintained by Google. It is now open source framework.
- AngularJS is completely based on HTML and JavaScript.
- Sometimes AngularJS is also pronounced just as **"Angular"**.

Unit II In Sem.

- The main aim of AngularJS is to convert static HTML into dynamic HTML. It adds built-in attributes and components as well as enhances ability to generate custom attributes with the help of simple JavaScript. This extends the ability of HTML.
- Now a day AngularJS is probably one of the best advanced web frameworks available. The main purpose of AngularJS is to develop Single Page applications.

2.15.1 Features of AngularJS

GQ. Write the features of AngularJS. **(2 Marks)**

- AngularJS is considered as very powerful JavaScript oriented framework used to develop RIA (RICH Internet Application).
- AngularJS provides an environment to developers for creation of client side applications using JavaScript.
- AngularJS can create applications which are compatible for all the browsers.
- AngularJS is a free and open source framework.

2.15.2 Advantages of AngularJS

GQ. Write the advantages of AngularJS. **(2 Marks)**

- Single page applications can be created in AngularJS which can be maintained easily.
- HTML can get data binding capacity with the help of AngularJS which provides a rich and responsive experience to the user.
- Unit testing is possible in AngularJS code.
- Dependency injection is used in AngularJS to separate different concerns.
- Reusability of components is supported by AngularJS.
- With less code, programmer can get more functionality in AngularJS.
- In AngularJS, views are the basic HTML static pages, and controllers are developed with JavaScript for the purpose of processing business logic.

2.15.3 MVC Architecture

GQ. Explain the MVC Architecture. **(4 Marks)**

- MVC stands for **Model View Controller**. MVC Architecture is a software design pattern which is used to develop web applications.
- The MVC Architecture contains three main components.

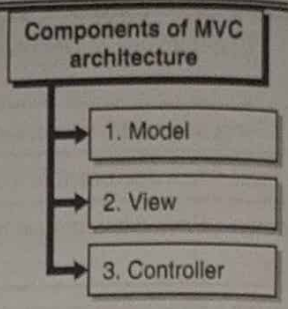


Fig. 2.15.1 : Components of MVC architecture

1. Model

It is considered as the lower level of the pattern which maintains the data.

2. View

It displays the data to the end user.

3. Controller

It is the interface between Model and View which controls their interactions. The main advantage of MVC Architecture is that it separates the application logic from user interface layer. All the requests for the applications are received by the Controller. Then with Model the controller creates the data required for the View. This data is utilized by the View to create the final presentable response.

The graphical representation of MVC is as shown in the Fig. 2.15.2.

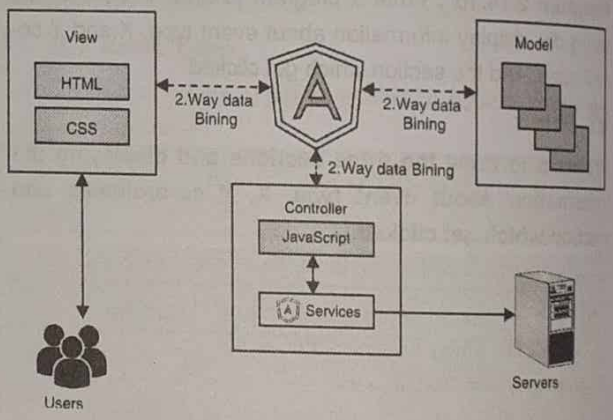


Fig. 2.15.2 : MVC Architecture

1. The Model

- The data is represented by the Model. The data in the model may be as simple as containing only variable declarations. For example, in an employee application, the model data could just contain employee id and name. Or it may also be very complex by containing a structured data model.

- The application data is managed by the Model. It gives response to the request of view as well as instructions from the controller to update itself.

2. The View

The functionality of View is to represent the presentation layer which is to be displayed to the end user.

3. The Controller

- The layer having the business logic is represented by the Controller. The Controller has functions in it which are called by the trigger of user events.
- The controller accepts input provided by the user, validates it, and then executes business operations because of which the state of data model is modified.

AngularJS is a MVC based framework.

2.15.4 Directives

GQ. Explain various directives in AngularJS with suitable example (program). **(8 Marks)**

AngularJS directives extend the functionality of HTML. These directives are the special attributes which starts with ng- prefix.

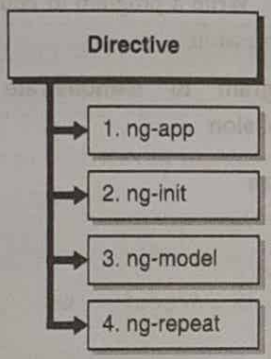


Fig. 2.15.3 : Directives

1. ng-app directive

- The AngularJS application is started with ng-app directive. The root element is defined by this directive. The application is initialized by ng-app directive when the web page is loaded. The different AngularJS modules are loaded by this directive in the application.
- In the following example, AngularJS application is created using ng-app attribute of a div element.

Syntax

```
<div ng-app = "">
  ...
</div>
```

2. ng-init directive

- The application data in the AngularJS is initialized by this directive. It assigns values to the application variables.
- In following example, an array of cities is initialized. Here JSON syntax is used to define the array.

Syntax

```
<div ng-app = "" ng-init = "Cities = [{code:'pn',name:'Pune'},
{code:'Mu',name:'Mumbai'},
{code:'Na',name:'Nasik'}]">
  ...
</div>
```

3. ng-model directive

- The ng-model directive is used to bind the AngularJS application data values with input controls of HTML.
- In following example, model "name" is defined.

Syntax

```
<div ng-app = "">
  ...
  <p>Enter your Name: <input type = "text"
ng-model = "name"></p>
</div>
```

4. ng-repeat directive

- **ng-repeat** directive is used to repeat HTML elements for each and every item in a collection
- In the following example, the array of cities is iterated.

```
<div ng-app = "">
  ...
    <p>List of Cities With Code :</p>
    <ol>
      <li ng-repeat = "City in Cities">
        {{ 'City : ' + City.name + ', Code : ' + City.code }}
      </li>
    </ol>
  </div>
```

Now in a single application, we will use all the above mentioned directives.

Program 2.15.1 : Write a program to demonstrate the use of AngularJS Directives.

☑ **Soln. :**

Program to demonstrate the use of AngularJS Directives

```
<html>
  <head>
  <title>Directives in AngularJS</title>
 </head>
 <body>
  <h1>Directives</h1>
<font size=5>
 <div ng-app ="" ng-init = "Cities = [{code:'pn',name:'Pune'},
{code:'Mu',name:'Mumbai'},
{code:'Na',name:'Nasik'}]">
```

Values are assigned to application variable cities

```
<p>Enter your Name : <input type = "text"
ng-model = "name"></p>
    <p>Hello <span ng-bind = "name"></span>!</p>
```

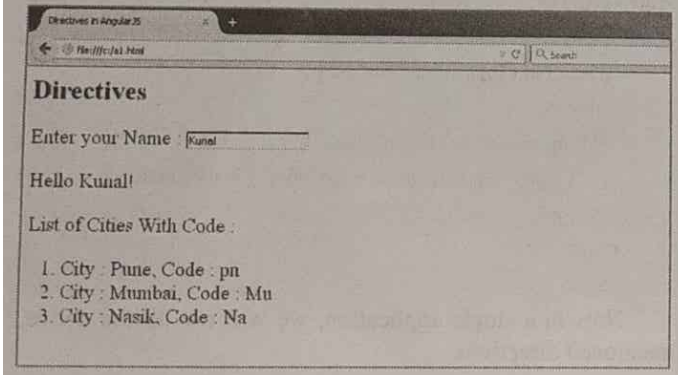
Textbox is bind with application data

```
    <p>List of Cities With Code :</p>
<ol>
        <li ng-repeat = "City in Cities">
```

 is repeated for each city

```
        {{ 'City : ' + City.name + ', Code : ' + City.code }}
       </li>
      </ol>
    </div>
</font>
 <script src ="https://ajax.googleapis.com/ajax/libs/
 angularjs/1.3.14/angular.min.js"></script>
  </body>
</html>
```

Output



2.15.5 Expression

GQ. Explain AngularJS Expression with suitable example. (Program). **(4 Marks)**

- Expressions are used to bind application data to html. Expressions are enclosed inside double braces like {{ expression statements }}.
- The behavior of expression is same as of the ng-bind directives. In AngularJS application, the expressions are purely JavaScript statements which generate the output where they are used.

☞ **Using numbers**

```
<p>Expense on Books : {{book_cost * book_quantity}} Rs</p>
```

☞ **Using strings**

```
<p>Hello {{stud.firstname + " " + stud.lastname}}!</p>
```

☞ **Using object**

```
<p>Roll No: {{stud.rollno}}</p>
```

☞ **Using array**

```
<p>Marks(Math): {{marks[3]}}</p>
```

Following program will show all case the above mentioned expressions.

Program 2.15.2 : Write a program to demonstrate the use of AngularJS Expression.

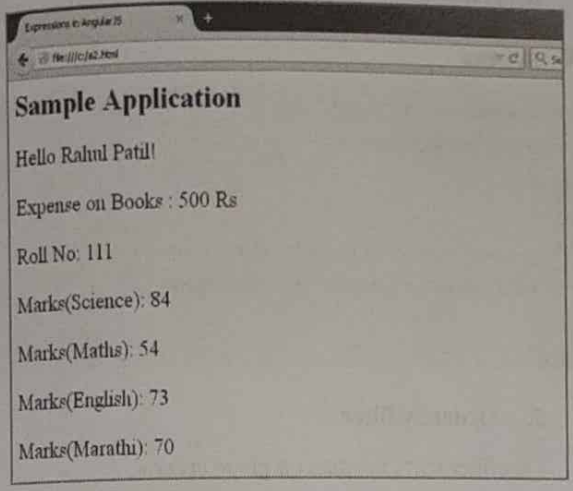
☑ **Soln.: Program to demonstrate the use of AngularJS Expression**

testAngularJS.htm

```
<html>
  <head>
  <title> Expressions in AngularJS</title>
 </head>
 <body>
  <h1>Sample Application</h1>
<font size=5>
  <div ng-app = "" ng-init = "book_quantity = 5;
   book_cost = 100; stud =
{firstname:'Rahul',lastname:'Patil',rollno:111};
marks = [84,54,73,70]">
    <p>Hello {{stud.firstname + " " +
   stud.lastname}}!</p>
    <p>Expense on Books : {{ book_cost *
   book_quantity}} Rs</p>
    <p>Roll No: {{stud.rollno}}</p>
```

```
    <p>Marks(Science): {{marks[0]}}</p>
    <p>Marks(Maths): {{marks[1]}}</p>
    <p>Marks(English): {{marks[2]}}</p>
    <p>Marks(Marathi): {{marks[3]}}</p>
  </div>
</font>
  <script src = "https://ajax.googleapis.com/ajax/libs/
   angularjs/1.3.14/angular.min.js"></script>
 </body>
</html>
```

Output



2.15.6 Controllers

GQ. Write short note on Controllers in AngularJS. **(4 Marks)**

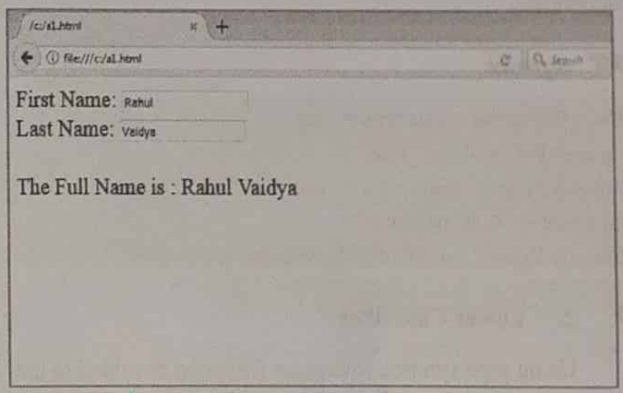
- A controller is a JavaScript object which contains the attributes and functions.
- The ng-controller directive is used to define the controllers.
- The controllers accept $scope as a parameter. The $scope refers to the application or the module which is to be controlled by the controller.
- In an AngularJS application, the flow of data is controlled by the controllers which get passed to the view. There is a two way communication between the scope and the view.

Program 2.15.3 : Write a program to accept name from user and display it using controller in AngularJS.

Soln. : Program to Accept name from user and displaying using controller in AngularJS.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.4.8/angular.min.js"></script>
<body>
<font size=5>
<div ng-app="FirstApp" ng-controller="FirstCtrl">
First Name: <input type="text" ng-model="first_Name"><br>
Last Name: <input type="text" ng-model="last_Name"><br>
<br>
The Full Name is : {{first_Name + " " + last_Name}}
</div>
<script>
var app1 = angular.module('FirstApp', []);
app1.controller('FirstCtrl', function($scope)
{
  $scope.first_Name = "Rahul";
  $scope.last_Name = "Vaidya";
});
</script>
</font>
</body>
</html>
```

Output



2.15.7 Filters

GQ. Explain various filters in AngularJS with example (program). **(8 Marks)**

Filters are used to transform or format the data in AnugularJS. These filters can be clubbed into expression as well as directives with the help of pipe character.

Unit II In Sem.

AngularJS provides number of filters as follows:

Sr. No.	Name	Description
1.	uppercase	Converts the string into upper case.
2.	lowercase	Converts the string into lower case.
3.	currency	Formats text in the form of currency.
4.	filter	Apply filter based on specified criteria and return the subset.
5.	orderby	Sorts the array elements based on specified criteria.

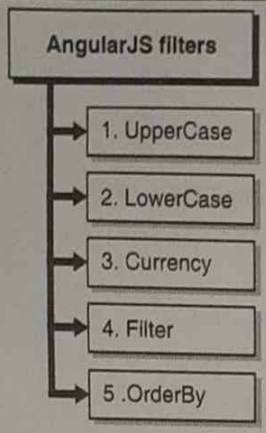


Fig. 2.15.4 : AngularJS filters

1. Upper Case filter

Using pipe symbol, uppercase filter can be added to the expression in an application to converts the data into uppercase.

Syntax

```
Enter first name:<input type = "text"
ng-model = "stud.first_Name">
Enter last name: <input type = "text"
ng-model = "stud.last_Name">
Name in Upper Çase: {{stud.full_Name() | uppercase}}
```

2. Lower Case filter

Using pipe symbol, lowercase filter can be added to the expression in an application to converts the data into lowercase.

Syntax

```
Enter first name:<input type = "text"
ng-model = "stud.first_Name">
Enter last name: <input type = "text"
ng-model = "stud.last_Name">
Name in Upper Case: {{stud.full_Name() | lowercase}}
```

3. Currency filter

The currency filter can be added to an expression using pipe character. It returns number.

Syntax

```
Enter fees: <input type = "text" ng-model = "stud.fees">
fees: {{stud.fees | currency}}
```

4. Filter filter

This filter is used to filter the data from list as per our requirement.

Syntax

```
Enter subject: <input type = "text"
ng-model = "subjectName">
Subject:
<ul>
 <li ng-repeat = "subject in stud.subjects | filter: subjectName">
  {{ subject.name + ', marks:' + subject.marks }}
 </li>
</ul>
```

5. OrderBy filter

This filter sorts the data on given criteria.

Syntax

```
Subject:
<ul>
 <li ng-repeat = "subject in stud.subjects | filter: subjectName
|orderBy:'marks'">
{{ subject.name + ', marks:' + subject.marks }}
 </li>
</ul>
```

Program 2.15.4 : Write a program to showcase all the AngularJS filters.

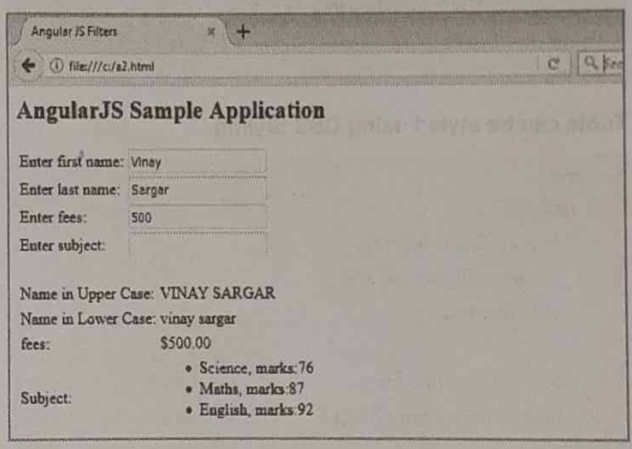
Soln. : Program to show all the cases in AngularJs filters.

Test1.htm

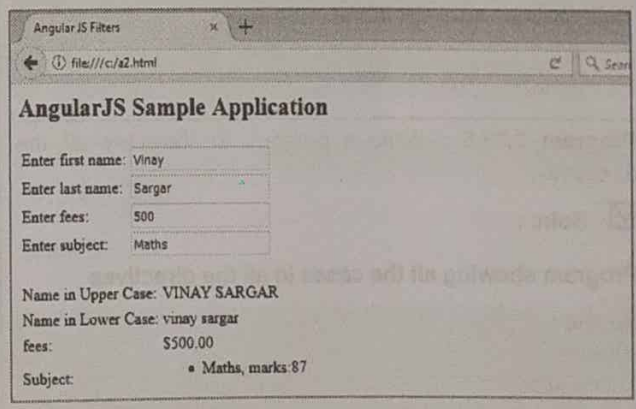
```
<html>
 <head>
  <title>Angular JS Filters</title>
  <script src = "https://ajax.googleapis.com/ajax/libs/
   angularjs/1.3.14/angular.min.js"></script>
 </head>
 <body>
  <h2>AngularJS Sample Application</h2>
```

```
<div ng-app = "mainApp" ng-controller = "studContr">
  <table border = "0">
    <tr>
      <td>Enter first name:</td>
      <td><input type = "text"
      ng-model = "stud.first_name"></td>
    </tr>
    <tr>
      <td>Enter last name: </td>
      <td><input type = "text"
      ng-model = "stud.last_name"></td>
    </tr>
    <tr>
      <td>Enter fees: </td>
      <td><input type = "text"
      ng-model = "stud.fees"></td>
    </tr>
      <tr>
      <td>Enter subject: </td>
      <td><input type = "text"
      ng-model = "subjectName"></td>
    </tr>
  </table>
  <br/>
  <table border = "0">
    <tr>
      <td>Name in Upper Case:
</td><td>{{stud.full_name() | uppercase}}</td>
    </tr>
    <tr>
      <td>Name in Lower Case:
</td><td>{{stud.full_name() | lowercase}}</td>
    </tr>
    <tr>
      <td>fees: </td><td>{{stud.fees | currency}}
      </td>
    </tr>
    <tr>
      <td>Subject:</td>
      <td>
        <ul>
          <li ng-repeat = "subject in stud.subjects |      filter:
subjectName |orderBy:'marks'">
          {{ subject.name + ', marks:' + subject.marks }}
          </li>
        </ul>
      </td>
    </tr>
  </table>
</div>
  <script>
    var mainApp = angular.module("mainApp", []);
    mainApp.controller('studContr', function($scope) {
      $scope.stud = {
        first_name: "Vinay",
        last_name: "Sargar",
        fees:500,
        subjects:[
          {name:'Maths',marks:87},
          {name:'Science',marks:76},
          {name:'English',marks:92}  ],
        full_name: function() {
          var studObject;
          studObject = $scope.stud;
          return studObject.first_name + " "
          + studObject.last_name;
        }   };
    });
  </script>
 </body>
</html>
```

Default Output window



Output window when subject is entered



Unit II In Sem.

2.15.8 Tables

GQ. Write short note on Tables in AngularJS with suitable example (program). **(4 Marks)**

Table is combination of rows and columns. In AngularJS, we can display the data in table format. As the elements are repeated in table, we can use ng-repeat directive to draw table easily.

Program 2.15.5 : Write a program to demonstrates the use of ng-repeat directive for drawing a table.

☑ **Soln. :**

Program to demonstrate the use of ng-repeat directive for drawing a table

```
<table>
  <tr>
    <th>Name</th>
    <th>Marks</th>
  </tr>
    <tr ng-repeat = "sub in stud.sub">
              <td>{{ sub.name }}</td>
              <td>{{ sub.marks }}</td>
            </tr>
</table>
```

Table can be styled using CSS Styling

```
<style>
      table, th , td {
        border: 2px solid grey;
        border-collapse: collapse;
        padding: 6px;
      }
      table tr:nth-child(odd) {
        background-color: #f2f2f2;
      }
      table tr:nth-child(even) {
        background-color: #ffffff;
      }
    </style>
```

Program 2.15.6 : Write a program to showcase all the directives.

☑ **Soln. :**

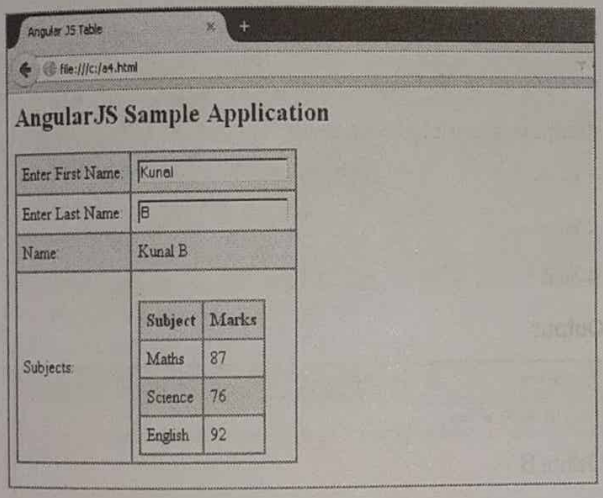
Program showing all the cases in all the directives

test.htm

```
<html>
  <head>
    <title>Angular JS Table</title>
    <script src = "https://ajax.googleapis.com/ajax/libs/
     angularjs/1.3.14/angular.min.js"></script>
    <style>
      table, th , td {
        border: 2px solid grey;
        border-collapse: collapse;
        padding: 6px;
      }
      table tr:nth-child(odd) {
        background-color: #f2f2f2;
      }
      table tr:nth-child(even) {
        background-color: #ffffff;
      }
    </style>
  </head>
  <body>
    <h2>AngularJS Sample Application</h2>
    <div ng-app = "mainApp"
     ng-controller = "student_Controller">
      <table border = "1">
        <tr>
          <td>Enter First Name:</td>
          <td><input type = "text"
            ng-model = "stud.first_Name"></td>
        </tr>
        <tr>
          <td>Enter Last Name: </td>
          <td>
            <input type = "text"
             ng-model = "stud.last_Name">
          </td>
        </tr>
        <tr>
          <td>Name: </td>
          <td>{{stud.full_Name()}}</td>
        </tr>
        <tr>
          <td>Subjects:</td>
          <td>
            <table>
              <tr>
                <th>Subject</th>.
                <th>Marks</th>
              </tr>
              <tr ng-repeat = "sub in stud.sub">
                <td>{{ sub.name }}</td>
                <td>{{ sub.marks }}</td>
              </tr>
            </table>
          </td>
        </tr>
      </table>
```

```
  </div>
  <script>
   var mainApp = angular.module("mainApp", []);
   mainApp.controller('student_Controller', function($scope) {
     $scope.stud = {
       first_Name: "Kunal",
       last_Name: "B",
       fees:500,
       sub:[
         {name:'Maths',marks:87},
         {name:'Science',marks:76},
         {name:'English',marks:92},
       ],
       full_Name: function() {
         var studentObject;
         studentObject = $scope.stud;
         return studentObject.first_Name + " " +
studentObject.last_Name;
       }
     };
   });
  </script>
 </body>
</html>
```

Output



2.15.9 Modules

GQ. What is module in AngularJS ? Explain with suitable example. **(4 Marks)**

- In AngularJS, the concept of module is used to define an application. Module works as a container for the different parts of AngularJS application such as controller, directives services, filters, etc.
- A module is used just like the main() method. Controller at all times belongs to a module.
- The module is defined in a separate js file.

Example

Creating Module

The AngularJS function angular.module() is used to create the module.

Syntax

```
<div ng-app="mainApp">...</div>
<script>var app1 = angular.module("mainApp", []);
</script>
```

- The "mainApp" parameter refers to the HTML element in which the application will be executed.
- We can add different elements like controllers, directives, filters etc. in the AngularJS application.

Unit II In Sem.

Adding a Controller

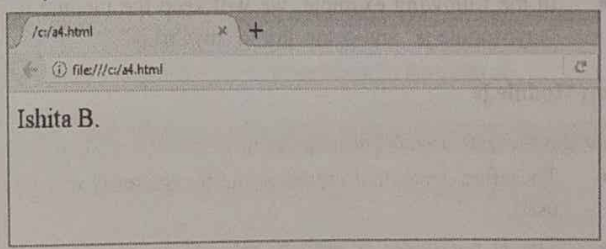
In the following code, we will add controller in our application and refer it with ng-Controller directive :

Program 2.15.7 : Write a program to add the controller in application and refer it with ng- controller directive.

Soln. : Adding the controller in application

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.6.4/angular.min.js"></script>
<body>
<font size=5>
<div ng-app="mainApp" ng-controller="myController">
{{ first_Name + " " + last_Name }}
</div>

<script>
var app = angular.module("mainApp", []);
app.controller("myController", function($scope) {
  $scope.first_Name = "Ishita";
  $scope.last_Name = "B.";
});
</script>
</font>
</body>
</html>
```

Output



Adding a Directive

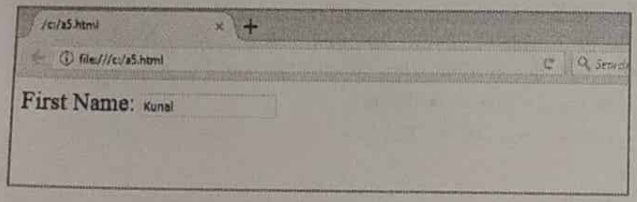
- AngularJS provides a set of built-in directives. These directives can be used to add functionality in the application.
- Now in the following example (code), we will use module to add the directives in the application.

Program 2.15.8 : Write a program to add the directives in application.

Soln. : Program to add the directives in application

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.6.4/angular.min.js"></script>
<body>
<font size=5>
<div ng-app="myApp" my-directive></div>
<script>
var app1 = angular.module("myApp", []);
app1.directive("myDirective", function() {
  return {
    template : "This an example of directive with module."
  };
});
</script>
</font>
</body>
</html>
```

Output



Modules and Controllers in Files

- In AngularJS applications we can keep the module and the controllers in JavaScript files.
- In the following example, we will keep the module in "myModule.js" and controller in 'myCtrl.js".

myModule.js

```
var app = angular.module("mainApp", []);
```

- To define dependent modules, the parameter [] will be used.
- The [] parameter is used to retrieve an existing one rather than creating a new module.

myCtrl.js

```
app.controller("myController", function($scope)
{
  $scope.first_Name = "Ishita";
  $scope.last_Name= "B";
});
```

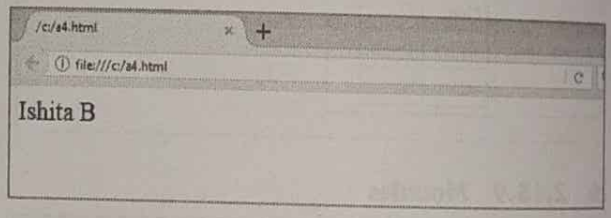
Program 2.15.9 : Write a program to use the module and controllers in JavaScript file.

Soln. :

Module and controllers in JavaScript file.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.6.4/angular.min.js"></script>
<body>
<font size=5>
<div ng-app="mainApp" ng-controller="myController">
{{ first_Name + " " + last_Name }}
</div>
<script src="myModule.js"></script>
<script src="myCtrl.js"></script>
</font>
</body>
</html>
```

Output



2.15.10 Forms

GQ. Write short note on Forms in AngularJS with different elements. Give suitable example (program) on each element. **(8 Marks)**

- In HTML, the form is a collection of input controls with the help of which user can enter the data.
- In AngularJS the forms provide the properties of data-binding and validation of input controls.

- There are various types of input controls which can be bind with data in AngularJS.
  1. Input elements
  2. Select elements
  3. Button elements
  4. Textarea elements

☞ Data-Binding with input controls

1. TextField

The ng-model directive can be used to bind the data with input controls :

```
<input type="text" ng-model="first_name">
```

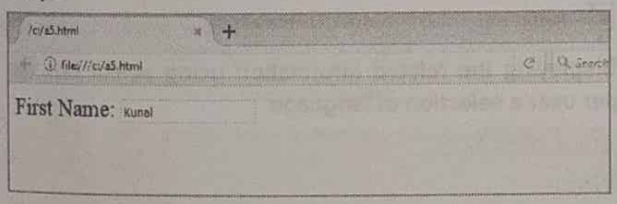
- There will be property named first_name in the application.
- The directive ng-model binds the input controller to the application data.
- The property first_name is referred in a controller.

Program 2.15.10 : Write a program to bind the data with input controls.

☑ Soln. : Binding the data with input controls

```
<!DOCTYPE html>
<html lang="en">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<body>
<font size=5>
<div ng-app="mainApp" ng-controller="formController">
 <form>
  First Name: <input type="text" ng-model="first_name">
 </form>
</div>
<script>
var app1 = angular.module('mainApp', []);
app1.controller('formController', function($scope) {
  $scope.first_name = "Kunal";
});
</script>
</font>
</body>
</html>
```

Output



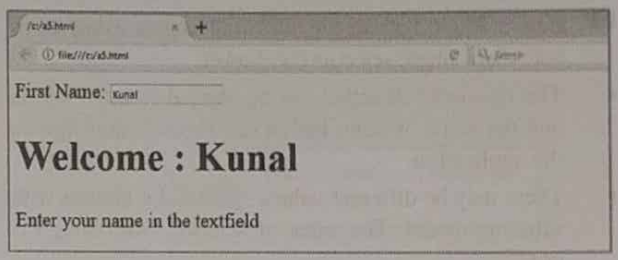
Now this data can referred at any place in the entire application.

Program 2.15.11 : Write a program to accept and display name of user in AngularJS.

☑ Soln. : Program to accept and display the name of user in AngularJS

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<body>
<font size=5>
<div ng-app="">
 <form>
  First Name: <input type="text" ng-model="first_name">
 </form>
 <h1>Welcome : {{first_name}}</h1>
</div>
<p>Enter your name in the textfield</p>
</font>
</body>
</html>
```

Output



2. Checkbox

- Checkboxes are used to give list of items where multi selection is allowed.
- The ng-model directive can be applied to checkbox and the value of checkbox can be used anywhere in the application.
- The checkbox has value either true or false.

Program 2.15.12 : Write a program to demonstrate the use of checkbox in AngularJS.

☑ Soln. :

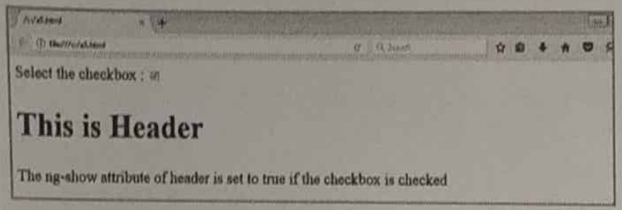
Demonstrating the use of checkbox in AngularJS.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<body>
```

Unit II In Sem.

```
<font size=5>
<div ng-app="">
 <form>
Select the checkbox :
  <input type="checkbox" ng-model="myModel">
 </form>
 <h1 ng-show="myModel">This is Header</h1>
</div>
<p>The ng-show attribute of header is set to true if the checkbox is checked</p>
</font>
</body>
</html>
```

Output



3. Radio button

☞ **Use :** Radio buttons are also used to give list of items but where single selection is allowed.

- The ng-model directive can be applied to radio button and the value of radio button can be used anywhere in the application.
- There may be different values of the radio buttons with same ng-model. The value of selected radio button is used.

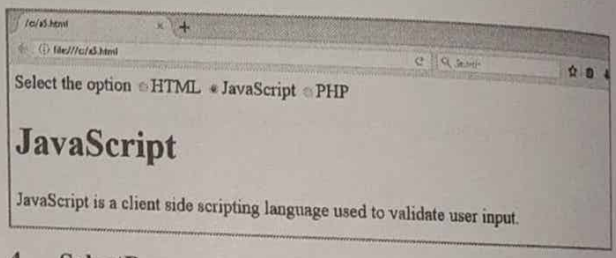
Program 2.15.13 : Write a program to display names of three languages using radio buttons. Display related information as per user's selection of language.

☑ **Soln. :**

Program displaying related information using radio buttons as per user's language selection.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<body ng-app="">
<font size=5>
<form>
Select the option
 <input type="radio" ng-model="myModel" value="HTML">HTML
 <input type="radio" ng-model="myModel" value="JavaScript">JavaScript
 <input type="radio" ng-model="myModel" value="PHP">PHP
</form>
<div ng-switch="myModel">
 <div ng-switch-when="HTML">
  <h1>HTML</h1>
  <p>HTML is a scripting language used to create static web pages.</p>
 </div>
 <div ng-switch-when="JavaScript">
  <h1>JavaScript</h1>
  <p>JavaScript is a client side scripting language used to validate user input.</p>
 </div>
 <div ng-switch-when="PHP">
  <h1>PHP</h1>
  <p>PHP is a server side scripting language used to create dynamic web pages.</p>
 </div>
</div>
</font>
</body>
</html>
```

Output



4. SelectBox

- SelectBox is also used to give list of items.
- The ng-model directive can be applied to SelectBox and the value of selected option can be used anywhere in the application.

Program 2.15.14 : Write a program to display names of three languages using SelectBox. Display related information as per user's selection of language.

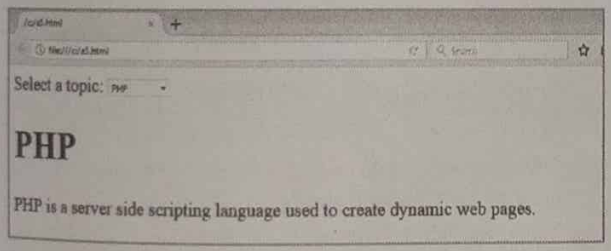
☑ **Soln. :**

Displaying the related information using select box as per user's selection of language

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
```

```
<body ng-app="">
<font size=5>
<form>
Select a topic:
<select ng-model="myModel">
  <option value="">
  <option value="HTML">HTML
  <option value="JavaScript">JavaScript
  <option value="PHP">PHP
 </select>
</form>
<div ng-switch="myModel">
 <div ng-switch-when="HTML">
   <h1>HTML</h1>
   <p>HTML is a scripting language used to create static web
pages.</p>
 </div>
 <div ng-switch-when="JavaScript">
   <h1>JavaScript</h1>
   <p>JavaScript is a client side scripting language used to validate
user input.</p>
 </div>
 <div ng-switch-when="PHP">
   <h1>PHP</h1>
   <p>PHP is a server side scripting language used to create
dynamic web pages.</p>
 </div>
</div>
</font>
</body>
</html>
```

Output



☞ Handling events in Form

- To associate with HTML elements, angularJs provides various events. For example the event ng-click is associated with the element button.
- Following is the list of various events provided by AngularJS

1. ng-click
2. ng-dbl-click
3. ng-keydown
4. ng-keyup
5. ng-keypress
6. ng-change
7. ng-mousedown
8. ng-mouseup
9. ng-mouseenter
10. ng-mouseleave
11. ng-mousemove
12. ng-mouseover

Program 2.15.15 : Write a program in AngularJS to reset the form data using on-click directive of a button.

☑ **Soln. : Reseting the form using on-click directive.**

```
<!DOCTYPE html>
<html lang="en">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/
1.6.4/angular.min.js"></script>
<body>
<font size=5>
<div ng-app="mainApp" ng-controller="formController">
 <form novalidate>
  First Name:<br>
  <input type="text" ng-model="user.first_Name"><br>
  Last Name:<br>
  <input type="text" ng-model="user.last_Name">
  <br><br>
  <button ng-click="reset()">Reset</button>
 </form>
```

Reset() function is called on click of button

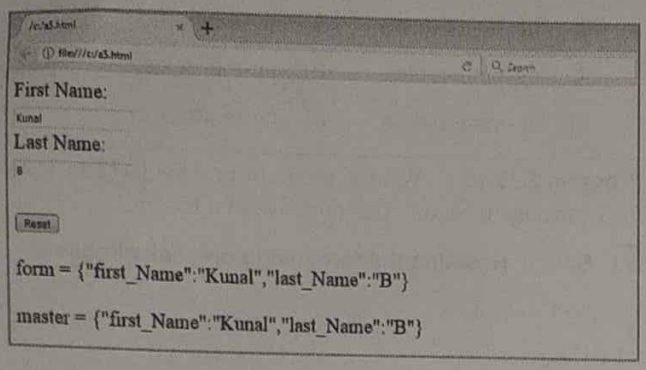
```
 <p>form = {{user}}</p>
 <p>master = {{master}}</p>
</div>
```

Displays data of user and master (Default)

```
<script>
var app = angular.module('mainApp', []);
app.controller('formController', function($scope) {
  $scope.master = {first_Name:"Kunal", last_Name:"B"};
  $scope.reset = function() {
    $scope.user = angular.copy($scope.master);
  };
  $scope.reset();
});
</script>
</font>
</body>
</html>
```

Copies data of master into user

Output



2.15.11 Includes

By default, HTML does not give any functionality to embed HTML page in another HTML page.

This can be achieved in AngularJS using the **ng-include** directive.

Syntax

```
<body ng-app="">
<div ng-include="'myFile.htm'"></div>
</body>
```

Program 2.15.16 : Write a program to embed HTML file in another HTML file using AngularJS.

Soln. : Embeding one HTML file into another HTML file.

- First we will create an HTML file.
- The HTML file which is included with the ng-include directive, can also have AngularJS code :

Data.html

```
<table border = "1">
    <tr>
        <td>Enter First Name:</td>
        <td><input type = "text"
        ng-model =     "stud.first_Name"></td>
    </tr>
    <tr>
    <td>Enter Last Name: </td>
    <td>
    <input type = "text" ng-model = "stud.last_Name">
    </td>
    </tr>
    <tr>
    <td>Name: </td>
    <td>{{stud.full_Name()}}</td>
    </tr>
</table>
```

Subject.html

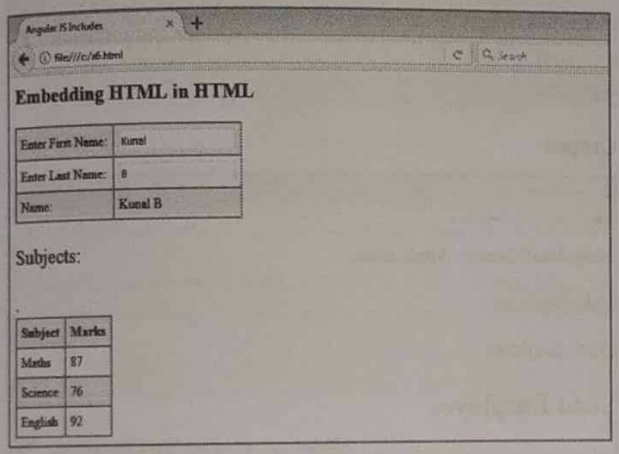
```
    <p>Subjects:</p>
    <table>
    <tr>
        <th>Subject</th>.
        <th>Marks</th>
        </tr>
        <tr ng-repeat = "sub in stud.sub">
        <td>{{ sub.name }}</td>
        <td>{{ sub.marks }}</td>
        </tr>
    </table>
```

main.html

```
<html>
    <head>
    <title>Angular JS Includes</title>
<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/
angular.min.js"></script>
    <style>
    table, th , td {
    border: 2px solid grey;
    border-collapse: collapse;
    padding: 6px;
    }
    table tr:nth-child(odd) {
    background-color: #f2f2f2;
    }
    table tr:nth-child(even) {
    background-color: #ffffff;
    }
    </style>
    </head>
    <body>
    <h2>Embedding HTML in HTML</h2>
    <font size=5>
    <div ng-app = "mainApp"
    ng-controller="studentController">
    <div ng-include = "'data.html'"></div>
    <div ng-include = "'subject.html'"></div>
    </div>
    <script>
    var mainApp = angular.module("mainApp", []);
mainApp.controller('studentController', function($scope) {
$scope.stud = {
    first_Name: "Kunal",
    last_Name: "B",
    fees:500,
    sub:[
```

```
{name:'Maths',marks:87},
{name:'Science',marks:76},
{name:'English',marks:92},
],
full_Name: function() {
var studentObject;
studentObject = $scope.stud;
return studentObject.first_Name + " " +
studentObject.last_Name;
}
};
});
</script>
</font>
</body>
</html>
```

Output



2.15.12 Views

GQ. Explain Views in AngularJS. **(4 Marks)**

- It is possible to create single page application through multiple views in AngularJS.
- For this purpose AngularJS provides directives like ng-view and ng-template and service like $routeProvider.

☞ ng-view

- This tag is used to create place holder for the corresponding view (html or ng-template view).
- This view is placed based on the configuration.

☞ Usage

In the main module a div can be created with ng-view.

```
<div ng-app = "myApp">
  ...
  <div ng-view></div>
</div>
```

☞ ng-template

This directive creates the html view with the help of <script> tag. The "id" attribute is used to map a view with a controller which is done by $routeProvider.

☞ Usage

In the main module, a script tag is created with type ng-template.

```
<div ng-app = "myApp">
  ...
  <script type = "text/ng-template" id = "addEmp.htm">
    <h2> Add Employee </h2>
    {{message}}
  </script>
</div>
```

☞ $routeProvider

$routeProvider service is used to set the configuration of URLs, It maps the URLs with the subsequent HTML page or ng-template, and also attach a controller with them.

☞ Usage

In the main module script, block is defined and routing configuration is set.

```
var myApp = angular.module("myApp", ['ngRoute']);
    myApp.config(['$routeProvider', function($routeProvider) {
      $routeProvider.
      when('/addEmp', {
        templateUrl: 'addEmp.htm',
        controller: 'AddEmpController'
      }).
      when('/viewEmp', {
        templateUrl: 'viewEmp.htm',
        controller: 'ViewEmpController'
      }).
      otherwise({
        redirectTo: '/addemp'
      });
    }]);
```

Unit II In Sem.

Some important points to note

- $routeProvider is defined as a method (function) under config of myApp module by setting the key as '$routeProvider'.
- The url "/addEmp" is mapped to "addEmp.htm" when it is defined by the $routeProvider. Here the main page and "addEmp.htm" are considered at the same location.
- The use of "controller" is to set the related controller for the view.

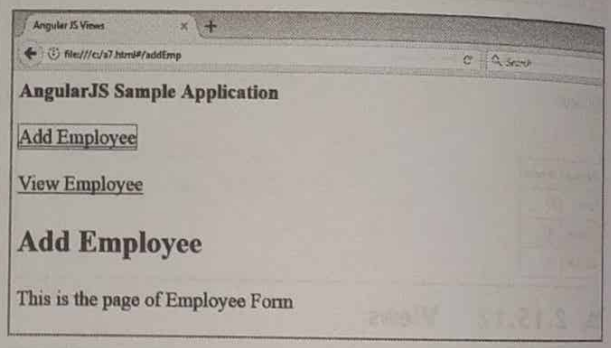
Program 2.15.17 : Write a program to demonstrate the use of views in AngularJS.

Soln. :

Program to demonstrate the use of views in AngularJS

```
<html>
    <head>
    <title>Angular JS Views</title>
     <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/
angular.min.js"></script>
    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/
angular-route.min.js"></script>
  </head>
    <body>
    <h2>AngularJS Sample Application</h2>
<font size=5>
    <div ng-app = "myApp">
      <p><a href = "#addEmp">Add Employee</a></p>
      <p><a href = "#viewEmp">View Employee</a></p>
      <div ng-view></div>
      <script type = "text/ng-template" id = "addEmp.htm">
        <h2> Add Employee </h2>
        {{message}}
      </script>
      <script type = "text/ng-template"
id = "viewEmp.htm">
        <h2> View employee </h2>
        {{message}}
      </script>
    </div>
      <script>
     var myApp = angular.module("myApp", ['ngRoute']);
     myApp.config(['$routeProvider', function($routeProvider) {
      $routeProvider.
      when('/addEmp',
{
        templateUrl: 'addEmp.htm',
        controller: 'AddEmpController'
      }).
      when('/viewEmp',
{
        templateUrl: 'viewEmp.htm',
        controller: 'ViewEmpController'
      }).
      otherwise(
{
        redirectTo: '/addemp'
      });
    }]);
    myApp.controller('AddEmpController', function($scope) {
      $scope.message = "This is the page of Employee Form";
    });
    myApp.controller('ViewEmpController', function($scope) {
      $scope.message = "This is the page displaying records of all
the employees";
    });
    </script>
    </font>
  </body>
</html>
```

Output



2.15.13 Scopes

GQ. Write short note on scopes in AngularJS. **(4 Marks)**

Definition : The Scope is an object which works as a binding part in between the view (HTML) and the controller (JavaScript). It joins the controller with views.

- The scope object has properties and methods.
- The model data is available in the scope. This model data is accessed in controllers through $scope object.

```
<script>
    var myApp = angular.module("myApp", []);
    myApp.controller("locationController", function($scope) {
```

```
    $scope.message = "In Location Controller";
    $scope.type = "Location";
    });
</script>
```

☞ Important points to note

- In the definition of constructor in controller, the first argument passed is $scope.
- The models $scope.message and $scope.type are used in the HTML documents.
- Here the models are set with values. This will be reflected in the application module in which the controller is locationController.
- Functions can also be defined in $scope.

☞ Scope Inheritance

Scopes are usually controller specific. When nested controllers are defined, the scope of parent controller is inherited by the child controller.

```
<script>
    var myApp = angular.module("myApp", []);

    myApp.controller("locationController", function($scope) {
      $scope.message = "In Location Controller";
      $scope.type = "Location";
    });

    myApp.controller("countryController", function($scope) {
      $scope.message = "In Country Controller";
    });
</script>
```

☞ Important points to note

- Here the values are set to models in locationController.
- There is overridden message in child controller countryController. If we use the "message" within module of the controller countryController, then the overridden message will be displayed.

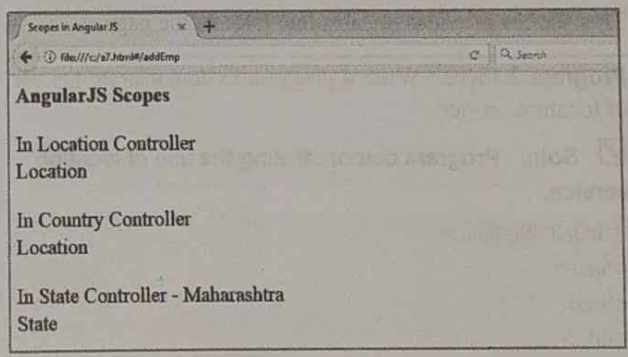
Program 2.15.18 : Write a program to display the data of Location, country and state to show the scope of controllers.

☑ **Soln. :**

Scope of controllers showing data of location, country and state

```
<html>
  <head>
    <title>Scopes in Angular JS</title>
  </head>
  <body>
    <h2>AngularJS Scopes</h2>
    <font size=5>
    <div ng-app = "myApp"
ng-controller = "locationController">
      <p>{{message}} <br/> {{type}} </p>
      <div ng-controller = "countryController">
        <p>{{message}} <br/> {{type}} </p>
      </div>
      <div ng-controller = "stateController">
        <p>{{message}} <br/> {{type}} </p>
      </div>
    </div>
    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/
angular.min.js"></script>
    <script>
      var myApp = angular.module("myApp", []);
      myApp.controller("locationController", function($scope) {
        $scope.message = "In Location Controller";
        $scope.type = "Location";
      });
      myApp.controller("countryController", function($scope) {
        $scope.message = "In Country Controller";
      });
      myApp.controller("stateController", function($scope) {
        $scope.message = "In State Controller - Maharashtra";
        $scope.type = "State";
      });
    </script>
    </font>
  </body>
</html>
```

Output



2.15.14 Services

GQ. Explain different Services in AngularJS. **(8 Marks)**

Unit II In Sem.

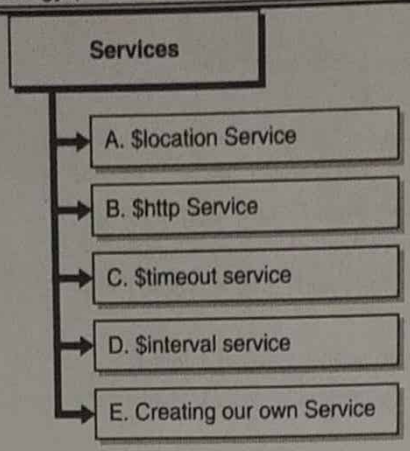


Fig. 2.15.5 : Services in AngularJS

- An AngularJS, Services are the JavaScript functions which are used to perform specific tasks. Services can be maintained and tested independently.
- Services provide a modular approach to AnglarJS.
- The services can be called by the Controllers and Filters as per requirement.
- The concept of dependency injection is used to inject the services.
- There are near about 30 services in AngularJS like $location, $route, $https, $window etc. These services perform various tasks.

☞ Examples

- **$https :** This service makes ajax call to obtain the server data.
- **$route :** This service defines the routing information.
- All the inbuilt services are prefixed with $ symbol.
- Now we will see some examples on inbuilt services.

▶ A. $location Service

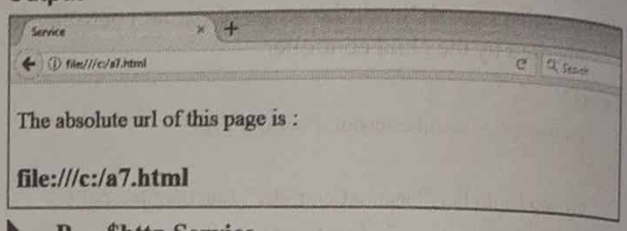
This service provides the absolute URL of the page.

Program 2.15.19 : Write a program to demonstrate the use of location service.

☑ Soln. : Program demonstrating the use of location service.

```
<!DOCTYPE html>
<html>
<head>
<title>
Service
</title>
</head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/
angular.min.js"></script>
<body>
<font size=5>
<div ng-app="mainApp" ng-controller="myController">
<p>The absolute url of this page is :</p>
<h3>{{theUrl}}</h3>
</div>
<script>
var app = angular.module('mainApp', []);
app.controller('myController', function($scope, $location)
 {
   $scope.theUrl = $location.absUrl();
});
</script>
</font>
</body>
</html>
```

Output



▶ B. $http Service

In AngularJS applications, this service is considered as one of the most commonly used services. A request is made by this service to the server and the application will handle the response. $http service is used to request data from the server:

```
var app = angular.module('mainApp', []);
app.controller('myController', function($scope, $http)
{
  $http.get("welcomePage.htm").then(function (response) {
    $scope.myWelcome = response.data;
  });
});
```

▶ C. $timeout service

This service is version of the window.setTimeout() function. The $timeout service runs a function after a specified number of milliseconds.

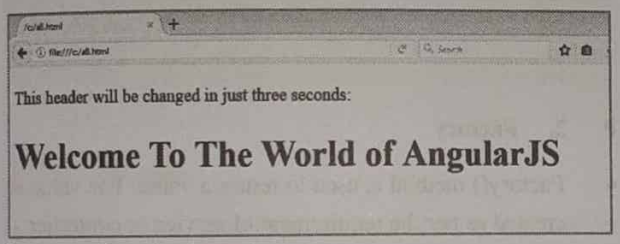
Program 2.15.20 : Write a program to show the use of timeout service.

☑ Soln. : Program showing the timeout service

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/
```

```
1.6.4/angular.min.js"></script>
<body>
<div ng-app="mainApp" ng-controller="myController">
<font size=5>
<p>This header will be changed in just three seconds:</p>
<h1>{{txtHeader}}</h1>
</div>
<script>
var app = angular.module('mainApp', []);
app.controller('myController', function($scope, $timeout) {
 $scope.txtHeader = "Welcome To The World of AngularJS";
 $timeout(function () {
    $scope.txtHeader = "Hello Friends ! How are you?";
 }, 3000);
});
</script>
</font>
</body>
</html>
```

Output



D. $interval service

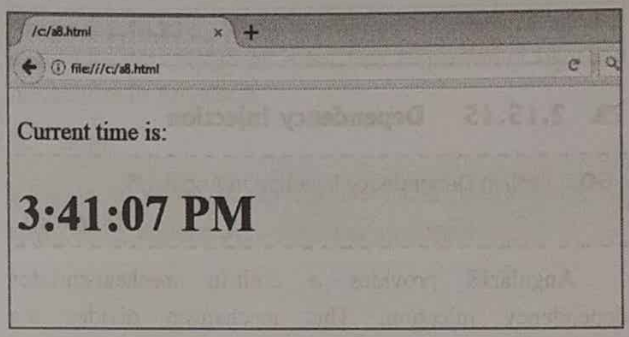
This service is version of the window.setInterval() function. The $interval service runs a function after every specified milliseconds.

Program 2.15.21 : Write a script to show the use of interval service.

☑ Soln. : Script showing the use of interval service

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/
1.6.4/angular.min.js"></script>
<body>
<div ng-app="mainApp" ng-controller="myController">
<font size=5>
<p>Current time is:</p>
<h1>{{currentTime}}</h1>
</div>
<script>
var app = angular.module('mainApp', []);
app.controller('myController', function($scope, $interval) {
 $scope.currentTime = new Date().toLocaleTimeString();
 $interval(function ()
{
    $scope.currentTime = new Date().toLocaleTimeString();
 }, 1000);
});
</script>
</font>
</body>
</html>
```

Output



E. Creating our own Service

GQ. How to create user defined Service ? Explain with program. **(2 Marks)**

We can create our own service. This can be added as dependency while defining the controller.

Program 2.15.22 : Write a program to create the own service.

☑ Soln. :

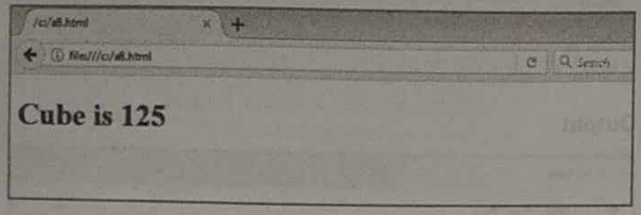
Program creating the own service

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/
1.6.4/angular.min.js"></script>
<body>
<div ng-app="mainApp" ng-controller="myController">
<h1>Cube is {{cb}}</h1>
</div>
<script>
var app = angular.module('mainApp', []);
app.service('cube', function() {
  this.myFunc = function (x) {
    return x*x*x;
  }
});
```

Unit II In Sem.

```
app.controller('myController', function($scope, cube) {
 $scope.cb = cube.myFunc(5);
});
</script>
</body>
</html>
```

Output



2.15.15 Dependency Injection

GQ. Explain Dependency Injection in AngularJS. **(8 Marks)**

AngularJS provides a built-in mechanism for dependency injection. This mechanism divides the application into various types of components. As a dependency, these components can be injected into each other.

Dependency Injection is a pattern of software design which indicates that; how the components obtain holds of their dependencies. In dependency injection the components are assigned their dependencies rather than coding them within the component. When an application is breaks into modules, it becomes easier to reuse, configure and testing the application components.

Important types of objects and components

Following is list of important types of objects and components :

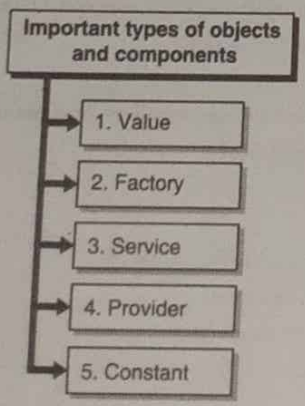


Fig. 2.15.6 : Types of objects and components

1. Value

- Value can be a number, string or JavaScript object. In the config phase, this object is used to pass values to controller.
- The value object can also be used to pass values in factories or services during the run and config phase.

```
// module is defined
    var myApp = angular.module("myApp", []);
// value object is created as "defaultInput" and pass it a data.
    myApp.value("defaultInput", 5);
...
//using its name "defaultInput", value is injected in the controller
    myApp.controller('myController', function($scope, CalcService,
defaultInput) {
     $scope.number = defaultInput;
     $scope.txtresult = CalcService.cube($scope.number);
$scope.cube = function() {
$scope.txtresult = CalcService.cube($scope.number);
     }
    });
```

2. Factory

- Factory() method is used to return a value. The value is created as per the requirement of service or controller.
- Now we will see how to define factory function on a module.

```
//define a module
    var myApp = angular.module("myApp", []);
// A factory "MathService" is created which gives a method multiply to
calculate and return multiplication //of three numbers
    myApp.factory('MathService', function() {
     var factory = {};
     factory.multiply = function(x, y, z) {
      return x * y * z;
     }
     return factory;
    });
// the factory "MathService" is injected in the factory to use the
multiply method of factory.
    myApp.service('CalcService', function(MathService){
     this.cube = function(x) {
      return MathService.multiply(x,x,x);
     }
    }); ...
```

3. Service

Service is also a JavaScript object which contains a set of methods to perform specific tasks. The service() method is used to define the services. These services can be then injected into controllers.

```
//define a module
    var myApp = angular.module("myApp", []);

...
// A service is created to define the method cube to return cube of
given number
myApp.service('CalcService', function(MathService){
      this.cube = function(x) {
        return MathService.multiply(x,x,x);
      }
    });
//inject the service "CalcService" into the controller
myApp.controller('myController', function($scope, CalcService,
defaultInput) {
      $scope.number = defaultInput;
      $scope.txtresult = CalcService.cube($scope.number);
      $scope.cube = function() {
    $scope.txtresult = CalcService.cube($scope.number);
      }
    });
```

4. Provider

- The services,factory etc. are created using the provider in config phase.
- In the following example, we have created a service MathService. Provider is a particular factory method. It has method named get() which returns the value or factory or service.

```
// module is defined
    var myApp = angular.module("myApp", []);

...
//With the help of provider, service is created which defines the
cube() method to return cube of number.
myApp.config(function($provide) {
      $provide.provider('MathService', function() {
        this.$get = function() {
          var factory = {};
          factory.multiply = function(x, y, z) {
            return x * y * z;
          }
          return factory;
        };
      });
    });
```

5. Constant

Use

In config phase, the values are passed by the **constants**

```
myApp.constant("configParam", "constant_value");
```

Program 2.15.23 : Write a program to accept a number from user and display its cube. Use Dependency Injection mechanism in AngularJS.

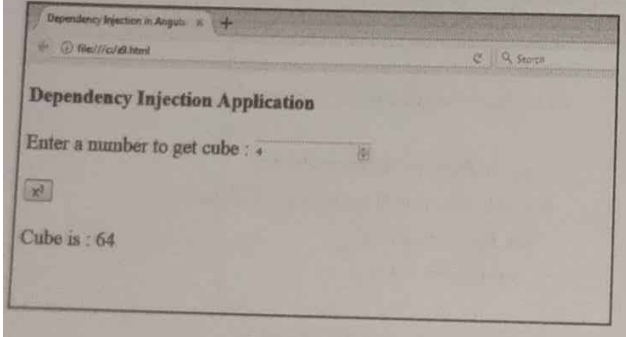
Soln. :

Program to accept a number from user and display its cube using dependency injection mechanism

```
<!DOCTYPE html>
<html>
  <head>
    <title>Dependency Injection in AngularJS</title>
  </head>
  <body>
    <h2>Dependency Injection Application</h2>

<font size=5>
    <div ng-app = "myApp" ng-controller = "myController">
      <p>Enter a number to get cube : <input type = "number" ng-
model = "number" /></p>
      <button ng-click = "cube()">X<sup>3</sup></button>
      <p>Cube is : {{txtresult}}</p>
    </div>

    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14
/angular.min.js"></script>

    <script>
      var myApp = angular.module("myApp", []);

      myApp.config(function($provide) {
        $provide.provider('MathService', function() {
          this.$get = function() {
            var factory = {};

            factory.multiply = function(x, y, z) {
              return x * y * z;
```

```
        }
        return factory;
      };
    });
  });

  myApp.value("defaultInput", 5);

  myApp.factory('MathService', function() {
    var factory = {};

    factory.multiply = function(x, y, z) {
      return x * y * z;
    }
    return factory;
  });

  myApp.service('CalcService', function(MathService){
    this.cube = function(x) {
      return MathService.multiply(x,x,x);
    }
  });

  myApp.controller('myController', function($scope, CalcService,
defaultInput) {
    $scope.number = defaultInput;
    $scope.txtresult = CalcService.cube($scope.number);

    $scope.cube = function() {
      $scope.txtresult = CalcService.cube($scope.number);
    }
  });
  </script>
</font>
  </body>
</html>
```

Output



2.15.16 Custom Directives

GQ. Write short note on custom directives in AngularJS. **(4 Marks)**

☞ **Use :** The functionality of HTML can be extended using the custom directives. The directive() function is used to define the custom directive. The elements for which the custom directives are activated are replaced by the custom directives.

- During the bootstrap, the matching elements are searched and one time activity is done with the help of method compile() of the custom directive. Then using link() method of custom directive the elements are processed based on the scope of directive.

Following types of elements can be processed in AngularJS

1. Element directives
2. Attributes
3. CSS
4. Comment

Program 2.15.24 : Write a program to demonstrate use of custom directives in AngularJS.

☑ **Soln. :**

Program demonstrating the use of custom directives in AngularJS.

```
<html>
  <head>
  <title>Custom Directives in AngularJS </title>
 </head>
 <body>
  <h2>Custom Directives </h2>
<font size=5>
  <div ng-app = "myApp"
ng-controller = "empController">
    <emp name = "Ishita"></emp><br/>
    <emp name = "Kunal"></emp>
  </div>
  <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/
angular.min.js"></script>
  <script>
    var myApp = angular.module("myApp", []);
    myApp.directive('emp', function() {
```

Create a directive, first parameter is the html element to be attached. We are attaching emp html tag. This directive will be activated as soon as any emp element is encountered in html

```
var directive = {};
```
Defines the directive object

```
directive.restrict = 'E';
```
Signifies that directive is Element directive

```
directive.template = "Employee: <b>{{emp.name}}</b> ,
ID : <b>{{emp.ID}}</b>";
```
Template replaces the complete element with its text

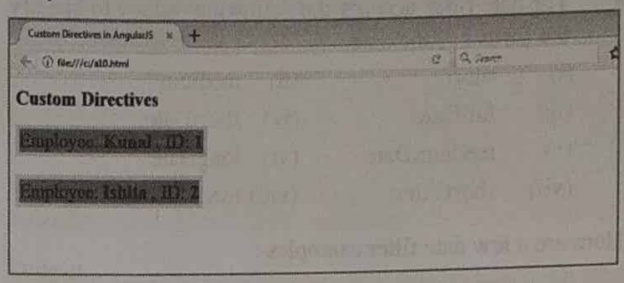
Scope is used to distinguish each emp element based on criteria

```
directive.scope = {
    emp : "=name"
}

directive.compile = function(element, attributes) {
```
Compile is called at the time of application initialization

```
element.css("border", "5px solid #cccccc");

var linkFunction = function($scope, element, attributes)
{
```
linkFunction is used to link each element with scope to get the element specific data

```
element.html("Employee: <b>"+$scope.emp.name
+"</b> , ID: <b>"+$scope.emp.ID+"</b><br/>");
        element.css("background-color", "grey");
    }
    return linkFunction;
}

return directive;
});
myApp.controller('empController', function($scope) {
    $scope.Ishita = {};
    $scope.Ishita.name = "Kunal";
    $scope.Ishita.ID = 1;
    $scope.Kunal = {};
    $scope.Kunal.name = "Ishita";
    $scope.Kunal.ID = 2;
});
</script>
</font>
</body>
</html>
```

Output



2.15.17 Internationalization

GQ. What is Internationalization? Explain various types of filters which support Internationalization. **(4 Marks)**

❒ **Definition :** Internationalization is the process which enhances the planning and implementing of specific products and services so that they should be adapted easily to specific local languages and cultures. The process of internationalization is also called as translation or localization enablement.

- There is situation for each and every software when internationalization becomes compulsory. Sometimes it happens because of regional specific clients or when the software has to be delivered to users in many countries.
- In AngularJS, some built-in filters support the concept of internationalization. The data, currency and number filters have built-in support in AngularJS for internationalization.

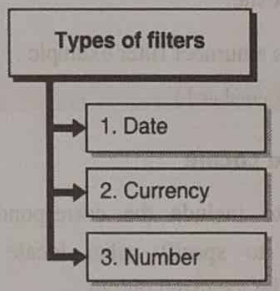


Fig. 2.15.7 : Types of filters

See how to use these filters

```
{{ theBDate | date: 'fullDate' }}
{{ theFees | currency }}
{{ theMarks | number }}
```

Unit II In Sem.

The BDate variable is formatted by the date filter according to the locale (language and country) which is selected in the web app. Same is applied for the the Fees and the Marks variables.

1. The Date Filter

- The date filter accepts the following values to specify the format of the date :

(i)	short	(ii)	medium
(iii)	fullDate	(iv)	shortDate
(v)	mediumDate	(vi)	longDate
(vii)	shortTime	(viii)	mediumTime

Here are a few date filter examples :

```
{{ theBDate | date: 'shortDate' }}
{{ theBDate | date: 'longDate' }}
```

2. The Currency Filter

- To format the currency, the currency symbol which is associated with the selected locale is used by the currency filter.
- In case if we want to use any other symbol, then we have to specify the currency symbol as follows :

```
{{ theFees | currency : '$' }}
{{ theFees | currency : '£' }}
{{ theFees | currency : ₹ }}
```

3. The Number Filter

- This is used to format the numerical values according to the selected locale.
- Now see, in English separator for the thousand is "." and the decimal separator is "," whereas in Danish it is exactly opposite.

Following is a number filter example :

```
{{ theRollno | number }}
```

Setting the Locale

- We have to include the corresponding AngularJS locale file to specify what locale to use while generating the localized output.
- Here is an example that includes the Danish locale :

```
<script src="https://code.angularjs.org/1.2.5/i18n/
angular-locale_da-dk.js"></script>
```

Include this after including the AngularJS main JavaScript file.

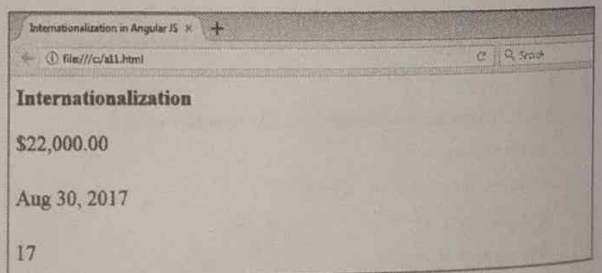
Program 2.15.25 : Write a program to demonstrate use of Danish Locale.

Soln. :

Program demonstrating the use of Danish locale

```
<html>
  <head>
    <title>Internationalization in Angular JS </title>
  </head>
  <body>
    <h2>Internationalization</h2>
<font size=5>
    <div ng-app = "myApp"
ng-controller = "StudController">
      {{theFees | currency }}  <br/><br/>
      {{theAdmDate | date }}  <br/><br/>
      {{theRollno | number }}
    </div>
    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/
angular.min.js"></script>
    <script src = "https://code.angularjs.org/1.3.14/i18n/
     angular-locale_da-dk.js"></script>
    <script>
      var myApp = angular.module("myApp", []);
      myApp.controller('StudController', function($scope)
          {
        $scope.theFees = 22000;
        $scope.theAdmDate  = new Date();
        $scope.theRollno = 17;
      });
    </script>
    </font>
  </body>
</html>
```

Output

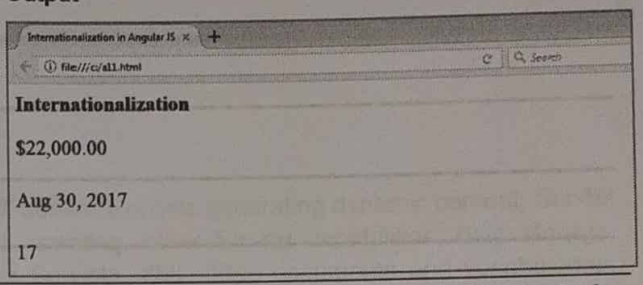


Program 2.15.26 : Write a program to demonstrate use of Browser Locale.

Soln. : Program to demonstrate the use of browser locale

```
<html>
 <head>
  <title>Internationalization in Angular JS </title>
 </head>
 <body>
  <h2>Internationalization</h2>
<font size=5>
  <div ng-app = "myApp"
  ng-controller = "StudController">
   {{theFees | currency }}  <br/><br/>
   {{theAdmDate | date }}  <br/><br/>
   {{theRollno | number }}
  </div>
  <script src = "https://ajax.googleapis.com/ajax/libs/
   angularjs/1.3.14/angular.min.js"></script>
  <!-- <script src = "https://code.angularjs.org/1.3.14/i18n/
angular-locale_da-dk.js"></script> -->
  <script>
   var myApp = angular.module("myApp", []);
   myApp.controller('StudController', function($scope)
   {
    $scope.theFees = 22000;
    $scope.theAdmDate  = new Date();
    $scope.theRollno = 17;
   });
  </script>
  </font>
 </body>
</html>
```

Output



Chapter Ends...

❑❑❑

UNIT III

CHAPTER 3

Java Servlets and XML

Syllabus

Servlet: Servlet architecture overview, A "Hello World" servlet, Servlets generating dynamic content, Servlet life cycle, parameter data, sessions, cookies, URL rewriting, other Servlet capabilities, data storage, Servlets concurrency, databases (MySQL) and Java Servlets. XML: XML documents and vocabularies, XML declaration, XML Namespaces, DOM based XML processing, transforming XML documents, DTD: Schema, elements, attributes. AJAX: Introduction, Working of AJAX.

3.1 INTRODUCTION TO SERVLET

UQ. What is a Servlet? SPPU - Dec. 2014, 2 Marks

- **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.
- **Use :** 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.

3.1.1 Need of Servlet

GQ. Explain the need of Servlet. (4 Marks)

- Servlet is considered as an integral part in J2EE web applications. The server side component of a Servlet provides a great mechanism for development of server side web applications. It plays significant role in the explosion of Internet with the help of its reusability, performance and scalability.
- We can create server side programs or applications fast and efficiently by using servlets.
- The major benefit of using servlets over CGI is, the CGI scripts run outside the web server because of which every time a new process should be started before running CGI programs.
- CGI scripts cannot handle multiple requests at a time. When execution gets completed, the CGI script returns the result in the web server and exit. Servlets can handle multiple requests at a time. Dynamic content is generated by the Servlets which is easily written and fast executed within web servers.
- The access to any J2SE and J2EE APIs is available to Servlet and it can take advantage of powerful JDK of Java.
- Servlets are the server side scripts which are component based, platform independent which creates the web based applications without the limitations of performance like CGI programs.

3.1.2 Advantages of Servlet

UQ. What are the advantages of Servlet? SPPU - Dec. 2014, 4 Marks

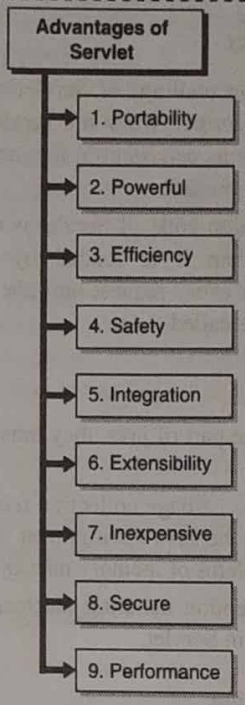


Fig. 3.1.1 : Advantages of Servlet

1. Portability

- Servlets are written entirely in java. As Servlets are part of java, they are portable on the basis of operating systems and server implementations.
- Servlets follows the Java Rule **"Writing Once, Run Anywhere" (WORA)**, because it is possible to develop a Servlet on Windows machine with the help of tomcat server or any other server and later on it can be deployed on any other operating system such as UNIX.
- As Servlets are tremendously portable they can be executed in any platform. Hence servlets are platform independent.

Unit III End Sem.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. Inexpensive

Servlet are executed with the help of web servers which are available without any cost for personal as well as commercial purpose.

8. 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.

9. Performance

- As compare to CGI, Servlets are relatively very faster because in CGI for every script, a new process is generated which takes a lot much 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 much 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.

3.1.3 Disadvantages of Servlet

UQ. What are the disadvantages of servlets?
SPPU - Dec. 2014, 4 Marks

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.

3.2 ARCHITECTURE OF SERVLET

UQ. How do servlets work?
SPPU - Dec. 2014, 4 Marks

GQ. Explain the architecture of Servlet. **(4 Marks)**

- 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

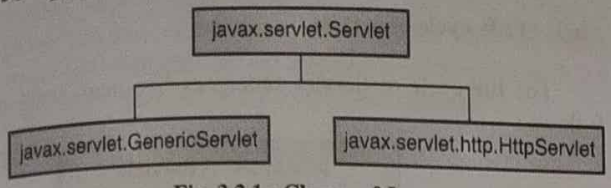


Fig. 3.2.1 : Classes of Java

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:

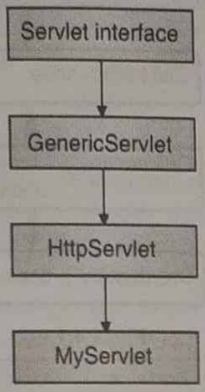


Fig. 3.2.2 : Architecture of package

(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.

(i) Client Interaction (ii) Servlet Execution

(i) Client Interaction

- When a Servlet accepts a call from a client, it receives two objects:

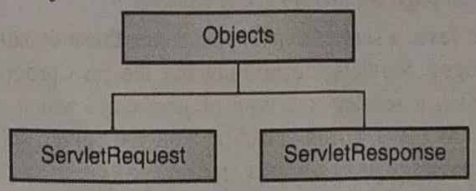


Fig. 3.2.3 : Objects

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.

Unit III End Sem.

(ii) Servlet Execution

- Fig 3.2.4 shows the execution of Servlet when client (browser) makes a request to the web server.

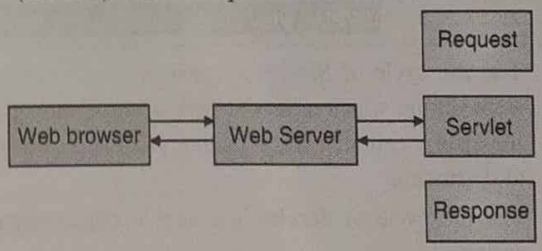


Fig. 3.2.4 : Servlet Execution

3.3 SERVLETS GENERATING DYNAMIC CONTENT

- Dynamic web pages are server-side web pages, each time it is viewed, we see different content. It is controlled by Application server processing server-side scripts.
- The dynamic web pages can also change their contents on the request of the client. They have the capability to generate new content according to time and need.

Which simply means that dynamic web pages are never the same for all users.

- We all are well aware of the need for dynamic web pages in day to day life. The best example of a dynamic web page we always see is captcha.
- In Java, a servlet is a way to create those dynamic web pages. Servlets are nothing but the java programs. In Java, a servlet is a type of java class which runs on JVM (java virtual machine) on the server side. Java servlets works on server side.
- Java servlets are able to handle large and complex problems and requests by users.
- Servlets generate dynamic content by responding to web clients. When an HTTP request is received by the application server, the web server determines based on the request URI, which servlet is responsible for answering that request and forwards the request to that servlet.
- The servlet then performs its logic and builds the response HTML that is returned back to the web client.

Steps for generating dynamic Web pages using servlet

(1) A client sends the request to a web server.
(2) The web server receives the request from client.
(3) Servlets receives the request.
(4) Servlets process the request and produce the output.
(5) Servlet sends the output to the web server.
(6) A web server sends it to the client's browser and browser display it on the client's screen.

3.4 LIFE CYCLE OF SERVLET

UQ. Explain the lifecycle of Servlet.
SPPU - May 2016, Dec. 2016, 6 Marks

- The life cycle of Servlet is considered as a series of steps during which a Servlet comes across through its life span, beginning from loading process till last step of destroying.
- The life cycle of Servlet is a simple object oriented design.
- Before going to life cycle of Servlet, we will discuss some technologies to understand the things better.
  (1) **Web Server :** This server is also called as HTTP Server. The main functionality of web server is to handle the requests means to accept the client requests and generates the response.
  (2) **Web Container :** This is also called as Servlet Container or Servlet Engine. It is an integral part of web server which communicates with the Servlets. The life cycle of Servlet is managed by the Web Container.

☞ Life Cycle of Servlet

Steps of life cycle of servlet

The life cycle of Servlet undergoes five main steps as follows :

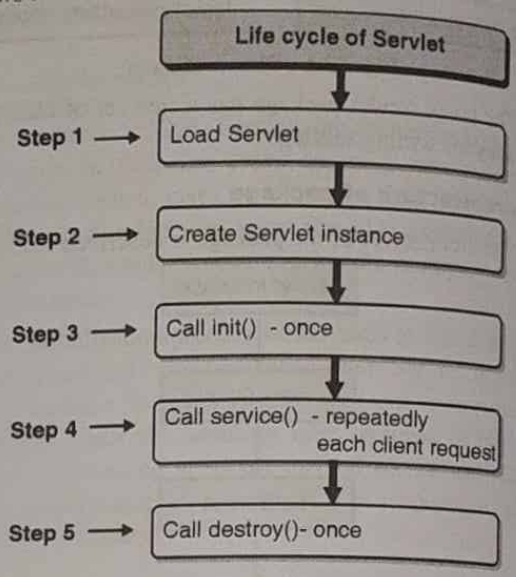


Fig. 3.4.1

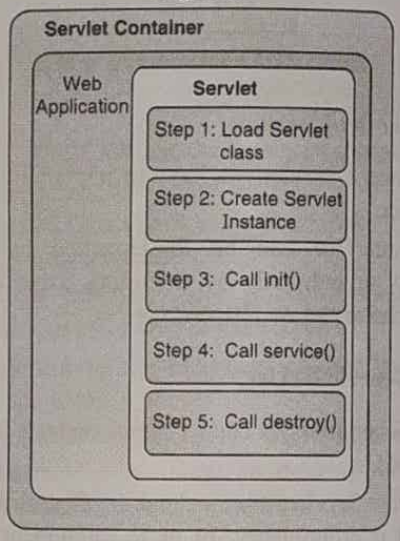


Fig. 3.4.2 : Life cycle of Servlet

Step 1 : Load Servlet

When the service of web server like Apache Tomcat is started, the Servlet is deployed and loaded by Servlet Container. Before invoking a Servlet, the Servlet Container should load its class definition first.

Step 2 : Creating instance of Servlet

As soon as the Servlet class is loaded, an instance of Servlet class is created by the Servlet Container. For every Servlet, only a single instance is created which handles all the client requests.

Step 3 : Call init() method

The init() method is considered as the beginning of Servlet's life cycle. As soon as the Servlet class is instantiated, the

server calls the init() method for instantiated Servlet. This method helps to initialize the Servlet. The init() method is called only once in the life cycle of Servlet. Here the server crates and initializes all the necessary resources which are required for Servlet to handle the client requests.

☞ Signature of init() method

The signature of init() method is as follows:

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

- In the init() method, the Servletconfig object is created as a parameter. This object can be saved to reference later.
- The init() method throws the ServletException if because of some reason the Servlet cannot initialize the resources which are necessary to handle the client requests.

▸ Step 4 : Call service() method

- Whenever the web server receives a client request for Servlet, it initiates a new thread which invokes service() method. That means it get called multiple times in the Servlet's life cycle. All the client requests are handled by the service() method. It is not possible to call the service() method before the execution of init() method.
- When the Servlet is extended from the GenericServlet class then the request is handled by the service() method itself, but if the Servlet is extended from HttpServlet then service() method receives the request and sends it to the correct handler method which depends upon the type of request.
- **Example :** If the request is Get Request then the service() method sends the request to the doGet() method by invoking the doGet() method with request parameters. In the same manner, for the requests such as Post, Head, Put etc. request is send by the service() method to the related method handlers like doPost(), doHead(), doPut() etc.

☞ Signature of service() method

The signature of service() method is as follows:

```
public void service(ServletRequest Request, ServletResponse
Response) throws ServletException, IOException;
```

- The service method has two parameters **ServletRequest** and **ServletRsponse**. The Servlet request object holds the information about the service request and ServletResponse object stores the data which we want to send back to client.

▸ Step 5 : Call destroy() method

- This method indicates the end of Servlet's life. When Servlet container shuts down which is generally happens when web server is stopped, it unloads all the running servlets from the memory and calls destroy() method for all the initialized servlets.
- This is where all the resources created by the init() method are cleaned up. The non-java resources like file handle or database connections if opened in the application can be closed here.

☞ Signature of destroy() method

- The signature of destroy() method is as follows :

```
Public void destroy();
```

⏭ 3.5 CREATING AND TESTING SAMPLE SERVLET

3.5.1 Simple Servlet – Example/Program

Program 3.5.1 **SPPU - May 2016, Dec. 2016, 4 Marks**

Write an example (program) of Servlet.

☑ **Soln. :**

Simple Servlet example printing welcome message

```
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();
  }
}
```

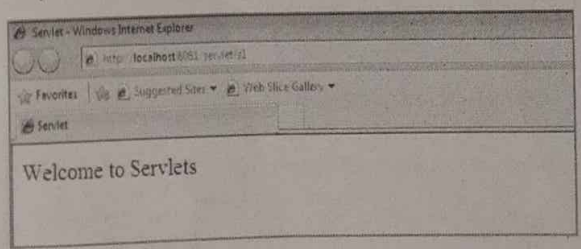
Packages are imported

Inherited from GenricServlet hence definition of Service() is must

Object to use HTML tags and print message

Here the Servlet class is inherited from GenericServlet, hence it is necessary for us to define the service method.

Output



Unit III End Sem.

3.5.2 Client Server Application – Example/Program

Program 3.5.2 SPPU - May 2017, 8 Marks

How can we read client data using Servlet?

OR

Create HTML page which should accept name from user. Send it to Servlet. The Servlet should display welcome message with the given name.

☑ **Soln. :**

Displaying welcome message accepting the name from user

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.

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();
  }
}
```

"n" is name of text field created in HTML

Explanation of Servlet code (s2.class file)

response.setContentType("text/html"); -

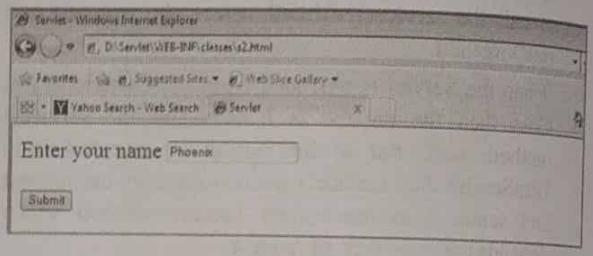
The setContentType() is method of ServletResponse. It is used to set the type of content which we will return to the browser on client side.

String nm = request.getParameter("n"); -

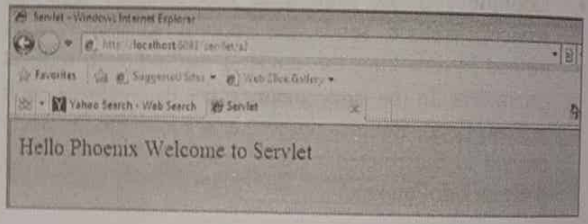
The getParameter() is method of ServletRequest. It takes argument as name of HTML form control and returns data provided by user in it or the status of form control like checked or unchecked. Here the "n" is the name of text field in HTML form.

Output

HTML



Servlet



3.5.3 doGet() and doPost() Methods

☞ **Use :** Both of these methods are identical means both 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.

3.5.4 Difference between GET and POST

UQ. What is the difference between doGet() and doPost() in Servlet?

SPPU - May 2016, May 2017, 8 Marks

Sr. No.	Parameter	GET	POST
1.	Data	The data to be passed is appended to the URL of resource requested.	The data is passed to the server in the form of HTTP body.
2.	Hide	Does not hide the data to be passed.	Hides the data to be passed.
3.	Display	Displays data on the address bar of browser.	Does not display data on the address bar of browser.
4.	Security	Security is not maintained, hence not secure for private important data.	Security is maintained, hence secure for private important data.
5.	Data Limit	Limit on the amount of data to be transferred. (2KB with most servers)	No any limitation on the amount of data to be transferred.

3.5.5 Multi-Valued Parameter – Example/Program

Sometimes, it may be possible that the input parameter have multiple values. Here we have to use the getParameterValues() method to accept the input provided by the user.

Program 3.5.3 :Create HTML page which shows list of products. The user will select the desired products and submit the data to Servlet. The Servlet should display the total bill generated.

☑ **Soln. :**

Example/program of shopping cart using multi-valued parameter

HTML Script

```
<html>
<head>
<title>Shopping Cart</title></head>
<body>
<font color="blue" size=6>
<form method ="get" ACTION="http://localhost:8081/servlet/s4">

<input type="checkbox" name="product" value=30000>Computer
: rs 30000
<br>
<input type="checkbox" name="product"

value=50000>Laptop : rs 50000

<br>
<input type="checkbox" name="product" value=20000>Mobile :
rs 20000
<br>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form></font></body>
</html>
```

All checkboxes should have same name

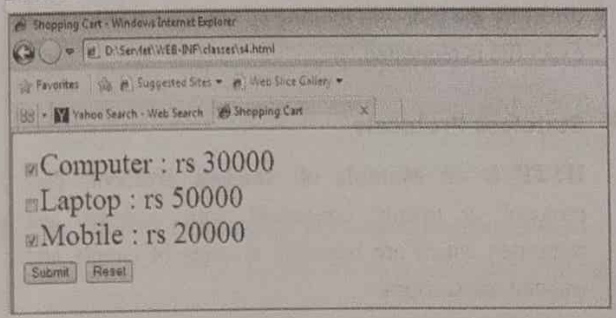
Servlet Code

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

public class s4 extends HttpServlet {

  protected void doGet(HttpServletRequest request,
HttpServletResponse 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>");
  double bill = 0;
  String[ ] amounts = request.getParameterValues("product");

  if(amounts != null)
  {
    for(int i=0;i<amounts.length;i++)
    {
      bill = bill + Double.parseDouble(amounts[i]);
    }
  }
  out.println("<font size=5>");
  out.println("The final bill Bill is : "+bill);
  out.println("</font>");
    out.println("</body>");
    out.println("</html>");

    out.close();
  }
}
```

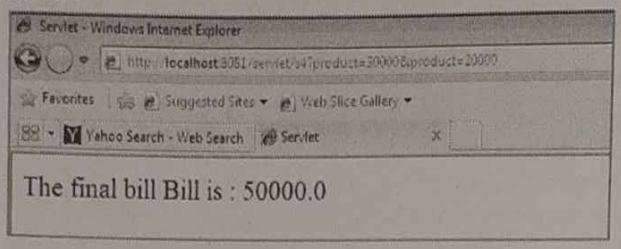
Get values of all selected checkboxes

Output

HTML



Servlet



3.6 BASIC CONCEPTS OF SESSIONS, COOKIES AND SESSION TRACKING

The communication protocols used in internet are generally divided into **two types : stateful and stateless**.

There is difference in the nature of connection which is defined between a client and a server.

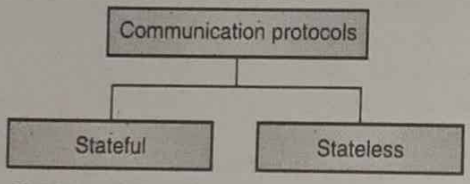


Fig. 3.6.1 : Types of Communication Protocols

Stateful Protocols

- **Telnet and FTP (File Transfer Protocol) are the** examples of stateful protocols. The client connects to the server. A series of transactions is performed by the client and then it disconnects.
- All these requests are associated by the server and it knows that they all came from the same user. That means the server associates a state with connection.
- Server knows everything about the user like who the user is, and what are his/her transactions or operations.
- Example : A file download request is sent by the FTP client, the FTP server can associate the request with previous request of the same user and work out which directory the user was looking at, and henceforth which exact file is requested.

Stateless Protocols

- **HTTP is an example of stateless protocol**. This protocol is mainly concerned with requests and responses which are basically a series of simple and isolated transactions.
- This is suitable for simple web browsing where each request is going to be converted into a file (an HTML document, GIF etc.) which is to be sent back to the client.
- There is no need for server to have knowledge of whether a series of requests come from the same, or from different client, or whether all these requests are related with each other or distinct.
- Stateless protocols are best suited for the web applications, but a big question is that; how to associate all the related requests together – how to maintain user state on the server.
- The process of maintaining state between series of requests to a Web Application is known as session tracking. This usually means passing data created from one request onward, so that it can be associated with the data created from subsequent requests.
- There are several approaches considered when the Web Application uses servlets to handle those requests.

3.6.1 Sessions

GQ. What is session?

Definition of 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 by default have knowledge about which session a specified request belongs to. It can be obtained as follows.
- 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.

3.6.2 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.

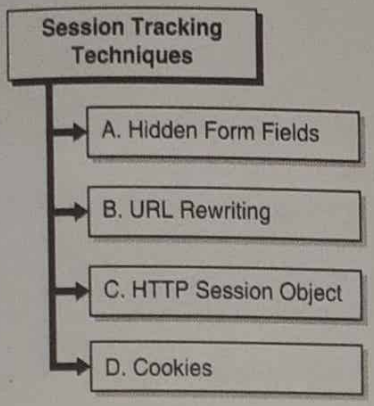


Fig. 3.6.2 : Session tracking techniques

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.

Example

```
<Input type="Hidden" name="pass' VALUE="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 cookie is disabled or not.

Disadvantages of Hidden Form Field

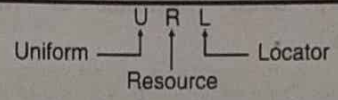
(i) The maintenance of Hidden Form Field is done at server side.

(ii) There is need of Extra form submission for each page.

(iii) Sending only text type of data is allowed.

B. URL Rewriting

GQ. Explain "URL Rewriting" with example. List its advantages and disadvantages.



- At the end of each URL (Uniform Resource Locator), we can add some extra text related to session. The session identifier is associated with session data by the server.

Example

- 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

(i) It will work perfectly every time without considering whether cookie is disabled or not.

(ii) 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.

Unit III End Sem.

☞ Commonly used methods of HttpSession interface

GQ. What are Commonly used methods of HttpSession interface ?

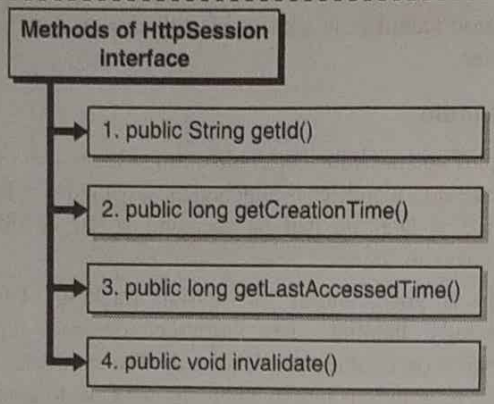


Fig. 3.6.3 : Methods of HttpSession interface

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 3.6.1 : Create HTML page which should accept the user name for user in one session scope in one Servlet and retrieve this value (name) from same session scope in second Servlet.

☑ **Soln. :**

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);
```

getSession() method returns the session object

```
    out.print("<br><br><a
href='http://localhost:8081/servlet/ServletSecond'>Click To
Proceed</a>");
    out.print("</font>");
```

Control is transferred to next Servlet

```
    out.close();

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

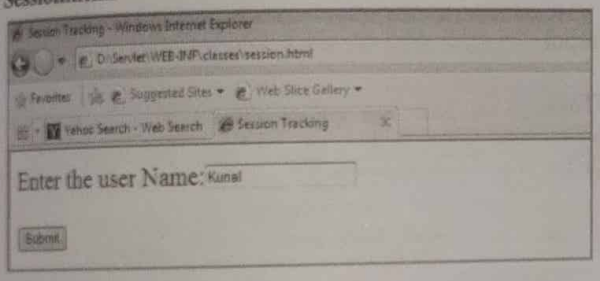
ServletSecond.java

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

public class ServletSecond extends HttpServlet
{

public void doGet(HttpServletRequest request, HttpServletResponse
response)
{
    try
```

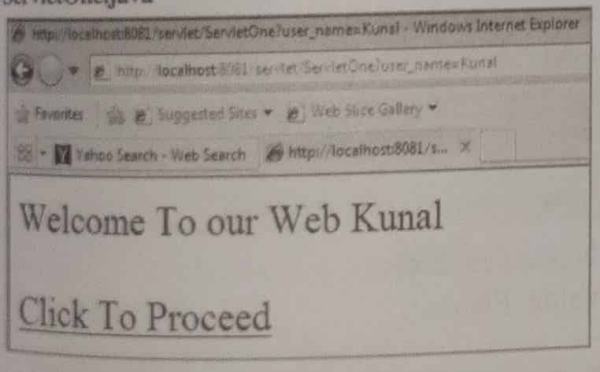
```
{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    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

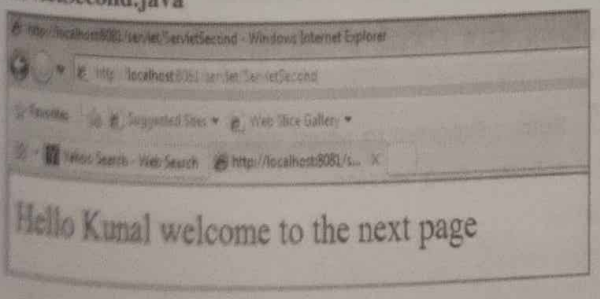
Session.html



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.

Unit III End Sem.

Program 3.6.2 : Write a program to set an expiry time/data for cookie.

☑ **Soln. : Setting cookie program**

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);
```

Cookie is created by passing name and value

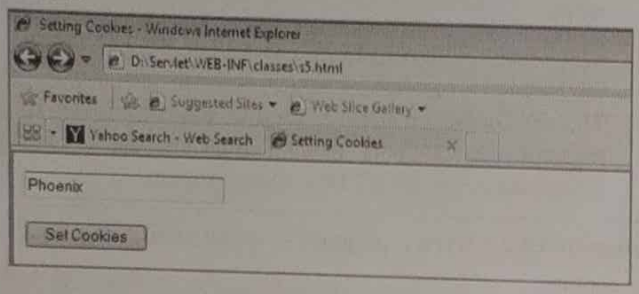
```
ck.setMaxAge(2*24*60*60);
```

Setting expiry time for cookie

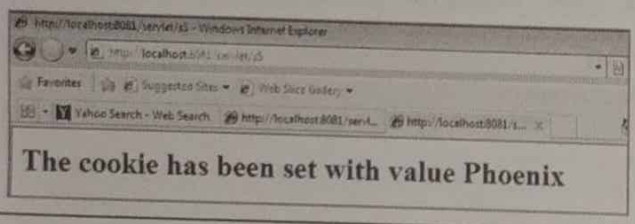
```
    response.addCookie(ck);
out.println("<b><font size=6>The cookie been set with value
"+data);
  }
}
```

Output

HTML



Servlet



Program 3.6.3 : Write a program of retrieving cookies.

☑ **Soln. :**

Retrieving Cookie

Use : getCookies() method is used to access the cookies that are added to response object.

HTML

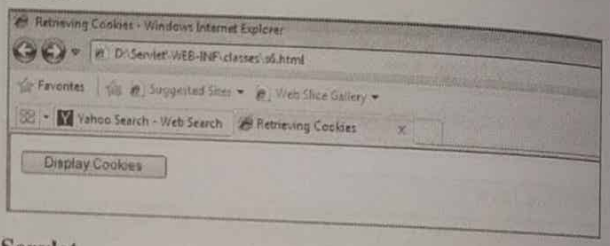
```
<html>
<head>
<title>Retrieving Cookies</title>
</head>
<body>
<form method ="post" action="http:\\localhost:8081\servlet\s6">
<input type="submit" value="Display Cookies">
</form>
</body>
</html>
```

Servlet

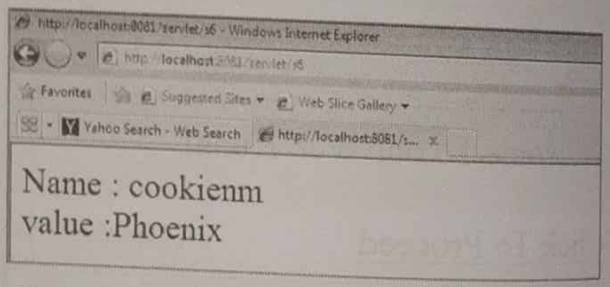
```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class s6 extends HttpServlet
{
public void doPost(HttpServletRequest request, HttpServletResponse
response)  throws ServletException, IOException
  {
    Cookie ck[] = request.getCookies();

    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



☞ **Example/Program on Session**

Program 3.6.4 : Write a program/example to count visit counter / hit counter of a website using session.

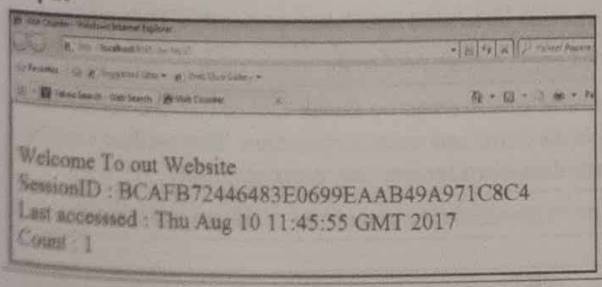
☑ **Soln. : Program to count visit counter / hit counter of a website using session**

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);
                                  Creating Session
    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



☞ Advantages and disadvantages of cookies

GQ. Write 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.

3.7 OTHER SERVLET CAPABILITIES

3.7.1 Additional HttpServletRequest Methods

Following table lists several methods on HttpServletRequest that return information that is part of the HTTP request received from the client.

Unit III End Sem.

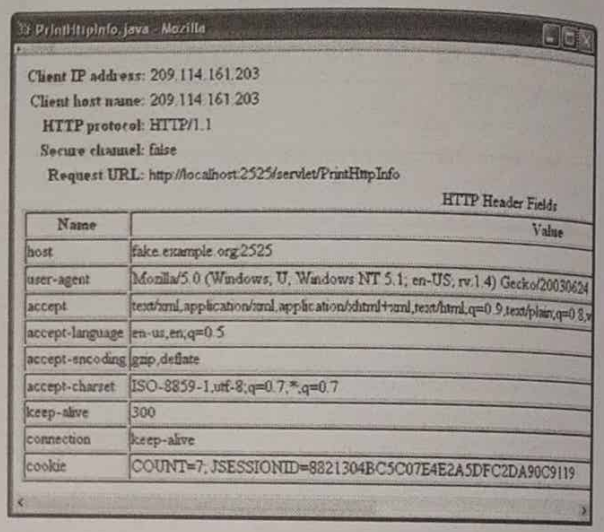
Method	Purpose
String getRemoteAddr0	Return IP address of the client machine making this request.
String getRemeteliost0	Return fully qualified name of the client making this request, or its IP address if name is not available.
String getprotocol()	Return the type and version of communication protocol used by the client to make this request (e.g, "HTTP/1.1")
boolean isSecure()	Return hoolean indicating whether or not this request was made over a secure communication channel.
StringBuffer getRequestURL ()	Return a StringBuffer containing the URL used to access this servlet, excluding any query string appended to the URL as well as any jsessi oni d path parameter.
Enumeration getHeaderNames1.	Return an Enumeration of String objects representing names of all header fields in the request.
StrIng getheader (String fieldName)	Given a valid header field name, return a String representing the value of the field, or nul1 if header field is not present in request. The match of fieldName against header field names is case insensitive.

The method printInfo() shown in the following program illustrates how these methods can be used to produce output.

☞ Method for displaying HTTP communication and header information

```
/**
* Add tables of HTTP protocol and header information to
* servletOut object
*/
private void printInfo(HttpServletRequest request,
PrintWriterservletOut)
throwsIOException
{
// Output protocol information
servletOut.println(
" <table border='0'> " +
" <tr><th align='right'>Client IP address:</th><td>" +
request.getRemoteAddr() + "</td></tr>" +
" <tr><th align='right'>Client host name:</th><td>" +
request.getRemoteHost() + "</td></tr>" +
" <tr><th align='right'>HTTP protocol:</th><td>" +
request.getProtocol() + "</td></tr>" +
" <tr><th align='right'>Secure channel:</th><td>" +
request.isSecure() + "</td></tr>" +
" <tr><th align='right'>Request URL:</th><td>" +
request.getRequestURL() + "</td></tr>" +
" </table>" );
// Output HTTP header fields
String headerName; // holds name portion of one header field
String headerValue; // holds value portion of one header field
Enumeration headerNames = request.getHeaderNames();
if (headerNames == null) {
servletOut.println(
" <p style='color:red'>Cannot access headers</p> " );
} else {
servletOut.println(
" <table border='1'> " +
" <caption>HTTP Header Fields</caption> " +
" <tr><th>Name</th><th>Value</th></tr>" );
while (headerNames.hasMoreElements()) {
headerName = (String)headerNames.nextElement();
headerValue = request.getHeader(headerName);
servletOut.println(
" <tr><td>" + headerName + "</td> " +
" <td>" + headerValue + "</td></tr>");
}
servletOut.println(
" </table>" );
}
}
```

Output



3.7.2 Additional HTTPServletResponse Methods

- Following Table lists several methods that can be called on the HttpServletResponse parameter (response object) in order to control various aspects of the HTTP response generated.

Method	Purpose
void setHeader (String name, String value)	Include a header field with the given name and value in the HTTP response.
void setDateHeader (St ring name, long value)	Include a date header field (such as Expires) with the given name in the HITTP response. The given value is converted from milliseconds since 00:00 0l January 1970 UTC to an equivalent time in HTTP date format.
void setContentLength(int 1 en)	Set the Content-Length header field to the given value.
void setRufferSize(int size)	Set the desired size of the response buffer (see text). The server may override the specified size and use a larger value. This method must be called before any data is written into the response buffer.
int getBufferSize()	Return an integer representing the actual size of the response buffer.

Method	Purpose
void setStdtus(int statusCode)	Set the status code in the HTTP response (status code is 200 (OK) by default). Any information contained in the response buffer is cleared. Use only for nonerror status codes.
void sendError(int statusCode, String msg)	Set the status code in the HTTP response to the given error statusCode (status code beginning with 4 or 5), and in the body of the response send a server-generated HTML error page containing the given msg.
void sendRedirect(String url)	Cause HTTP response with status code 307 (Temporal), Redirect) to be sent to the client, causing the client to send a new HTTP request to the given url. Client will behave as if it had sent request to the specified ur1.
void encodeRedi rectURL(String url)	Perform URL rewriting (for session management) on url that will be used for redirection.

- In order to effectively use many of these methods, it is important to understand how the server generates an HTTP response.
- All output sent to the PrintWriter object of the HttpServletResponse is stored in a data structure known as the response buffer. When this buffer becomes full, the server flushes the buffer, that is, sends its content to the client and clears the buffer in preparation for storing additional information.
- However, before this buffer can be flushed the first time, the HTTP headers of the response must be sent to the client, since the HTTP protocol requires that the header precede the body of the response. Therefore, all calls to methods that set HTTP header fields or the status code must be made before the first buffer flush, since later calls cannot cause additional header fields to be sent to the client. Methods that must be called before the first buffer flush include setHeader(), setDateHeader(), setContentLength(), setStatus(), sendError(), and sendRedirect().
- You might, then, be tempted to set the buffer size extremely large to ensure that the entire page generated by the servlet will fit in the buffer. This would have at least two advantages.
- First, the HTTP header fields and status code could be set at any time during generation of the page.
- Second, if the entire page fits within the buffer, then the server will automatically set the Content-Length header field before sending the HTTP response containing the page. This is useful because the TCP/IP connection over which the server and client communicate via HTTP can be kept open if content length information is contained in the response, and keeping the connection open means that subsequent requests can be made without taking the time to establish another connection.
- On the other hand, if the page content generated by your servlet will exceed the size of the buffer and you want the connection kept open, then your servlet should determine the total length of the content and set the Content-Length field explicitly by a call to setContentLength() before the first buffer flush.

3.7.3 Support for other HTTP Methods

- In addition to doGet() and doPost(), five other methods defined by HttpServlet corresponding to HTTP 1.1 request methods can be overridden.
- These methods all have the same signatures as doGet() and doPost(). They are doDelete(), doHead(), doOptions(), doPut(), and doTrace(). Both doOptions() and doTrace() have default implementations that typically should not be overridden. The doHead() method is also automatically defined if the doGet() method is overridden by the servlet code.

Unit III End Sem.

3.8 DATA STORAGE

- All web applications that is, software systems that include dynamic generation of HTML documents which interact with some form of data storage, usually either a file system or a database management system.
- For example, if you order a textbook from a Web site, the software implementing the site must have access to the current inventory of books being offered at the site and will also typically save your order in a data store of some type.
- Proper choice of a data storage mechanism and efficient implementation of software interacting with this mechanism is often crucial to developing a responsive and reliable web application.

3.9 SERVLETS AND CONCURRENCY

Concurrency in Web Servers

- Concurrent processing is a topic that is typically encountered in the study of operating systems, which must manage multiple user and system programs simultaneously. But concurrency is also encountered

within certain software applications, particularly web-based applications.

- For example, if you have noticed a web browser loading several images at the same time, or have observed that the web browser can respond to your mouse clicks while it is still downloading information from a web server, then you have seen concurrent processing in action on the client side.
- However, while such concurrency is present in the browser itself, a client-side software developer typically doesn't need to deal explicitly with issues related to concurrent processing in his or her code. This is because each JavaScript function runs to completion, without interruption by other JavaScript functions.

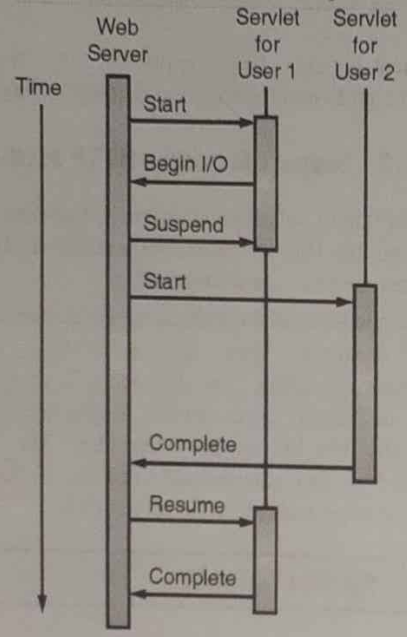


Fig. 3.9.1 : Schematic of server running servlets for two users concurrently

- Without concurrent processing, the second user's HTTP request could not begin to be processed until the first user's request was completed, and the CPU might sit virtually idle while waiting for the input/output operation of the first servlet to complete.
- Generally speaking, this idle time will lead to longer overall processing times if servlets are run sequentially each running to completion before the next can begin rather than concurrently.

3.10 DATABASES (MYSQL) AND JAVA SERVLETS

Servlet Database Connectivity

- Requirements to connect with the MySQL database in Servlet:
- MySQL should be installed in your computer.
- MySQL connector JAR (should be kept into lib folder).

Create Table

To create the Employees table in TEST database, use the following steps

Step 1

Open a **Command Prompt** and change to the installation directory as follows –

```
C:\>
C:\>cd ProgramFiles\MySQL\bin
C:\Program Files\MySQL\bin>
```

Step 2

Login to database as follows

```
C:\Program Files\MySQL\bin>mysql-u root -p
Enter password:********
mysql>
```

Step 3

Create the table **Employee** in **TEST** database as follows –

```
mysql>use TEST;
mysql> create table Employees(
idintnotnull,
ageintnotnull,
firstvarchar(255),
lastvarchar(255)
);
Query OK,0 rows affected (0.08 sec)
mysql>
```

Create Data Records

Finally you create few records in Employee table as follows –

```
mysql> INSERT INTO Employees VALUES (100,18,'Zara','Ali');
Query OK,1 row affected (0.05 sec)

mysql> INSERT INTO Employees VALUES
(101,25,'Mahnaz','Fatma');
Query OK,1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (102,30,'Zaid','Khan');
Query OK,1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (103,28,'Sumit','Mittal');
Query OK,1 row affected (0.00 sec)
mysql>
```

Accessing a Database

Here is an example which shows how to access TEST database using Servlet.

```
// Loading required libraries
import java.io.*;
importjava.util.*;
importjavax.servlet.*;
importjavax.servlet.http.*;
```

```
importjava.sql.*;

publicclassDatabaseAccessextendsHttpServlet{

publicvoiddoGet(HttpServletRequest request,HttpServletResponse
response)
throwsServletException,IOException{

// JDBC driver name and database URL
staticfinalString JDBC_DRIVER ="com.mysql.jdbc.Driver";
staticfinalString DB_URL="jdbc:mysql://localhost/TEST";

// Database credentials
staticfinalString USER ="root";
staticfinalString PASS ="password";

// Set response content type
response.setContentType("text/html");
PrintWriterout=response.getWriter();
String title ="Database Result";

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

out.println(docType+
"<html>\n"+
"<head><title>"+ title +"</title></head>\n"+
"<body bgcolor = \"#f0f0f0\">\n"+
"<h1 align = \"center\">"+ title +"</h1>\n");
try{
// Register JDBC driver
Class.forName("com.mysql.jdbc.Driver");

// Open a connection
Connection conn =DriverManager.getConnection(DB_URL, USER,
PASS);

// Execute SQL query
Statementstmt=conn.createStatement();
Stringsql;
sql="SELECT id, first, last, age FROM Employees";
ResultSetrs=stmt.executeQuery(sql);

// Extract data from result set
while(rs.next()){
//Retrieve by column name
int id =rs.getInt("id");
int age =rs.getInt("age");
String first =rs.getString("first");
Stringlast=rs.getString("last");
//Display values
out.println("ID: "+ id +"<br>");
out.println(", Age: "+ age +"<br>");
out.println(", First: "+ first +"<br>");
out.println(", Last: "+last+"<br>");
}
out.println("</body></html>");
// Clean-up environment
rs.close();
stmt.close();
conn.close();
}catch(SQLException se){
//Handle errors for JDBC
se.printStackTrace();
}catch(Exception e){
//Handle errors for Class.forName
e.printStackTrace();
}finally{
//finally block used to close resources
try{
if(stmt!=null)
stmt.close();
}catch(SQLException se2){
}// nothing we can do
try{
if(conn!=null)
conn.close();
}catch(SQLException se){
se.printStackTrace();
}//end finally try
}//end try
}
}
```

Unit III End Sem.

Now let us compile above servlet and create following entries in web.xml

```
....
<servlet>
<servlet-name>DatabaseAccess</servlet-name>
<servlet-class>DatabaseAccess</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>DatabaseAccess</servlet-name>
<url-pattern>/DatabaseAccess</url-pattern>
</servlet-mapping>
....
```

Now call this servlet using URL http://localhost:8080/DatabaseAccess which would display following response –

```
Database Result
ID : 100, Age : 18,  First : Zara,    Last : Ali
ID : 101, Age : 25,  First : Mahnaz,  Last : Fatma
ID : 102, Age : 30,  First : Zaid,    Last : Khan
ID : 103, Age : 28,  First : Sumit,   Last : Mittal
```

3.11 XML

3.11.1 Introduction to XML

UQ. What is XML? **SPPU - May 2017, 4 Marks**

- XML stands for eXtensible Markup Language.
- XML is basically designed to store and transport data.
- XML supports both human- and machine-readable data format.
- XML was designed to be self-descriptive.
- XML is a W3C Recommendation.
- XML is a simple text-based format which represents information in structured formats like documents, transactions, data, invoices, books etc. This language is derived from comparatively older standard format called SGML **(Standard Generalized Markup Language),** to make it more suitable for Web use.

3.11.2 Need of XML

UQ. Why we need XML? **SPPU - May 2017, 4 Marks**

There are several reasons for the need of XML as follows:

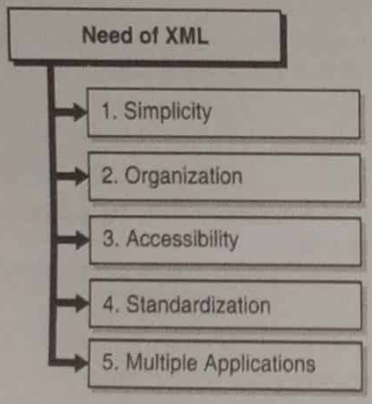


Fig. 3.11.1 : Need for XML

1. Simplicity

- XML can be easily understood. We can create our own tags and build the application.
- We are free to develop the system as per our requirements and with our own conventions. This makes the thing very simple for us.

2. Organization

- The design process can be segmented to build the platform. Data can be stored on one page while the formatting rules can be stored on another page.
- It is possible to create the data page to store the content first and later on we can work on design. XML allows us to create the website in stages and stay organized in the entire process.

3. Accessibility

- Data can be divided in XML.
- This makes the access of data easy and fast whenever there is need of making change in the data.

4. Standardization

- XML is an international standard.
- This means XML document can be viewed anywhere in the world.

5. Multiple Applications

- "Write once, use anywhere, any number of times" rule is applied to XML. For XML data we can create any number of display pages as we want.
- XML allows us to create various styles and formats for a single page as per requirement.

3.11.3 XML Key Components

GQ. Explain various Key Components of XML. **(8 Marks)**

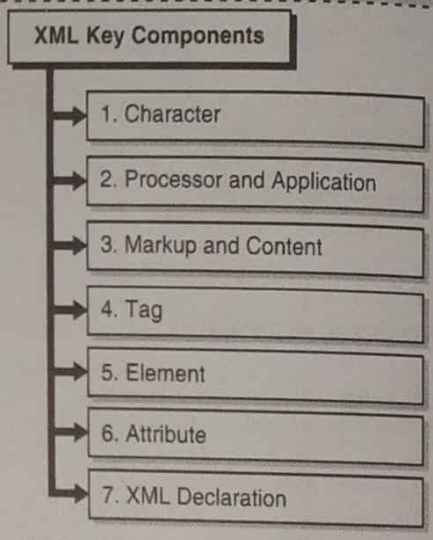


Fig. 3.11.2 : XML Key Components

1. Character

- An XML document is a string of characters.
- Almost every legal Unicode character may appear in an XML document.

2. Processor and Application

- The processor analyzes the markup and passes structured information to an application.
- The specification places requirements on what an XML processor must do and not do, but the application is outside its scope.
- The processor (as the specification calls it) is often referred to formally as an XML parser.

3. Markup and Content

- The characters making up an XML document are divided into markup and content, which may be distinguished by the application of simple syntactic rules.
- Generally, strings that constitute markup either begin with the character < and end with a >, or they begin with the character & and end with a ;. Strings of characters that are not markup are content.

4. Tag

A tag is a markup construct that begins with < and ends with >. Tags come in three flavors:

- Start-tag, such as <section>
- End-tag, such as </section>
- Empty-element tag, such as <line-break />

5. Element

- An element is a logical document component that either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag.
- The characters between the start-tag and end-tag, if any, are the element's content, and may contain markup including other elements which are called as child elements.

An example is

```
<greeting>Hello, world!</greeting>.
```

Another is

```
<line-break />.
```

6. Attribute

- An attribute is a markup construct consisting of a name–value pair that exists within a start-tag or empty-element tag.

Example

```
<img src="Rose.jpg" alt=" Rose " />,
```

where the names of the attributes are "src" and "alt", and their values are "Rose.jpg" and "Rose" respectively.

- Another example is <step number="3">Connect A to B.</step>, where the name of the attribute is "number" and its value is "3".
- An XML attribute can only have a single value and each attribute can appear at most once in each element.
- In the common situation where a list of multiple values is desired, this must be done by encoding the list into a well-formed XML attribute with some format beyond what XML defines itself.
- Usually this is either a comma or semi-colon delimited list or, if the individual values are known not to contain spaces, a space-delimited list can be used.
- Welcome! , where the attribute "class" has both the value "inner greeting-box" and also indicates the two CSS class names "inner" and "greeting-box".

7. XML Declaration

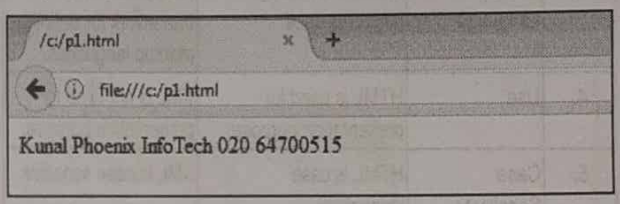
- XML documents may begin with an XML declaration that describes some information about themselves.
- An example is <?xml version="1.0" encoding="UTF-8"?>.

Program 3.11.1 : Write a code to display a data with the help of XML.

Soln. :

```
<?xml version="1.0" encoding="UTF-8"?>
<contact-info>
  <name>Kunal</name>
  <company>Phoenix InfoTech</company>
  <phone>020 64700515</phone>
</contact-info>
```

Output

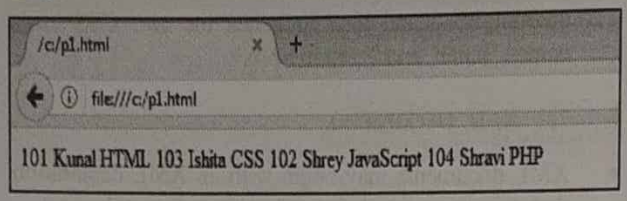


OR

```
<?xml version = "1.0"? encoding="UTF-8"?>
<student-data>
<student>
  <rno>101</rno>
  <name>Kunal</name>
  <subject>HTML</subject>
</student>
<student>
  <rno>103</rno>
  <name>Ishita</name>
  <subject>CSS</subject>
</student>
<student>
  <rno>102</rno>
  <name>Shrey</name>
  <subject>JavaScript</subject>
</student>
<student>
  <rno>104</rno>
  <name>Shravi</name>
  <subject>PHP</subject>
</student>
</student-data>
```

Unit III End Sem.

Output



3.11.4 Difference between HTML and XML

UQ. Differentiate between HTML and XML.
SPPU - May 2016, 8 Marks

Sr. No	Parameter	HTML	XML
1.	Long Form	HTML stands for HyperText Markup Language.	XML stands for eXtensible Markup Language.
2.	Purpose	HTML was designed to display data which concentrates on look of data.	XML was designed to store and transport the stored data.
3.	Markup Language	HTML is a markup language itself.	XML provides a framework for defining markup languages.
4.	Use	HTML is used for presentation purpose.	XML is not used for presentation purpose.
5.	Case Sensitivity	HTML is case insensitive.	XML is case sensitive.
6.	Tags	HTML uses only predefined tags.	XML allows user defined tags
7.	Restrictions	HTML is flexible related to syntax. E.g .no need to close the tags.	XML makes it compulsory for the user to close all the tags.
8.	White Space	HTML does not preserve white space.	XML preserves white space.
9.	Static or Dynamic	HTML is static.	XML is dynamic.

3.11.5 Transforming XML into XSLT

GQ. How to transform XML into XSLT? **(8 Marks)**

- **XSLT (eXtensible Stylesheet Language Transformations) is** the standard style sheet language for XML.
- XSLT is better than CSS. XSLT helps to add/remove elements and attributes to or from the output file. Also the elements can be rearranged and sort, execute tests and decide which elements to hide or display.
- XPath is used by XSLT to search information in an XML document.
- XSL stands for EXtensible Stylesheet Language. It considered as same to XML as CSS is to HTML.

Need for XSL

- In HTML all tags used are predefined such font, body, heading etc. and browser has knowledge of adding styles to them and display them to the end user with the help of CSS styles. This is not same in XML as it consists of user defined tags.
- Hence to understand and style an XML document, W3C (World Wide Web Consortium) has provided XSL which can work as XML based Stylesheet Language. An XSL document is used to specify the way by which a browser should render an XML document.

Following are the main parts of XSL -

- XSLT - Converts XML document into number of other types of documents.
- XPath - Helps to navigate through XML document.
- XSL-FO - Helps to format XML document.

What is XSLT ?

XSLT stands for Extensible Stylesheet Language Transformations. It used to transform XML data from one format into other.

How XSLT Works ?

- In an XSLT stylesheet we can define the transformation rules which is to be applied on the target XML document. The format of XSLT stylesheet is XML.
- The XSLT stylesheet is taken by the XSLT Processor which apply the transformation rules on the target XML document and further creates a formatted document in the format of XML, HTML, or text.
- XSLT formatter then use this formatted document to generate the actual output to display to the end user.

Advantages of XSLT

1. Programming is independent. An xsl file (which is a XML document) is used to write the transformations.
2. It is possible to update the output by just making changes in the transformations in xsl file. There is no need of changing any code. Hence it is easily possible for web designers to update the stylesheet and observe the change in the output quickly.

Program 3.11.2 : Write a code to transform the XML document in XSLT.

Soln. : Code to transform the XML document in XSLT

Syntax

Let's consider the following sample students.xml file, which is required to be transformed into a well-formatted HTML document.

students.xml

```
<?xml version = "1.0"?>
<class>
 <student rno = "101">
  <fname>Kunal</fname>
  <lname>B.</lname>
  <marks>85</marks>
 </student>
 <student rno = "102">
  <fname>Ishita</fname>
  <lname>B.</lname>
  <marks>90</marks>
 </student>
 <student rno = "103">
  <fname>Atharv</fname>
  <lname>P.</lname>
  <marks>75</marks>
 </student>
</class>
```

Now for above XML document we have to define an XSLT style sheet document to fulfill the following criteria

- The title of page should be Students.
- Page should contain table having student details.
- Columns should have give headers: Roll_No, First_Name, Last_Name, Marks

Step 1 : Create XSLT document

Now to fulfill above requirements, we will create an XSLT document.

students.xsl

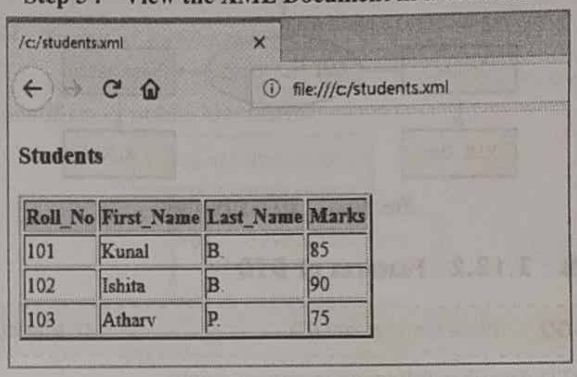
```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0"
xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

 <xsl:template match = "/">

  <html>
   <body>
    <h3>Students</h3>

    <table border = "2">
     <tr bgcolor = "lightgray">
      <th>Roll_No</th>
      <th>First_Name</th>
      <th>Last_Name</th>
      <th>Marks</th>
     </tr>
     <xsl:for-each select="class/student">
      <tr>
       <td>
       <xsl:value-of select = "@rollno"/></td>
       <td><xsl:value-of select = "fname"/></td>
       <td><xsl:value-of select = "lname"/></td>
       <td><xsl:value-of select = "marks"/></td>
      </tr>
     </xsl:for-each>

    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Step 2 : Link the XSLT Document to the XML Document

Make changes in student.xml document by adding xml-stylesheet tag. Set href value to students.xsl

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>
<class>
--------
</class>
```

Step 3 : View the XML Document in a browser

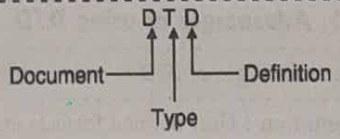


Unit III End Sem.

3.12 DTD

3.12.1 Introduction

UQ. Explain the term Document Type Definition (DTD). **SPPU - Dec. 2014, 4 Marks**



❒ **Definition :** XML Document Type Declaration which is also known as DTD is a way which helps to describe specifically the XML language. DTDs examine the validity of structure and vocabulary of an XML document against the grammatical rules and regulations of the suitable XML language.

- An application can take help of a DTD to check that XML data is valid.

- Generally an XML document is defined as:
  - **Well-formed :** An XML document is considered as well-formed if it follows all the XML rules like tags must be correctly nested, open-close tags are balanced, and empty tags ends with '/>'

  OR

  - **Valid :** Valid means, any XML document is not just only well-formed but also ensures availability of DTD that indicates which tags it uses, attributes of those tags, and which tags can be nested inside other tags, among other properties.
- In the Fig. 3.12.1 we can observe that a DTD is used to structure the XML document:

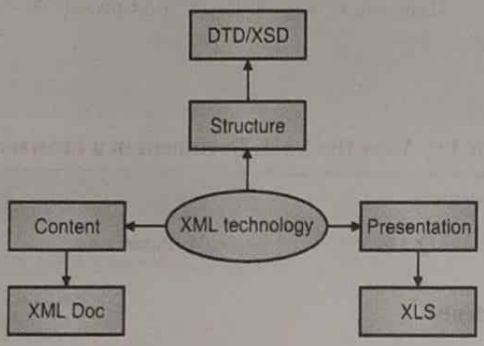


Fig. 3.12.1 : DTD structure

3.12.2 Features of DTD

GQ. Write features of DTD. **(2 Marks)**

DTD describes following important points:

1. The elements which can appear in an XML document.
2. The order of elements.
3. Optional and mandatory elements.
4. Attributes of elements and also whether they are optional or mandatory.
5. Whether or not attributes have default values.

3.12.3 Advantages of using DTD

GQ. State advantages of DTD. **(2 Marks)**

1. **Documentation :** User defined formats are allowed for XML files. These formats help user/developer to understand the structure of the data.
2. **Validation :** It gives a system to ensure the validity of XML files by verifying whether the elements are in proper sequence, mandatory elements and attributes are exactly at their locations and placed in correct way etc.

3.12.4 Schema

GQ. Write note on Schema of DTD. **(4 Marks)**

GQ. Write note on internal and external DTD. **(4 Marks)**

On its declaration basis, there are two types of DTD in the XML document:

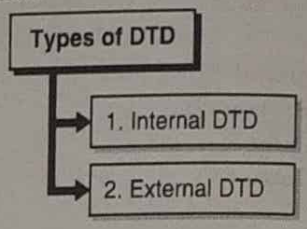


Fig. 3.12.2 : Types of DTD

1. **Internal DTD :** Declared inside the file itself
2. **External DTD :** Declared in a separate file

Syntax

```
<!DOCTYPE element DTD identifier
[
  declaration1
  declaration2
  .....
]>
```

In the above syntax

- <!DOCTYPE delimiter is beginning of DTD
- An **element** indicates the parser to parse the document from the given root element.
- **DTD identifier** is an identifier for the document type definition. This may be the path of a file on the system or URL of a file on the internet. When DTD points to external path, it is known as **external subset.**
- The **square brackets** [ ] indicates optional list of entity declarations which is called as **internal subset**.

1. Internal DTD

In this DTD elements are declared inside the XML files. In XML declaration the *standalone* attribute is set to yes to reference it as internal DTD.

Syntax

```
<!DOCTYPE root-element [element-declarations]>
root-element - name of root element
element-declarations - is where the elements are declared.
```

Example (Declaration)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE details [
  <!ELEMENT details (emp_name,company,emp_phone)>
  <!ELEMENT emp_name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT emp_phone (#PCDATA)>
]>
<Details>
  < emp_name>Kunal</ emp_name>
  <company>Phoenix InfoTech</company>
  < emp_phone>020 64700515</ emp_phone>
</ Details >
```

Let us discuss the above code:

Start Declaration

XML declaration is started with following statement

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes" ?>
```

DTD

Immediately after the XML header, the *DTD* follows, usually referred as the DOCTYPE:

```
<!DOCTYPE details [
```

There is an exclamation mark (!) at the beginning of the element name in DOCTYPE declaration. The DOCTYPE instructs the parser that a DTD is linked with this XML document.

DTD Body

After DOCTYPE declaration, there is body of the DTD. Here we can declare elements, attributes, entities, and notations:

```
<!ELEMENT details
(emp_name,company,emp_phone)>
<!ELEMENT emp_name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT emp_phone (#PCDATA)>
```

Several elements are declared here. <!ELEMENT name (#PCDATA)> implies the element *name* to be of type "#PCDATA". Here the meaning of #PCDATA is parse-able text data.

End Declaration

At the end, DTD's declaration section is closed with a closing bracket and a closing angle bracket (]>). This efficiently ends the definition, and then, the XML document follows immediately.

Rules

- The document type declaration should be at the beginning of the document. It can only be preceded by the XML header.
- The element declarations should begin with an exclamation mark.
- The Name in the DTD and element type of the root element must be same.

2. External DTD

- In this DTD the elements are declared out of the XML file. They are retrieved by mentioning the system attributes which may be either the legal *.dtd* file or any valid URL.
- When *standalone* attribute in the XML declaration is set to **no**, it indicates that declaration get information from the source outside the file.

Syntax

```
<!DOCTYPE root-element SYSTEM "file-name">
```

Here the *file-name* is file with *.dtd* extension.

Example

UQ. Write suitable example of DTD. **SPPU - Dec. 2014, 4 Marks**

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE Details" Details.dtd">
< Details>
 <emp_name>Kunal</emp_name>
 <company>Phoenix InfoTech</company>
 <emp_phone>020 64700515</emp_phone>
</Details>
```

The content of the DTD file **address.dtd** are as shown:

```
<!ELEMENT details (emp_name,company,emp_phone)>
<!ELEMENT emp_name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT emp_phone (#PCDATA)>
```

Unit III End Sem.

Types of External DTD

The external DTD can be referred by using **system identifiers** or **public identifiers**.

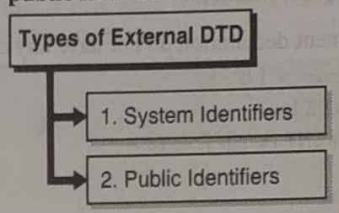


Fig. 3.12.3 : Types of External DTD

1. System Identifiers

This helps to specify the location of an external file which contains DTD declarations.

```
<!DOCTYPE name SYSTEM "Details.dtd" [...]>
```

2. Public Identifiers

A mechanism is provided by public identifier to locate DTD resources.

```
<!DOCTYPE name PUBLIC "-// XML Docs//DTD
Exm.//EN">
```

It starts with keyword PUBLIC and then followed by a specialized identifier.

3.12.5 Elements

GQ. Explain the XML Elements with their types. **(8 Marks)**

Definition : XML elements are the building blocks of an XML document. Elements are considered as a container to store elements, attributes, text, media objects or mixture of all.

☞ **Use :** An ELEMENT declaration is used to declare DTD element. When DTD validates the XML file, parser at the beginning checks the root element and then the child elements.

☞ **Syntax**

```
<!ELEMENT elementname (content)>
```

☞ **Element Content Types**

Elements contents in the declaration of DTD are categorized as:

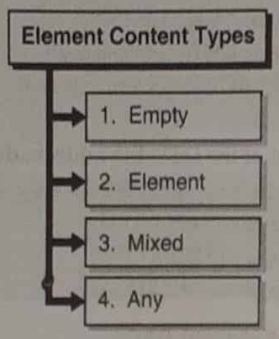


Fig. 3.12.4 : Element Content Types

1. Empty content

The element declaration do not have any content.

```
<?xml version="1.0"?>
<!DOCTYPE hr[
  <!ELEMENT Details EMPTY>
]>
<Details />
```

2. Element content

Here the content is included in the elements within parentheses.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE Details [
  <!ELEMENT Details (emp_name,company,emp_phone)>
  <!ELEMENT emp_name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT emp_phone (#PCDATA)>
]>
<Details>
  < emp_name>Kunal</ emp_name>
  <company>Phoenix InfoTech</company>
  < emp_phone>020 64700515</emp_phone>
</Details >
```

3. Mixed Element Content

In this we use the combination of (#PCDATA) and children elements. PCDATA means parsed character data, that means, text which is not markup. Here the text can appear by itself or it can be placed between the elements.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE Details [
  <!ELEMENT Details (#PCDATA|name)*>
  <!ELEMENT emp_name (#PCDATA)>
]>
<Details>
The text is combined with child element
  <emp_name>Kunal</emp_name>
</Details>
```

4. Any Element Content

An element can be declared using the ANY keyword in our content. It is usually referred as mixed category element. When the type of content is not fixed, ANY is useful.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE Details [
  <!ELEMENT Details ANY>
]>
< Details >
Some Text can be written here
</ Details >
```

3.12.6 Attributes

GQ. Explain the attributes of XML Elements. **(4 Marks)**

- Attribute provides some more data about an element or more precisely it defines a property of an element. In XML the attribute is in the form of a name-value pair.
- There may be multiple unique attributes to an element.

☞ **Syntax**

Basic syntax of DTD attributes declaration is as follows:

```
<!ATTLIST element-name attribute-name attribute-type
attribute-value>
```

In the above syntax

- The DTD attributes start with <!ATTLIST keyword if the element contains the attribute.
- **element-name** specifies the name of the element to which the attribute applies.
- **attribute-name** specifies the name of the attribute which is included with the element-name.
- **attribute-type** defines the type of attributes.
- **attribute-value** takes a fixed value that the attributes must define.

☞ **Example**

```
<?xml version = "1.0"?>
```

☞ **<!DOCTYPE Syntax**

The syntax of DTD attributes declaration:

```
<!ATTLIST element-name attribute-name attribute-type
attribute-value>
```

In this syntax

- If there is attribute in element, then the DTD attributes start with <!ATTLIST keyword.

- **element-name** - name of the element to which the attribute applies.
- **attribute-name** - name of the attribute which is included in the element-name.
- **attribute-type** - type of attributes.
- **attribute-value** - fixed value

Example

```
<?xml version = "1.0"?>
<!DOCTYPE Details [
<!ELEMENT Details (emp_ name )>
<!ELEMENT emp_name ( #PCDATA )>
<!ATTLIST emp_name id CDATA #REQUIRED>
]>
<Details>
  <emp_name id="123">Kunal</emp_name>
</ Details >
```

3.12.7 Rules of Attribute Declaration

GQ. Explain the rules of Attribute Declaration. **(2 Marks)**

- It is necessary to declare all the attributes in Document Type Definition (DTD) using an Attribute-List Declaration which we want to use in XML document
- Attributes may appear in either start or empty tags.
- ATTLIST keyword should be in upper case
- Duplication of attribute names is not allowed in the attribute list for a given element.

3.12.8 Attribute Types

GQ. List the attribute types. **(2 Marks)**

When attributes are declared, we can specify how the processor will handle the data of attribute value. The attribute types are categorized in three different categories:

1. String type
2. Tokenized types
3. Enumerated types

3.13 AJAX

3.13.1 Introduction

GQ. What is AJAX? **(3 Marks)**

- **AJAX stands for Asynchronous JavaScript and XML.**

Use

- AJAX is an advanced technique used to create web applications which are better, interactive, and faster with the help of different scripting languages like HTML, CSS, Java Script and XML.
- Ajax uses XHTML (Extensible Hypertext Markup Language) for content, CSS (Cascading Style Sheets) for presentation, and Document Object Model plus JavaScript for creation of dynamic contents.
- AJAX allows the web pages to be modified asynchronously by exchanging small bunch of information with the server. This means that it is easily possible to make changes in part of a web page, without reloading the entire page.
- The conventional web pages, (which are not using AJAX) reload the whole page if there is need to make changes in the content.

Examples of Applications using AJAX

- Google Maps, Gmail, Youtube, and Facebook tabs.
- We will take simple example. You may have seen the live score of cricket match on different websites like www.yahoo.com. There are updations of each and every ball. In such case only small part of webpage is needed to be changed. Only that much information is sent by the server and updated in the webpage rather than updating the entire page. This is implemented by the AJAX.
- Just consider if it is done with conventional web pages without AJAX, then it will take lot much of duration every time to reload the whole page.

Unit III End Sem.

3.13.2 Working of AJAX

GQ. Explain Working of AJAX. **(3 Marks)**

- AJAX uses the XMLHttpRequest object to communicate with the server. The Fig 3.13.1 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.

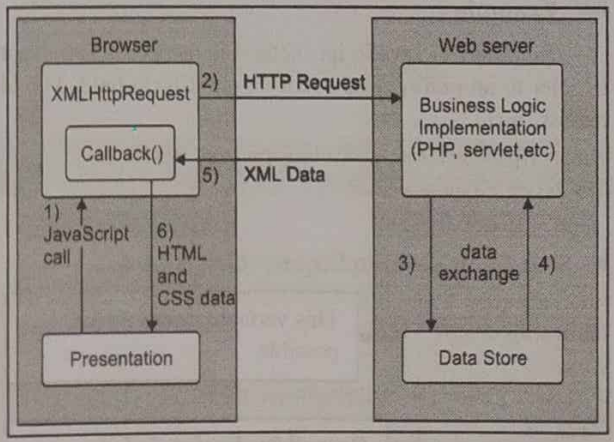


Fig. 3.13.1

- 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.
- 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.

3.13.3 AJAX Processing Steps and Ajax Script

GQ. Write steps to process AJAX **(8 Marks)**

Steps to process AJAX

- **Step 1 :** An event by the client is fired.
- **Step 2 :** An XMLHttpRequest object is created.
- **Step 3 :** The XMLHttpRequest object is configured.
- **Step 4 :** An asynchronous request is made by the XMLHttpRequest to the Webserver.
- **Step 5 :** The Webserver returns the result in XML format.
- **Step 6 :** The callback() function is called by the XMLHttpRequest object and processes the result.
- **Step 7 :** The HTML DOM is updated.

Let us look at these steps one by one.

- **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("Msxml2.XMLHTTP");
    }
catch (e)
{
    try{
      ajaxReqObj = new
ActiveXObject("Microsoft.XMLHTTP");
    }catch (e){
      // Exception occurs
      alert("Your browser broke!");
      return false;
    }
  }
 }
}
```

Internet Explorer Browser

- **Step 3 :** The XMLHttpRequest object is configured.

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;

 if (!target) target = document.getElementById("myid");
 var url = "validate?id=" + escape(target.value);
 ajaxReqObj.open("GET", url, true);
 ajaxReqObj.send(null);
}
```

processReqFun() is callback function

- **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;

 if (!target) target = document.getElementById("myid");
 var url = "validate?id=" + escape(target.value);

 ajaxReqObj.open("GET", url, true);
 ajaxReqObj.send(null);
}
```

processReqFun() is callback function

Internet Explorer

Now 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.

```
function processReqFun()
{
 if (req.readyState == 4)
{
   if (req.status == 200)
{
    var msg = ...;
...
}
```

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:

- Using DOM API, 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.

```
<script type="text/javascript">
<!--
function setMsgDOM(msg) {
  var userMessageElement =
document.getElementById("userIdMsg");
  var msgText;

  if (msg == "false")
{
    userMessageElement.style.color = "red";
    msgText = "Wrong User Id";
  }
  else
  {
    userMessageElement.style.color = "green";
    msgText = "Valid User Id";
  }
    var msgBody = 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,
userMessageElement.childNodes[0]);
  }
  else
  {
    userMessageElement.appendChild(msgBody);
  }
}
-->
</script>
<body bgcolor="gray">
<div id="userIdMsg"><div>
</body>
```


...Chapter Ends

❑❑❑

UNIT IV

CHAPTER 4

JSP and Web Services

Syllabus

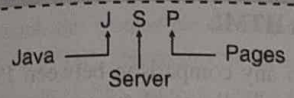
JSP: Introduction to Java Server Pages, JSP and Servlets, running JSP applications, Basic JSP, JavaBeans classes and JSP, Support for the Model-View-Controller paradigm, JSP related technologies. Web Services: Web Service concepts, Writing a Java Web Service, Writing a Java web service client, Describing Web Services: WSDL, Communicating Object data: SOAP. Struts: Overview, architecture, configuration, actions, interceptors, result types, validations, localization, exception handling, annotations.

Exemplar/Case Studies : Transform the blogging application from a loose collection of various resources (servlets, HTML documents, etc.) to an integrated web application that follows the MVC paradigm.

4.1 JSP

4.1.1 Introduction to JSP

UQ. What is JSP? SPPU - Dec. 2015, 2 Marks



- **JSP stands for Java Server Pages**. JSP is a server side Java technology used to create dynamic contents just like Servlet. JSP is an extension to Servlet and provides more functionality as compared to Servlet.
- **Advantage :** A JSP page is combination of HTML tags and JSP tags. The JSP pages are comparatively easy to maintain than those of Servlet because it is possible to separate designing and development in JSP. JSP provides some important extra features like Expression Language, Custom Tags etc.
- JSP can perform various operations like accepting input from user through the forms of web pages, present records from any data source on the server, and create dynamic web pages.
- The JSP tags provide various functionalities like accessing data from the database, transfer control between different web pages, using components of JavaBeans, registration of preferences regarding users, transferring various type of information within different requests as well as pages etc.

4.1.2 Advantages of JSP over Servlet

GQ. Write advantages JSP over Servlet. (4 Marks)

JSP has number of advantages over Servlet as follows :

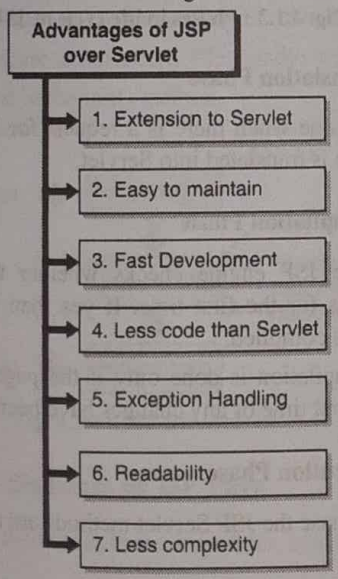


Fig. 4.1.1 : Advantages of JSP over Servlet

1. Extension to Servlet

JSP technology is considered as the extension to previous Servlet technology. JSP provides all the features of Servlet by adding some new features like implicit objects, custom tags, predefined tags, expression language. All these new features ease the development of dynamic web pages.

2. Easy to maintain

- In JSP the business logic and presentation logics are implemented independently and hence separated. It helps to manage the JSP easily.
- In Servlet both business and presentation logics are mixed which make management of Servlet complicated.

3. Fast Development

- In case of JSP, only the first time JSP is compiled. There is no need of recompilation and deployment of project for each call.
- While in case of Servlet, there is need to update and recompile the page when any changes are made in look and feel of the application.

4. Less code than Servlet

- JSP provide number of built in elements like implicit objects, action tags etc. which helps to reduce the code.
- Such elements are not available in Servlet.

Unit IV End Sem.

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.

4.1.3 Advantages of JSP over Other CGI

UQ. Enlist advantages of JSP over CGI? SPPU - Dec. 2015, 8 Marks

- JSP page can use dynamic Elements in HTML documents rather than having separate files. Hence performance is better than CGI.
- Before being processed by the Servlet, JSP is compiled. While the CGI requires the server to load an interpreter and the target script every time whenever the page is requested.
- JSP has the access of very powerful Enterprise Java API like EJB, JAXP JDBC, JNDI etc.
- The servlets can handle the business logic and JSP pages can work in combination with servlets.
- JSP is part of Java; hence it has access to entire JDK which helps to create simple to robust applications.

4.1.4 Advantages of JSP over Other Technologies

GQ. Write advantages of JSP over different technologies. **(4 Marks)**

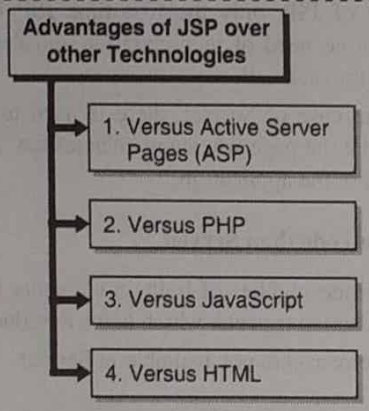


Fig. 4.1.2 : Advantages of JSP over different technologies

1. Versus Active Server Pages (ASP)

- ASP is a product of Microsoft. Just like JSP it is a tag base scripting language used to create dynamic web pages. ASP code is not portable while as JSP is a part of Java, it is portable means the JSP script can be executed on any operating system.
- Also JSP can work with various web servers while ASP is stick with web servers like IIS (Internet Information Server) or PWS (Personal Web Server).

2. Versus PHP

PHP is an HTML embedded scripting language used to create dynamic pages and it is also open source such as Java. PHP is also a good scripting language but JSP has an inbuilt support of entire class library of Java. Java is more powerful, flexible, reliable and portable as compared to PHP.

3. Versus JavaScript

JavaScript is basically a client side scripting language. It can create client side applications. It is not possible to create server side applications in JavaScript like JSP. JavaScript also cannot access database of the server.

4. Versus HTML

There is no any comparison between JSP and HTML. HTML is basically used to create static web pages. Also HTML cannot access server side resources like database. JSP can have both static and non-static content. JSP contains HTML as static part.

4.1.5 Lifecycle of JSP

UQ. What do you mean by JSP processing? How JSP pages are handled? Explain with architecture **SPPU - Dec. 2015, 8 Marks**

There are four phases in the life cycle of JSP which are used to **process** the JSP page. These are shown in Fig. 4.1.3.

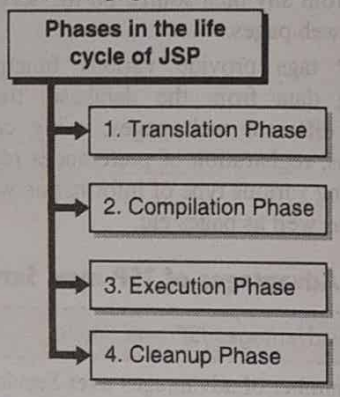


Fig. 4.1.3 : Phases in life cycle of JSP

1. Translation Phase

At first time when there is a request for JSP page, the JSP page is translated into Servlet.

2. Compilation Phase

- Here the JSP engine checks whether the JSP page request is for the first time. If yes then the generated Servlet is compiled.
- The compilation is done only if the page is requested for the first time or any changes have been made in it.

3. Execution Phase

In this phase the JSP Servlet methods are executed.

(a) **public void jspInit()**

- This method is called when the JSP is initialized before serving any requests. This method is same as of the init() method of Servlet. If there is any need of JSP-specific initialization process then the jspInit() method can be overridden.
- In general, this method can be called only once in the life cycle of JSP. This method is used to load some non-java resources like database connections, file handles etc.

(b) **public void jspService(HTTPServletRequest request, HTTPServletResponse response) throws ServletException, IOException**

- This method is basically corresponds to the body of the JSP page. This is the main method which handles the request response paradigm. This method is implemented by the JSP Container.
- The request object holds the information about the service request and response object stores the data which we want to send back to client.
- This method is called repeatedly for each client request.

4. Cleanup Phase

This phase indicates the end of JSP life cycle. In this phase the JSP is destroyed and removed from the container.

(c) **jspDestroy()**

This method get called when the container is about to destroy the JSP. This method is similar to Servlet's destroy() method. In this method we can perform the cleanup operation by releasing the non Java resources like database connection, file handle which we have initialized in the init() method.

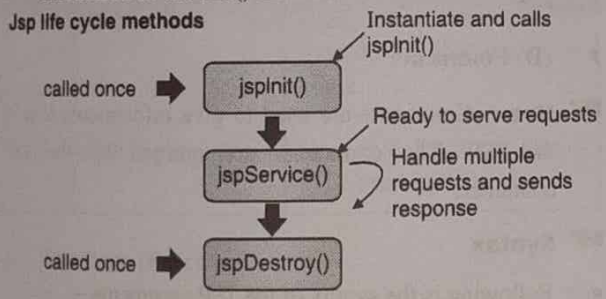


Fig. 4.1.4 : JSP life cycle method

4.1.6 Elements of JSP Page

In JSP there are three important types of elements available.

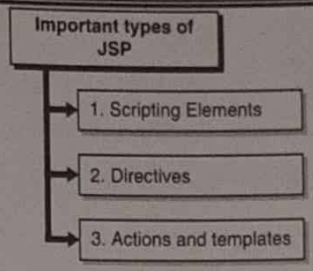


Fig. 4.1.5 : Types of JSP

I. 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. **(4 Marks)**

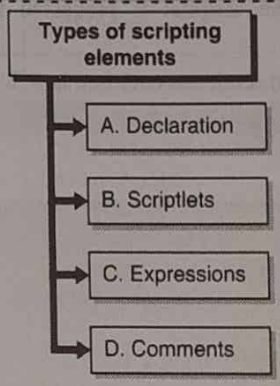


Fig. 4.1.6 : Types of Scripting Elements

(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>
```

Unit IV End Sem.

```
<%!
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

Use : 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>
```

II. Directives

UQ. Explain various JSP directives. SPPU - Dec. 2015, 4 Marks

- 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.

Syntax

```
<%@ page attribute = "value" %>
```

Different attributes of Page Directive

GQ. Explain different attributes of page directive in JSP. (4 Marks)

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" >
```

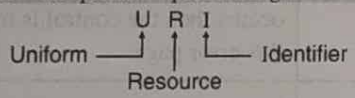
- In the include directive we have to specify the relative URL of the external file which we want to include in our JSP. If the filename is given without any specific path, then the JSP compiler search for the file in the same directory in which the current JSP is stored.

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.

Programs

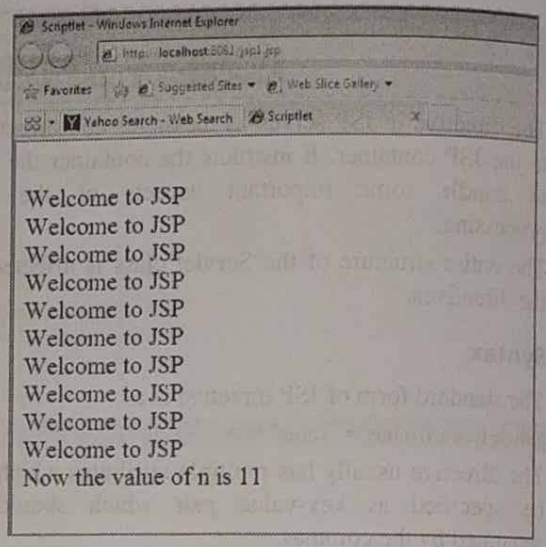
Program 4.1.1 : Write a simple JSP program.

Soln. :

Simple JSP program

```
<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>
```

Output



Program 4.1.2 : Create HTML page which should accept name from user. Send it to JSP. The JSP should display welcome message with the given name.

Soln. :

HTML

```
<html><head>
<title>Servlet</title>
</head>

<body>
<font color="blue" size=5>
<form method =
"post" ACTION="http://localhost:8081/jsp2.jsp">
Enter your name <input type="text" name="n"><br><br>
<input type="submit" value="Submit">
</form>
</font>
</body>
</html>
```

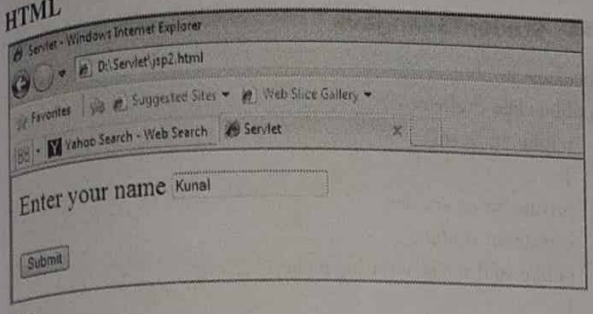
JSP

```
<%@page contentType="text/html"%>
<html>
<head><title>JSP Page</title></head>
<body>
<font color="blue" size=6>
```

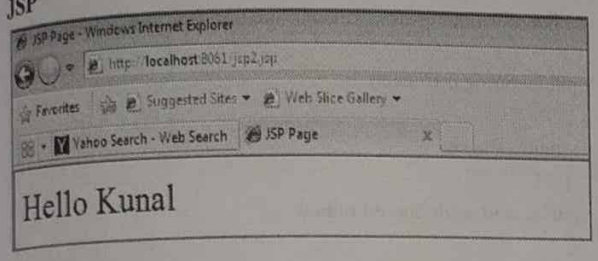
```
<%
String nm = request.getParameter("n");
out.println("Hello "+nm);
%>
</font>
</body>
</html>
```

Output

HTML



JSP



III. Actions and Templates

GQ. Enlist and explain different types of actions in JSP. **(4 Marks)**

Types of Actions

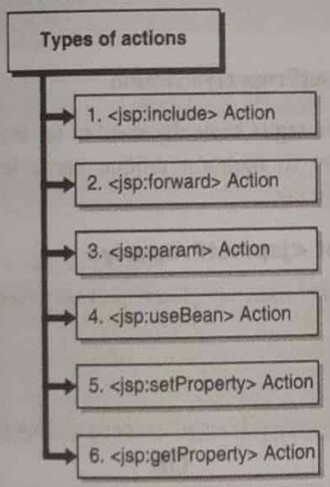


Fig. 4.1.7 : Types of Actions

Use : 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, reuse JavaBeans components or creation of HTML content for the Java plugin.

Syntax of actions

```
<jsp:action_name attribute = "value" />
```

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>
```

Unit IV End Sem.

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 also some other tags which are not frequently used.

7. <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.

<jsp:body> Action

<jsp:element> Action

<jsp:text> Action

<jsp:attribute> Action

4.2 IMPLICIT OBJECTS IN JSP

GQ. Explain implicit objects in JSP. **(8 Marks)**

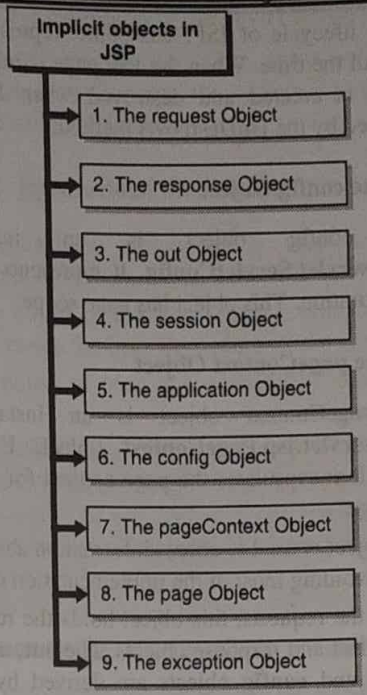


Fig. 4.2.1 : Implicit objects in JSP

1. The request Object

- This object is an instance of a **javax.servlet.http.HttpServletRequest** object. Whenever there is request for JSP page from client, a new object is created by JSP engine to represent that request.
- The HTTP header information containing form data, cookies, HTTP methods can be accessed using request object. This object is passed to the JSP by the container as a parameter.

2. The response Object

- The response object is an instance of a **javax.servlet.http.HttpServletResponse** object. When the JSP engine creates request object, at the same time it creates the response objet to handle the response for client.
- The response object encapsulates the response generated by the JSP, to be sent back to client in response to request. This object is passed to the JSP by the container as a parameter.

Unit IV End Sem.

3. The out Object

The out implicit object is an instance of a **javax.servlet.jsp.JspWriter** object. It represents output stream and is used to send content in response.

4. The session Object :

The session object is an instance of **javax.servlet.http.HttpSession**. The session object track the client session between different client requests. It is valid only for HTTP requests. Sessions are created automatically hence this variable exists even if there was no incoming session reference.

5. The application Object

- It is an instance of a **javax.servlet.ServletContext** object. It represents the context within which the JSP is executing.

- In the lifecycle of JSP, this object represents the JSP page all the time. When the JSP page is initialized, this object is created and destroyed when JSP page is removed by the **jspDestroy**() method.

6. The config Object

The config object is an instance of **javax.servlet.ServletConfig**. It represents the Servlet configuration. This object has page scope.

7. The pageContext Object

- The pageContext object is an instance of a **javax.servlet.jsp.PageContext** object. It has page scope. It encapsulates the page context for the specific JSP page.
- This object is used to access information about the page while avoiding most of the implementation details.
- For all the requests, this object holds the references to the request and response objects. The **out, application, session** and **config** objects are derived by retrieving attributes of this object.

8. The page Object

This object is considered as real reference to the instance of the JSP page. It is an object which represents the complete JSP page.

9. The exception Object

- This object is an instance of class java.lang.Throwable. It refers to the runtime exception which is resulted in the error page being invoked. This object has page scope.
- The exception object is a container which holds the exception which is thrown by the preceding page. It generates suitable response to the error situation.

4.3 JAVABEANS CLASSES AND JSP

4.3.1 What is JavaBeans ?

- JavaBeans is a portable, platform-independent model written in Java Programming Language. Its components are referred to as beans.
- In simple terms, JavaBeans are classes which encapsulate several objects into a single object. It helps in accessing these object from multiple places. JavaBeans contains several elements like Constructors, Getter/Setter Methods and much more.
- JavaBeans has several conventions that should be followed :
  (1) Beans should have a default constructor (no arguments)
  (2) Beans should provide getter and setter methods
  (3) A getter method is used to read the value of a readable property
  (4) To update the value, a setter method should be called
  (5) Beans should implement java.io.serializable, as it allows to save, store and restore the state of a JavaBean you are working on.

What are JavaBean Properties ?

- A JavaBean property can be accessed by the user of the object.
- The feature can be of any Java data type, containing the classes that you define.
- It may be of the following mode: read, write, read-only, or write-only. JavaBean features are accessed through two methods:

1. getEmployeeName ()

For example, if the employee name is firstName, the method name would be getFirstName() to read that employee name. This method is known as an accessor. Properties of getter methods are as follows :

- Must be public in nature
- Return-type should not be void
- The getter method should be prefixed with the word get
- It should not take any argument

2. setEmployeeName ()

For example, if the employee name is firstName, the method name would be setFirstName() to write that employee name. This method is known as a mutator. Properties of setter methods :

- Must be public in nature
- Return-type should be void
- The setter method has to be prefixed with the word set
- It should take some argument

Program : Implementation of JavaBeans

```
public class Employee implements java.io.Serializable
{
privateint id;
private String name;
public Employee()
  {
  }
public void setId(int id)
```

```
  {
    this.id = id;
  }
publicintgetId()
  {
return id;
  }
public void setName(String name)
  {
    this.name = name;
  }
public String getName()
  {
return name;
  }
}
```

Next program is written in order to access the JavaBean class that we created above:

```
public class Employee1 {
public static void main(String args[])
{
  Employee s = new Employee();
s.setName("TechNeo");
System.out.println(s.getName());
}
}
```

Output

```
TechNeo
```

☞ Advantages of JavaBeans

The following list enumerates some of the benefits of JavaBeans :

(1) Portable

- JavaBeans components are built purely in Java, hence are fully portable to any platform that supports the Java Run-Time Environment.
- All platform specifics, as well as support for JavaBeans, are implemented by the Java Virtual Machine.

(2) Compact and Easy

- JavaBeans components are simple to create and easy to use. This is an important focus sector of the JavaBeans architecture.
- It doesn't take much effort to write a simple Bean. Also, a bean is lightweight, so, it doesn't have to carry around a lot of inherited baggage to support the Beans environment.

(3) Carries the Strengths of the Java Platform

- JavaBeans is pretty compatible, there isn't any new complicated mechanism for registering components with the run-time system.

4.3.2 jsp:useBean Action Tag

- The jsp:useBean action tag is used to locate or instantiate a bean class.
- If bean object of the Bean class is already created, it doesn't create the bean depending on the scope.
- But if object of bean is not created, it instantiates the bean.

```
Syntax of jsp:useBean action tag
<jsp:useBean id= "instanceName" scope= "page | request |
session | application"  class= "packageName.className" type=
"packageName.className" beanName="packageName.className |
<%= expression >" >
</jsp:useBean>
```

Attributes and Usage of jsp:useBean action tag

(1) **id :** is used to identify the bean in the specified scope.

(2) **scope :** represents the scope of the bean. It may be page, request, session or application. The default scope is page.

- **Page :** specifies that you can use this bean within the JSP page. The default scope is page.
- **Request :** specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.
- **Session :** specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.
- **Application :** specifies that you can use this bean from any JSP page in the same application. It has wider scope than session.

(3) **class :** Instantiates the specified bean class (i.e. creates an object of the bean class) but it must have no-arg or no constructor and must not be abstract.

(4) **type :** Provides the bean a data type if the bean already exists in the scope. It is mainly used with class or beanName attribute. If you use it without class or beanName, no bean is instantiated.

(5) **beanName :** instantiates the bean using the java.beans.Beans.instantiate() method.

☞ Example of jsp:useBean action tag

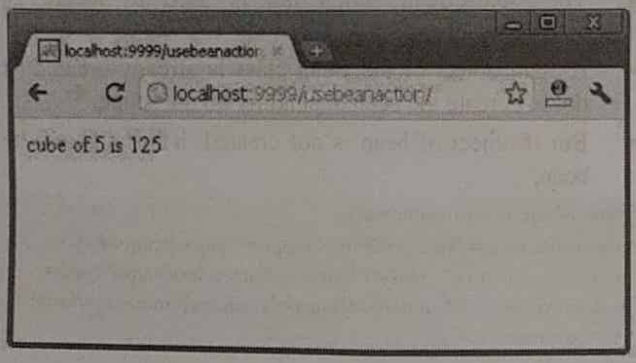
```
Calculator.java (a simple Bean class)
packagecom.javatpoint;
public class Calculator{
publicint cube(int n){return n*n*n;}
 }
```

Unit IV End Sem.

index.jsp file

```
<jsp:useBean id="obj" class="com.javatpoint.Calculator"/>
<%
int m=obj.cube(5);
out.print("cube of 5 is "+m);
%>
```

Output



4.3.3 jsp:setProperty and jsp:getProperty Action Tags

- The setProperty and getProperty action tags are used for developing web application with Java Bean.
- In web devlopment, bean class is mostly used because it is a reusable software component that represents data.
- The jsp:setProperty action tag sets a property value or values in a bean using the setter method.

Syntax of jsp:setProperty action tag

```
<jsp:setProperty name="instanceOfBean" property= "*" |
property="propertyName" param="parameterName" |
property="propertyName" value="{ string | <%= expression
%>}"
/>
```

- Example of jsp:setProperty action tag if you have to set all the values of incoming request in the bean

```
<jsp:setProperty name="bean" property="*" />
```

- Example of jsp:setProperty action tag if you have to set value of the incoming specific property

```
<jsp:setProperty name="bean" property="username" />
```

- Example of jsp:setProperty action tag if you have to set a specific value in the property

```
<jsp:setProperty name="bean" property="username" value="Kuma
r" />
```

jsp:getProperty action tag

The jsp:getProperty action tag returns the value of the property.

Syntax of jsp:getProperty action tag

```
<jsp:getProperty name="instanceOfBean" property="propertyName
" />
```

Simple example of jsp:getProperty action tag

```
<jsp:getProperty name="obj" property="name" />
```

Example of bean development in JSP

In this example there are 3 pages:

- index.html for input of values
- welocme.jsp file that sets the incoming values to the bean object and prints the one value
- User.java bean class that have setter and getter methods

index.html

```
<form action="process.jsp" method="post">
Name:<input type="text" name="name"><br>
Password:<input type="password" name="password"><br>
Email:<input type="text" name="email"><br>
<input type="submit" value="register">
</form>
```

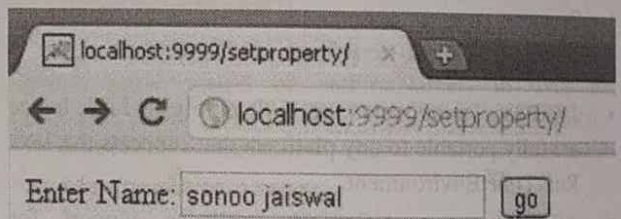
process.jsp

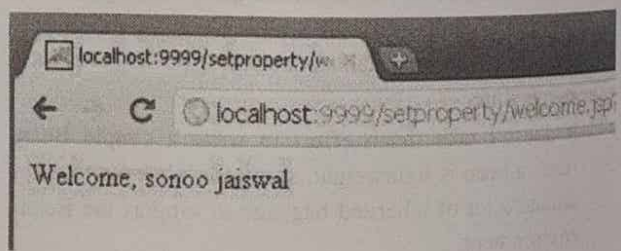
```
<jsp:useBean id="u" class="org.sssit.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>
Record:<br>
<jsp:getProperty property="name" name="u"/><br>
<jsp:getProperty property="password" name="u"/><br>
<jsp:getProperty property="email" name="u" /><br>
```

User.java

```
packageorg.sssit;

public class User {
private String name,password,email;
//setters and getters
}
```

Output





4.4 SUPPORT FOR THE MODEL-VIEW-CONTROLLER PARADIGM

Many web applications are developed within a conceptual framework known as the model-view-controller (MVC) paradigm, and several JSP facilities are often used when developing within this framework.

4.4.1 MVC in JSP

MVC is an architecture that separates business logic, presentation and data. In MVC,

- M stands for Model
- V stands for View
- C stands for controller.

MVC is a systematic way to use the application where the flow starts from the view layer, where the request is raised and processed in controller layer and sent to model layer to insert data and get back the success or failure message. The MVC Architecture diagram is represented below :

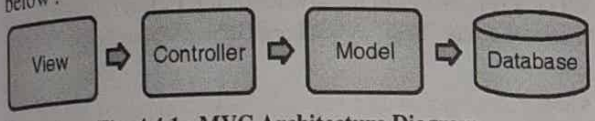


Fig. 4.4.1 : MVC Architecture Diagram

1. **Model Layer**

- This is the data layer which consists of the business logic of the system.
- It consists of all the data of the application
- It also represents the state of the application.
- It consists of classes which have the connection to the database.
- The controller connects with model and fetches the data and sends to the view layer.
- The model connects with the database as well and stores the data into a database which is connected to it.

2. **View Layer**

- This is a presentation layer.
- It consists of HTML, JSP, etc. into it.
- It normally presents the UI of the application.
- It is used to display the data which is fetched from the controller which in turn fetching data from model layer classes.
- This view layer shows the data on UI of the application.

3. **Controller Layer**

- It acts as an interface between View and Model.
- It intercepts all the requests which are coming from the view layer.
- It receives the requests from the view layer and processes the requests and does the necessary validation for the request.
- This requests is further sent to model layer for data processing, and once the request is processed, it sends back to the controller with required information and displayed accordingly by the view.

4.4.2 Advantages of MVC Architecture

The advantages of MVC are :

1. Easy to maintain
2. Easy to extend
3. Easy to test
4. Navigation control is centralized

Example of JSP Application Design with MVC Architecture

In this example, we are going to show how to use MVC architecture in JSP.

- We are taking the example of a form with two variables "email" and "password" which is our view layer.
- Once the user enters email, and password and clicks on submit then the action is passed in mvc_servlet where email and password are passed.
- This mvc_servlet is controller layer. Here in mvc_servlet the request is sent to the bean object which act as model layer.
- The email and password values are set into the bean and stored for further purpose.
- From the bean, the value is fetched and shown in the view layer.

Mvc_example.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>MVC Guru Example</title>
</head>
<body>
<form action="Mvc_servlet" method="POST">
Email: <input type="text" name="email">
<br />
Password: <input type="text" name="password" />
```

Unit IV End Sem.

```
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

Mvc_servlet.java

```
packagedemotest;
importjava.io.IOException;

importjavax.servlet.RequestDispatcher;
importjavax.servlet.ServletException;
importjavax.servlet.http.HttpServlet;
importjavax.servlet.http.HttpServletRequest;
importjavax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Mvc_servlet
 */
public class Mvc_servlet extends HttpServlet {
      private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
publicMvc_servlet() {
super();
      // TODO Auto-generated constructor stub
   }

      protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
            // TODO Auto-generated method stub
            String email=request.getParameter("email");
      String password=request.getParameter("password");

TestBeantestobj = new TestBean();
testobj.setEmail(email);
testobj.setPassword(password);
request.setAttribute("gurubean",testobj);
RequestDispatcherrd=request.getRequestDispatcher("mvc_success.j
sp");
rd.forward(request, response);
      }

}
```

TestBean.java

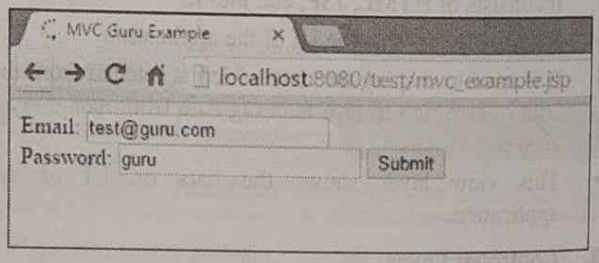
```
packagedemotest;

importjava.io.Serializable;

public class TestBean implements Serializable{
      public String getEmail() {
            return email;
      }
      public void setEmail(String email) {
            this.email = email;
      }
      public String getPassword() {
            return password;
      }
      public void setPassword(String password) {
            this.password = password;
      }
      private String email="null";
      private String password="null";

}
```

Mvc_success.jsp

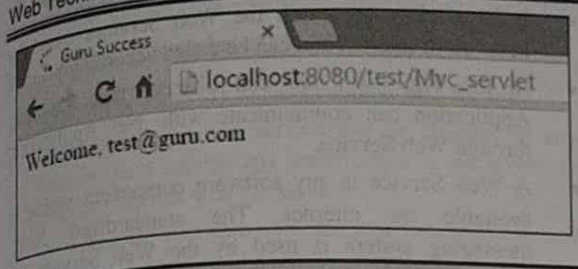
```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="demotest.TestBean"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Guru Success</title>
</head>
<body>
<%
TestBeantestguru=(TestBean)request.getAttribute("gurubean");
out.print("Welcome, "+testguru.getEmail());
%>
</body>
</html>
```

Output

When you enter email and password in screen and click on submit then, the details are saved in TestBean and from the TestBean they are fetched on next screen to get the success message.



After clicking on submit the output is shown as below



4.5 JSP RELATED TECHNOLOGIES

4.5.1 JSP Pages with Scriptlets

- Many existing JSP based web applications use an older version of JSP that is not as compatible with XML; text written in this older version of JSP is referred to as a JSP page rather than a JSP document.
- A JSP page uses various special symbols as part of the string beginning a tag in order to indicate that the tag is not template data. For example, an include directive within a JSP page might look like

```
<%@ include file="aForm.html" %>
```

where the string <%@ denotes the beginning of a directive.

- Early versions of JSP also did not incorporate the EL language, and as a result earlyJSP pages tended to make extensive use of embedded Java scriptlet code (enclosed in<% and %> tag delimiters) for accessing and processing data. This approach produces JSPpages with a mixture of markup and Java, such as

```
<p>Hello
<%
out.println(request.getRemoteHost());
%>
</p>
```

- As we have seen, newer versions of JSP make it possible to implement MVC view components without the need for embedding scriptlets.

4.5.2 Active Server Pages and ASP.NET

- Microsoft Active Server Pages (ASP) technology is supported by Microsoft's Internet Information Services (IIS) web server as well as by several other web servers.
- The syntax shown in the previous subsection for JSP pages is similar to that used in ASP (which is not surprising, since JSP was released after ASP had been in use for some time).
- Embedded code in ASP pages is written in either JScript (the Microsoft implementation of ECMAScript) or VBScript (a scripting language based on Microsoft's popular Visual Basic development language).
- There is no analog to Java servlets in ASP, but if an MVC approach is used with ASP then the controller portion of the web application can be written as an ASP page consisting entirely of script.
- The model portion of an ASP application is typically written using some variant of Microsoft's COM technology rather than JavaBeans classes.
- ASP.NET goes beyond ASP in a number of ways. First, in addition to JScript and VBScript, a wide variety of other languages - including the C# language, which is similar in many respects to Java - can be used for embedding code in an ASP.NET page.
- Also, ASP.NET pages are compiled to an intermediate form before being served, much as JSP pages are translated to servlets and compiled to Java byte code.
- Thus, an ASP.NET web application typically executes in much less time than a comparable ASP application, in which each page is interpreted each time it is accessed.
- In addition, the .NET framework provides a large number of COM objects designed to simplify the development of web applications by automating various common tasks, such as generating HTML and JScript code representing some form controls that are not directly supported by HTML forms.

4.5.3 PHP: Hypertext Preprocessor

- PHP (an acronym with the infinitely recursive definition "PHP: Hypertext Preprocessor") is a Perl-like scripting language that can be embedded in HTML documents much as Java scriptlets can be embedded in JSP pages or scripting code in ASP. One nice difference is that the syntax

```
<?php ... ?>
```

can be used to embed PHP code, which means that an XML parser will interpret the tags as XML processing instructions with target php. Thus, PHP pages can be written so that they are fully compatible with XML tools.

- Code implementing the PHP scripting language can be run on a variety of operating system platforms (Linux, Windows, Macintosh, etc.) as well as with a variety of web servers, including both Apache and IIS.

4.5.4 ColdFusion

- ColdFusion, although one of the earliest of the technologies for embedding program logic in HTML documents, is a relatively clean approach to the task.

Unit IV End Sem.

- All program logic is embedded as XML elements (with names beginning with the characters cf), so a ColdFusion document is XML compatible.
- A ColdFusion document may also contain expressions (e.g., function calls or variable references) enclosed in # characters, which are evaluated when a ColdFusion document is requested and replaced with the values obtained. This means that the author of a ColdFusion page must add the character # to the list of XML characters that must be treated in a special way.
- Like JSP tag libraries, ColdFusion allows developers to create custom elements that can be used within ColdFusion documents. In fact, some versions of ColdFusion can use JSP tag libraries directly. In addition, MVC model software can be implemented in several ways, including as either COM or JavaBeans objects.
- Much like PHP, ColdFusion software can be run on multiple operating system platforms and with multiple web servers, including Apache and IIS. In fact, much like Tomcat, ColdFusion software also comes bundled with its own web server, so ColdFusion does not require a separate front-end server as current versions of PHP do.

4.6 WEB SERVICES

4.6.1 Overview and Features of Web Services

(A) Overview

GQ. What is Web Service? **(2 Marks)**

- Now days there are various programming platforms available to develop web-based applications. Some applications are developed in Java, some in .Net while some are in Node.js, AngularJS etc.
- Sometimes theses heterogeneous applications required to communicate with each other. But as these applications are developed in different platforms, it becomes difficult to set communication between them.
- In such situation, the **Web Services** comes in picture. Web Services offers a common platform which helps the various applications built on different platforms to communicate with each other.
- A Web Service is a software component which is developed to perform specific tasks regarding communication between two platforms.

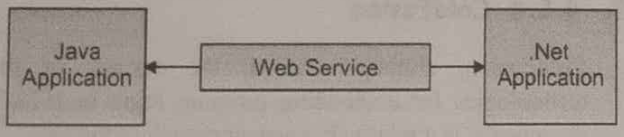


Fig. 4.6.1 : Web Service

- It is possible to search the Web Services over the network or internet and can be called accordingly.
- In the Fig. 4.6.1 we can observe that the Java Application can communicate with .Net Application through Web Service.
- A Web Service is any software component which is available on internet. The standardized XML messaging system is used by the Web Service for communication purpose. All communications are encoded through XML.

☞ **Example :** An XML message is sent by the client to invoke the Web Service and also waits for XML response.

- The Web Service is not dependent on any operating system since the communication is done in XML. Application in any platform can interact with any other application of any platform.
- Web Services are modular, dynamic, distributed and self contained software components which can be easily published, located and also called over the network or internet.
- Web Services are built on the standard languages like HTML, Java, XML and with standard protocols like TCP/IP and HTTP.

(B) Features of Web Services

UQ. What are general features of web services?
SPPU - Aug. 2015, In sem, 4 Marks

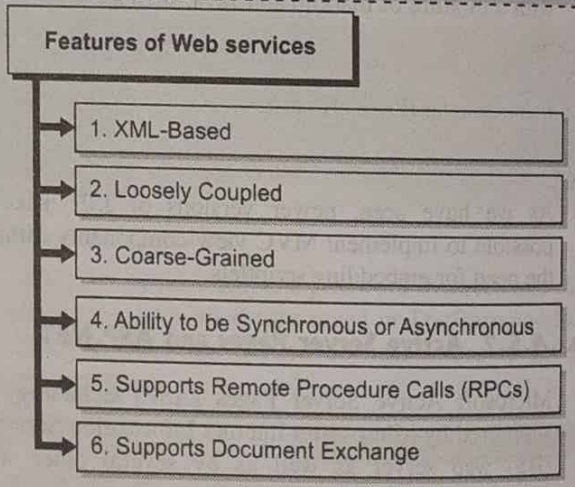


Fig. 4.6.2 : Features of web services

1. XML-Based

- Web services uses XML for transfer of data in the internet.
- XML is not dependent on any specific operating system, networking environment or platform.

2. Loosely Coupled

- The web services are loosely coupled with the user.
- The interface of web service can be changed at any time without making any compromise in the ability of client to interact with the service.
- It makes the software systems more manageable and different systems can be easily integrated.

3. Coarse-Grained

Web services provide a mechanism to define the coarse-grained services which can access the exact part of business logic.

4. Ability to be Synchronous or Asynchronous

- In synchronous method, the client is bind with the service execution. Client has to wait for completion of service execution before continuing.
- In Asynchronous method, client can invoke the service as well as execute other functions.
- Web services provide both of the options.

5. Supports Remote Procedure Calls (RPCs)

In Web Services, the user can call the procedures and functions on remote object with the help of XML-based protocol.

6. Supports Document Exchange

Web service provides functionality to exchange documents which facilitate business integration.

4.6.2 Web Service Components

GQ. Explain Web Service Components. **(4 Marks)**

There are three major Web Service components.

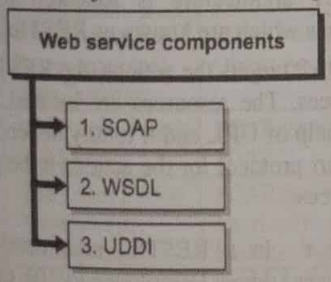


Fig. 4.6.3 : Web Service Components

1. SOAP

- SOAP stands for **Simple Object Access Protocol**.
- SOAP is a XML-based protocol which is used to access the Web Services.
- SOAP is a W3C recommendation for communication between various types of applications.
- As SOAP is XML based, it is platform and language independent. That means it can be used with any language on any platform.
- SOAP is simple and extensible.

2. WSDL

- WSDL was developed jointly by Microsoft and IBM.
- WSDL stands for **Web Services Description Language**.
- WSDL is the standard format for describing a Web Service.
- WSDL is an XML based document which contains the information regarding Web Services like names of methods, method parameters and how to access the methods. WSDL also contains information regarding which operation the Web Service will perform.
- WSDL is used for information exchange in decentralized and distributed environments.
- WSDL is a part of UDDI, an XML-based worldwide business registry.
- It works as an interface between various Web Service applications.
- WSDL is pronounced as wiz-dull.

3. UDDI

- UDDI stands for **Universal Description, Discovery and Integration**.
- UDDI is an XML-based standard which is used to describe, publish, and search Web Services.
- UDDI is a specification which is used in the distributed registry of Web Services.
- The SOAP, CORBA, and Java RMI Protocols are used by the UDDI for communication purpose.
- UDDI takes help of WSDL for describing interfaces to Web Services.
- UDDI is platform independent open framework
- UDDI is an open industry initiative which helps the various businesses to discover each other and decide the way through which they can interact with each other.

Unit IV End Sem.

4.6.3 Types of Web Services

GQ. Explain types of Web Service with advantages and disadvantages. **(4 Marks)**

There are mainly two types of Web Services.

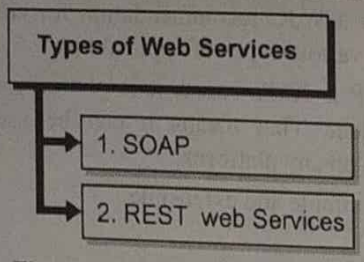


Fig. 4.6.4 : Types of Web Services

1. SOAP Web Services

- We have seen that SOAP is an XML-based protocol. The main benefit of using the SOAP Web Service is that it has its own security system.
- An envelope is provided by the SOAP to send messages related to Web Service over the internet using the HTTP protocol. The format of messages is usually XML.

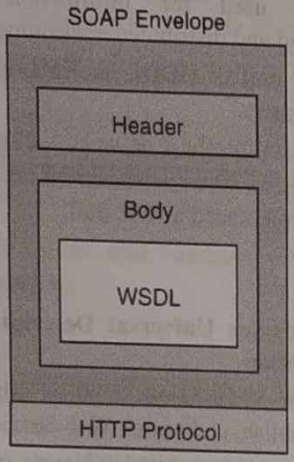


Fig. 4.6.5 : SOAP Format

- In simple words SOAP is a mechanism which is used to send a request in XML format over the Internet using HTTP protocol, and in return accessing response in XML format.
- Considering a real world example, if a client wants to get information about an employee by sending the employee id in the request, he can perform this operation using Web Services.
- The applications which serves the SOAP requests, has a WSDL file. WSDL describes all the methods with parameters contained in the Web Service, with their types of request and response. It elaborates the contract which is implemented between service and client.

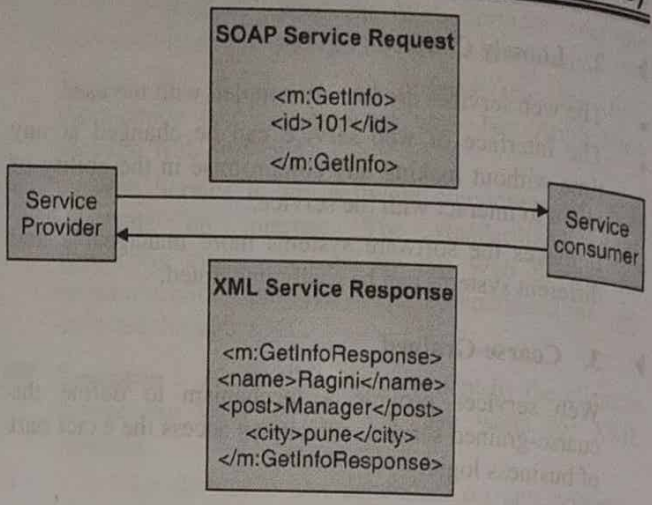


Fig. 4.6.6 : SOAP processing

Advantages of Soap Web Services

i. **WS Security :** SOAP has WS Security which is its own strong security mechanism.

ii. **Language and Platform independent :** The scripts of SOAP Web Services can be coded in any language which supports it and also can be executed on any web platform.

Disadvantages of Soap Web Services

i. **Slow :** The XML format which is used by the SOAP should be parsed before read. SOAP defines number of standards which should be followed in development of the SOAP applications. Hence SOAP is slow and requires more bandwidth and resources.

ii. **WSDL dependent :** SOAP has only one mechanism to discover the Web Service and that is WSDL.

2. REST Web Services

- The acronym of REST is Representational State Transfer. REST is a style of architecture. It is not any set of standard rules.
- The REST architecture is followed by number of applications which are known as RESTful.
- While SOAP targets the actions, the REST concentrates on resources. The resources are located by the REST with the help of URL and it totally depends on the type of transport protocol for the actions to be performed on the resources.

Example : In a RESTful architecture, this URL http://{serverAddress}/employee/id/101 can be used to retrieve employee information, we can send REST call of GET type, and the service will sends back the information of employee with id 101.

- The same service can also be used to make changes in the employee data, by sending the new updated values as form data in a PUT request.

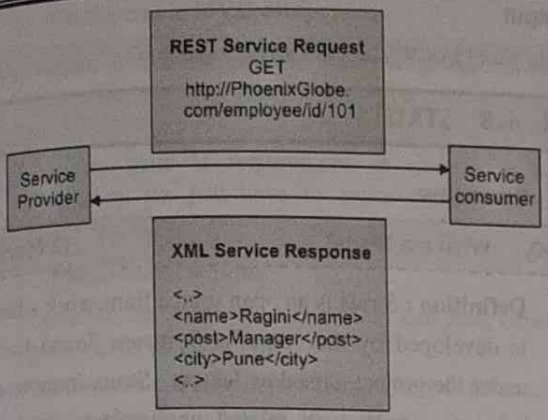


Fig. 4.6.7

☞ Advantages of RESTful Web Services

1. **Fast :** As RESTful Web Services do not have any strict specification like SOAP, they are comparatively fast. Also the bandwidth and resources consumed by the RESTful Web Services are very less.
2. **Language and Platform independent :** It is possible to write the RESTful Web Services in any programming language and also they can be executed on any platform.
3. **Can use SOAP :** SOAP can be used by the RESTful Web Services as the implementation.
4. **Permits different data format :** In RESTful Web Service, data of different formats like HTML, XML, JSON and even plain text can be used.

4.6.4 Difference between REST and SOAP

GQ. Write difference between REST and SOAP. **(4 Marks)**

Here are some of the basic differences between the two types of Web Services:

Parameter	REST	SOAP
Definition	REST is a style of software architecture.	SOAP is a protocol or a set of standards.
Full Form	REST stands for Representational State Transfer	SOAP stands for Simple Object Access Protocol
REST and SOAP	SOAP can be used by REST	REST cannot be used by SOAP
Business Logic	REST uses URI to implement the business logic.	SOAP uses the service interface to implement the business logic.
Standards	There are no strict standards in REST	There are strict standards in SOAP
Security	REST can use security of underlying transport protocols.	SOAP has its own security layer.
Data Formats	REST can accepts various types of data formats like HTML, JSON, XML or even Plain Text.	SOAP can work with only XML format.

4.7 JAVA WEB SERVICES

- Java web service application performs communication with the help of Web Services Description Language (WSDL).
- Java provides two options to write web service application code : SOAP and RESTful.

☞ Java Web Services API

There are two important APIs provided by Java for the purpose of developing web service applications since JavaEE 6.

1) **JAX-WS** : For SOAP web services. There are 2 options to write JAX-WS application code: through RPC style and Document style.
2) **JAX-RS** : For RESTful web services. There are mainly two implementations currently in use for creating JAX-RS application : Jersey and RESTeasy.

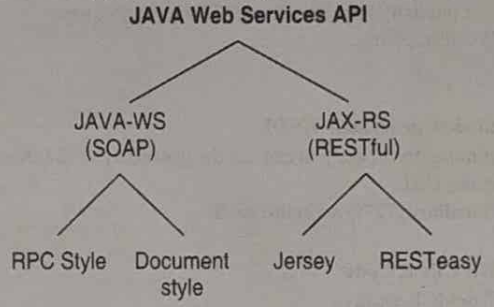


Fig. 4.7.1 : Java Web Service API

4.7.1 Writing a Java web service client

- Creating JAX-WS example is an easy task because it requires no extra configuration settings.
- JAX-WS API is inbuilt in JDK, so you don't need to load any extra jar file for it. Let's see a simple example of JAX-WS example in RPC style.
- There are created 4 files for hello world JAX-WS example:
  1. HelloWorld.java
  2. HelloWorldImpl.java
  3. Publisher.java
  4. HelloWorldClient.java
- The first 3 files are created for server side and 1 application for client side.

Unit IV End Sem.

JAX-WS Server Code

HelloWorld.java

```
importjavax.jws.WebMethod;
importjavax.jws.WebService;
importjavax.jws.soap.SOAPBinding;
importjavax.jws.soap.SOAPBinding.Style;
//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld{
   @WebMethod String getHelloWorldAsString(String name);
}
```

HelloWorldImpl.java

```
importjavax.jws.WebService;
//Service Implementation
@WebService(endpointInterface = "com.abc.HelloWorld")
public class HelloWorldImpl implements HelloWorld{
   @Override
public String getHelloWorldAsString(String name) {
return "Hello World JAX-WS " + name;
   }
}
```

Publisher.java

```
importjavax.xml.ws.Endpoint;
//Endpoint publisher
public class HelloWorldPublisher{
public static void main(String[] args) {
Endpoint.publish("http://localhost:7779/ws/hello", new
HelloWorldImpl());
      }
}
```

How to view generated WSDL

After running the code, you can see the generated WSDL file by visiting the URL:

http://localhost:7779/ws/hello?wsdl

JAX-WS Client Code

HelloWorldClient.java

```
import java.net.URL;
importjavax.xml.namespace.QName;
importjavax.xml.ws.Service;
public class HelloWorldClient{
public static void main(String[] args) throws Exception {
   URL url = new URL("http://localhost:7779/ws/hello?wsdl");

   //1st argument service URI, refer to wsdl document above
   //2nd argument is service name, refer to wsdl document above
QNameqname = new QName("http://abc.com/",
"HelloWorldImplService");
      Service service = Service.create(url, qname);
HelloWorld hello = service.getPort(HelloWorld.class);
System.out.println(hello.getHelloWorldAsString("Java web
service client RPC "));
   }
}
```

Output

```
Hello World JAX-WS Java web service client RPC
```

4.8 STRUTS

Overview

GQ. What are Struts? **(2 Marks)**

Definition : Struts is an open source framework which is developed by the "Apache Software Foundation" under the project named as Jakarta. Struts framework helps to create web related applications for java environment with MVC2 architecture.

- In 2004, the framework "struts 1" was released but it failed to fulfill the requirements of users as per current trends.
- "Struts 2" is then developed by Apache with OpenSymphony.
- We can say : Struts 2 = Webwork2 + Struts 1
- Webwork2 is the framework developed by OpenSymphony.
- In Struts 2 there is support for validation checking, AJAX and integration supports with different frameworks like Spring, Hibernate, Tiles and so on.

4.8.1 Features of Struts 2

GQ. Write features of Struts 2. **(2 Marks)**

- Struts 2 provides many important features which were not as a part in struts 1. The important features of struts 2 framework are as follows :

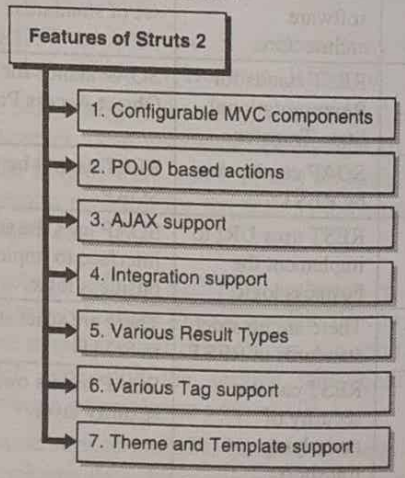


Fig. 4.8.1 : Features of Struts 2

1. Configurable MVC components

- In struts 2 framework, the information of all the components like view and action is given in struts.xml file.
- Whenever there is requirement of change in any information, we just have to make changes in the struts.xml file.

2. POJO based actions

- In struts 2, the action class is a Plain Old Java Object, shortly called as POJO.
- That means it is a simple java class. There is no need to extend a class or implement any interface.

3. AJAX Support

- Struts 2 have a strong support to AJAX technology. It helps to send an asynchronous request (non-blocking request).
- Instead of sending the whole data, only necessary data is sent which enhances the processing speed.

4. Integration Support

The integration of various frameworks like Spring, Hibernate, Tiles is very easy in struts 2 application.

5. Various Result Types

As result it is possible to use various technologies like JSP, Freemarker, and Velocity in struts 2 application.

6. Various Tag Support

There are number of tags provided by struts 2 like UI, Data and control tags. These tags provide various functionalities to Struts 2 application.

7. Theme and Template Support

- There are three types of themes supported in struts 2 application; XHTML, Simple and CSS_XHTML.
- The default theme is XHTML. The look and feel is set with the help of these themes.

4.8.2 Architecture of Struts

GQ. Explain architecture of Struts and Request lifecycle. **(8 Marks)**

- The architecture of Struts is purely MVC (Model-View-Controller architecture) based. The MVC architecture we have already seen in previous section.
- There are five core components in the MVC architecture of Struts 2.

Architecture Components

1. Actions
2. Interceptors
3. Value stack/OGNL
4. Results/Result types
5. View technologies

- There is slight difference between Struts 2 and traditional MVC framework in the manner that the role of model is taken by action instead of controller.
- The Fig. 4.8.1 depicts the MVC architecture of Struts 2. The Model, View and Controller are implemented as follows:
  - (1) **Model :** Implemented with actions.
  - (2) **View :** Implemented with combination of result types and results.
  - (3) **Controller :** implemented with a Struts2 dispatch servlet filter and interceptors
- The functionality of value stack and OGNL (Object-Graph Navigation Language) generally is to provide common thread, linking and supporting the integration in between different application components.

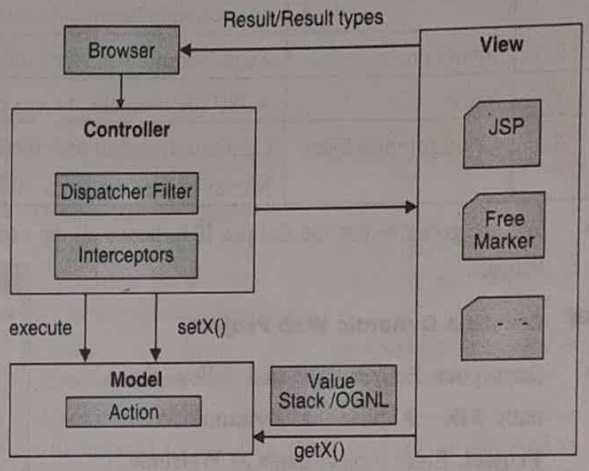


Fig. 4.8.2 : Struts Architecture

- Apart from all these components, there is vast information regarding the configuration. Configuration for the web application, and also configuration for various elements like actions, results, interceptors etc.

Request life cycle

- Based on the above Fig 4.8.2 the request life cycle in Struts 2 can be explained as follows:
- Request is sent by user to server for some resource (i.e. web pages).
- This request is accepted by the FilterDispatcher and appropriate action is determined.
- Functionalities of Configured interceptors like validation, file upload etc. are applied.

Unit IV End Sem.

- To carry out the requested operation, selected action is implemented.
- Another time, the configured interceptors are applied to execute any post-processing if necessary.
- Finally the View generates the result and returned it to the user.

4.8.3 Executing an Application of Struts

GQ. Write steps to execute an application in Strut. **(8 Marks)**

- When the HTML form is submitted in a Struts 2 web application, the input which is sent to java class file named "Actions" is controlled by the Controller.
- As soon as the execution of "Actions" is completed, a resource is selected by the Result to render the response.
- The result is usually in the form of JSP but sometimes it may be Java Applet, an Excel spreadsheet or even a PDF file.
- Now we will see a simple example to print Welcome message. The application will accepts name from user and display it with Welcome message.

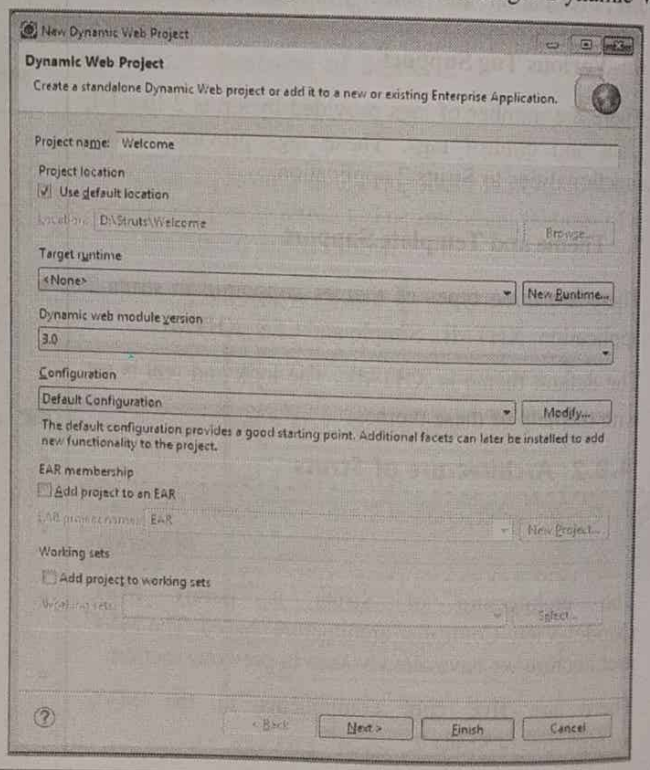
- For Strut 2 project, following four components are created :

Sr. No.	Components	Description
1	Action	Has to create Action class. This class will have the business logic as well it will control the interaction between user, model and view
2	Interceptors	As per requirement, new interceptor is created or existing is used. It is part of Controller
3	View	A JSP file is created to accept user input and display the final result.
4	Configuration Files	Configuration files are created to map the Actions, View and Controllers. Names of files are struts.xml, web.xml, struts. properties.

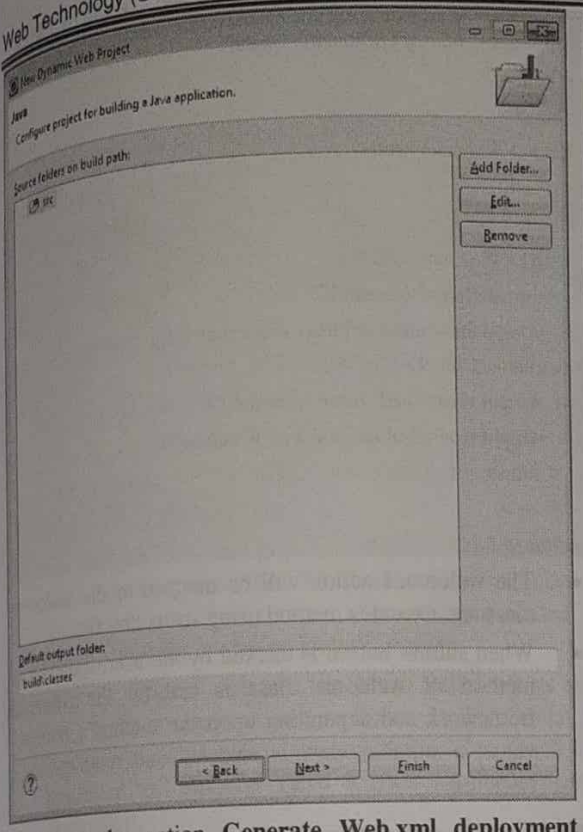
- We are going to use the Eclipse IDE, hence all the necessary components are generated under a single Dynamic Web Project.

Create a Dynamic Web Project

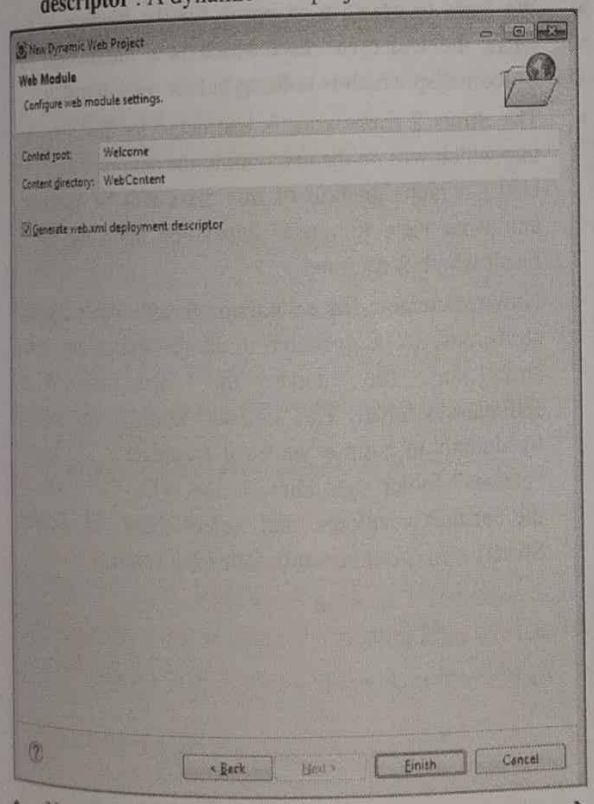
- Start your Eclipse IDE and follow the path **File → New → Dynamic Web Project.** Enter project name as **Welcome** and set remaining options as per the following screen :



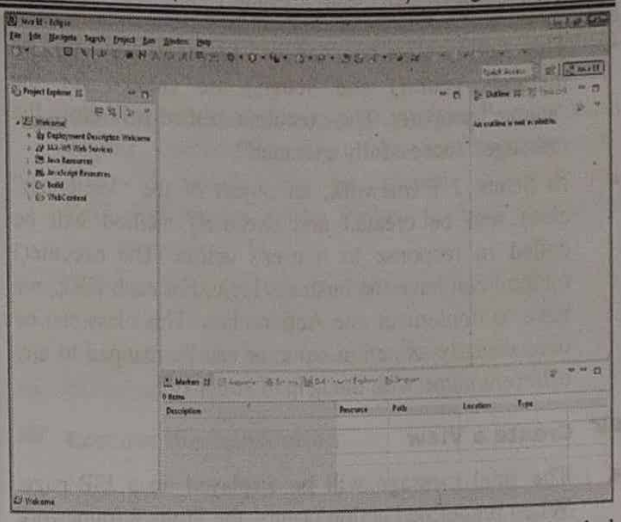
- Select next from following screen



- Select the option **Generate Web.xml deployment descriptor** . A dynamic web project will be created.



- Now follow the path **Windows → Show View → Project Explorer**, to see the project window. It will look like a follows:



- Now copy all the following files from downloaded struts 2 lib folder C:\struts-2.2.3\lib to our project's WEB-INF\lib folder.

```
commons-fileupload-x.y.z.jar
commons-io-x.y.z.jar
commons-lang-x.y.jar
commons-logging-x.y.z.jar
commons-logging-api-x.y.jar
freemarker-x.y.z.jar
javassist-.xy.z.GA
ognl-x.y.z.jar
struts2-core-x.y.z.jar
xwork-core.x.y.z.jar
```

☞ Create Action Class

- Creating Action class is the main part in Struts 2 application as we have to implement the important business logic in action class.
- Create a java file **welcome.java** under **Java Resources > src** with a package named **mypack**.
- When click event is fired by the user, the Action class will respond to the user. The methods in the Action class will be executed and result will be returned in string format.
- A specific JSP page will be rendered depending upon the result value.

```
package mypack;
public class welcome{
  private String sname;
  public String execute() throws Exception {
    return "successfully executed";
  }
  public String getName() {
    return sname;
  }
  public void setName(String sname) {
    this.sname = sname;
  }
}
```

Unit IV End Sem.

- This Action class has "sname" property. The standard methods getter() and setter() are created for the "sname" property. The execute() method will return the message "successfully executed".
- In Struts 2 framework, an object of the "Welcome" class will be created and execute() method will be called in response to a user's action. The execute() method can have the business logic. For each URL, we have to implement one Action class. This class can be used directly as action name or can be mapped to any different name with the help of struts.xml.

☞ Create a View

- The final message will be displayed on a JSP page. When predefined action occurs, the Struts 2 framework gives class to this JSP page. This mapping will be defined in struts.xml file.
- Create jsp file named as **Welcome.jsp** in the **WebContent** folder in the eclipse project.
- For this purpose, right click on the folder named **"WebContent"** in the project explorer and select **New >JSP File**

```
@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
<title>Welcome</title>
</head>
<body>
  Welcome to my web page <s:property value="sname"/>
</body>
</html>
```

- The taglib directive is used to inform the Servlet container that page in which it is declared is using the Struts 2 tags and these tags will be preceded by "s".
- The s:property tag displays the value of action class property "sname"> which is returned by the method getName()of the welcome class.

☞ Create main page

- Now we have to create index.jsp in the WebContent folder.
- This file will be used to provide as the initial action URL where a user can click to instruct the Struts 2 framework to call the defined method of the welcome class and make the welcome.jsp view.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Welcome</title>
</head>
<body>
  <h1>Welcome</h1>
  <form action="welcome1">
    <label for="name">Please enter your
     name</label><br/>
    <input type="text" name="sname"/>
    <input type="submit" value="Welcome"/>
  </form>
</body>
</html>
```

- The welcome1 action will be mapped to the welcome class and executes method using struts.xml file.
- When submit button is clicked by the user, the execute method of welcome class is run by the Struts 2 framework and depending upon the method's returned value, a suitable view is selected and rendered as a response.

☞ Configuration Files

- We have to create a mapping for the binding of the URL, the welcome class which is a Model and the welcome.jsp which is nothing but the view together.
- The Struts 2 framework is instructed by this mapping that which will be the responding the action of user (the URL), which function of this class will be executed, and what view to render depending upon the String result which is returned.
- Now we create a file called struts1.xml. Struts 2 needs struts1.xml to be available in classes folder. So create struts1.xml file under the WebContent/WEB-INF/classes folder. The "classes" folder is not created by default in eclipse we have to create it. To create "classes" folder right click on the WEB-INF folder in the project explorer and select New → Folder. Struts1.xml should contain following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="welcome1"
```

```
    class="welcome"
    method="execute">
    <result name="successfully
Executed">/welcome.jsp</result>
  </action>
 </package>
</struts>
```

- We are running in development environment and desire to see log messages. To enable this we have used struts.devMode and set it to true. In our example, we named our action as "welcome1" which is corresponding to the URL /welcome1.action and is backed up by the welcome.class.
- The execute() function of welcome.class is the function that is executed when the URL /hello.action is invoked. If the execute method returns "successfully executed", then we shift the control to HelloWorld.jsp.
- Now we create a web.xml file which will work as an entry point for any request to Struts 2. The Struts 2 application contains the entry point as a filter which is described in web.xml (also known as deployment descriptor).
- Therefore we will describe an entry of org.apache.struts2.dispatcher.FilterDispatcher class in web.xml. Eclipse had already created a Skelton web.xml file for you when you created the project. Therefore we modify it as given follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id="WebApp_ID" version="3.0">
 <display-name>Struts 2</display-name>
 <welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>
 <filter>
  <filter-name>struts2</filter-name>
  <filter-class>
   org.apache.struts2.dispatcher.FilterDispatcher
  </filter-class>
 </filter>
 <filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/*</url-pattern>
 </filter-mapping>
</web-app>
```

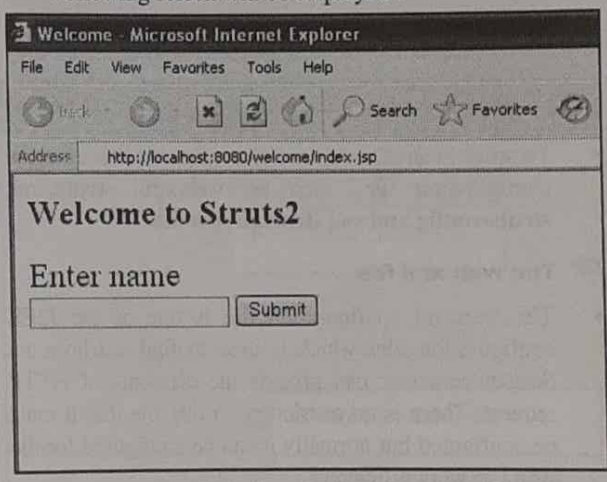
Enable Detailed Log

- **logging.properties** file can be generated to facilitate logging functionality when we are working with Struts 2 which can be placed inside the **WEB-INF/classes** folder. Following lines should be kept into property file:

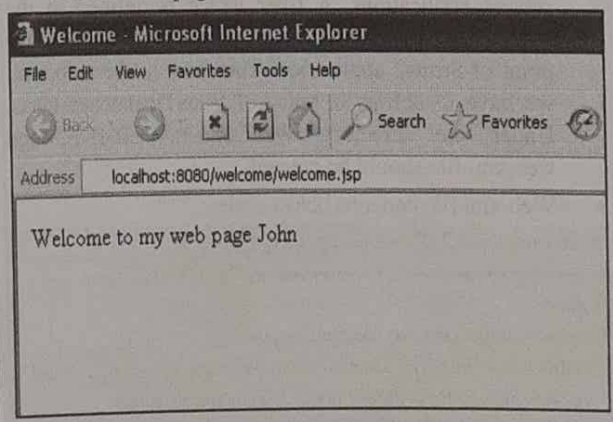
```
org.apache.catalina.core.ContainerBase.[Catalina].level
= INFO
org.apache.catalina.core.ContainerBase.[Catalina].handlers = \
java.util.logging.ConsoleHandler
```

Execute the Application

- Now we have to create a War file to be stored in Tomcat's Webapps directory. To create a WAR file right click on the project name and select **Export > WAR File**.
- Then store it in tomcat's webapps directory. Now, start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp.
- Following screen will be displayed :



- Enter a value "John" and submit the page. You should see the next page



- To execute program via action that you are going to execute you can just define **myindex** as an action in

Unit IV End Sem.

struts.xml file and in such case you can call index page as http://localhost:8080/welcome/myindex.action. Defining index as an action

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="myindex">
        <result >/index.jsp</result>
    </action>
    <action name="welcome1"
        class="mypack.welcome"
        method="execute">
        <result name="success">/welcome.jsp</result>
    </action>
  </package>
</struts>
```

4.8.4 Configuration

GQ. Explain configuration in Struts. **(4 Marks)**

- To work in strut2 you have to configure some essential configuration files such as: **web.xml, struts.xml, struts-config.xml** and **struts.properties.**

The web.xml file

- The web.xml configuration file is one of the J2EE configuration files which is used to find out how the Servlet container can process the elements of HTTP request. There is no restriction on this file that it must be configured but normally it can be configured for the strut2 to work efficiently.
- An entry point is provided by this file for various web related applications. A filter which is defined in the deployment descriptor (web.xml), works as an entry point of Struts2 application. Therefore in the web.xml, we have to define an entry of FilterDispatcher class. Under the directory WebContent/WEB-INF, the web.xml file should be created.
- Web.xml file contains below code.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id="WebApp_ID" version="3.0">
  <display-name>Example - Struts 2</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

- Note that in the filter-mapping tag we map the Struts 2 filter to /* which indicate that all URLs will be parsed by the struts filter.

The struts.xml file

- This file includes the configuration information that you will be changing as actions are developed. This file can be used to delete the previous default settings and add new settings for an application. This file can be placed inside the folder **WEB-INF/classes**.
- Consider the struts.xml file which was created in the welcome example.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="welcome1"
        class="welcome"
        method="execute">
        <result name="Successful">/welcome.jsp</result>
    </action>
  </package>
</struts>
```

- In the above code DOCTYPE must have the correct value. <struts> tag is known as the root element and inside this tag we can place many packages using the <package> tag. When we have big project then that project can be divided into different modules. This division and modularization can be done with the help of <package>tag.
- Consider project which contains three domains such as Bank_applicaiton, Users_application, Employee_application.

- For this we have to create three packages and store associated actions in the corresponding package.

Attributes of package tag

- The package tag has the following attributes:

Attribute	Description
name	The unique name for the package
Extends	It can be used to know which package extends from this package. Struts-default is the base package for all packages.
Abstract	If it set to true, then package is not accessible to end user.
namespace	Unique namespace for the actions

- To see debug messages in the log file we have to set the **struts.devnMode** property.
- We define action tags related to every URL which we like to access. Also we define a class having execute () method which can be called when we access corresponding URL.
- Results are used to decide what should be returned to the browser after a particular action is executed. The name of a result is the string returned from the action. Results have optional **name** and **type** attributes. The default name value is "success".
- When Struts.xml grows bigger then one of the ways is to break it in modules under a package. Another way is that we can divide the file into different xml files and can import as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <include file="mystruts1.xml"/>
   <include file="mystruts2.xml"/>
</struts>
```

The struts-config.xml file

- The struts-config.xml is used for making link between the Model and View component of the MVC framework. The configuration file mainly contains following elements :

Sr. No.	Interceptor	Description
1.	struts-config	It is considered as the root node of the configuration file.
2.	form-beans	It is used for mapping purpose. It maps the ActionForm subclass to a name. This name is used all the way through the struts-config.xml file as ActionForm subclass.
3.	global forwards	The global forward avoids hard coding URLs on your web pages.
4.	action-mappings	In this interceptor you can declare form handlers.
5.	controller	Controller is used to configure the internal of struts.
6.	plug-in	The plug-in instructs the Struts where to find your properties files, which contain prompts and error messages.

Following is the sample struts-config.xml file:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">
<struts-config>
  <form-beans>
  <form-bean name="login" type="test.struts.LoginForm" />
```

Form Bean Definition

```
  </form-beans>
  <global-forwards>
  </global-forwards>
```

Global forward Definition

```
<action-mappings>
```

Action Mapping Definition

```
   <action
     path="/login"
     type="test.struts.LoginAction" >
<forward name="valid" path="/jsp/MainMenu.jsp" />
<forward name="invalid" path="/jsp/LoginView.jsp" />
   </action>
  </action-mappings>
  <controller
   contentType="text/html;charset=UTF-8"
```

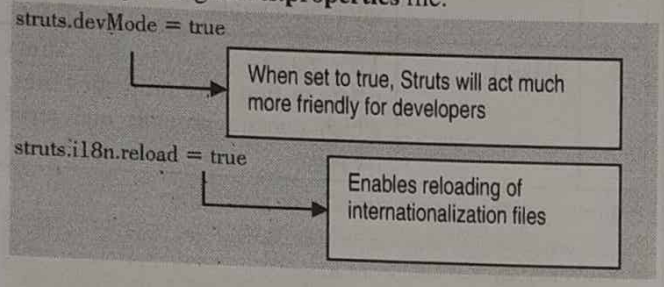
Controller Definition

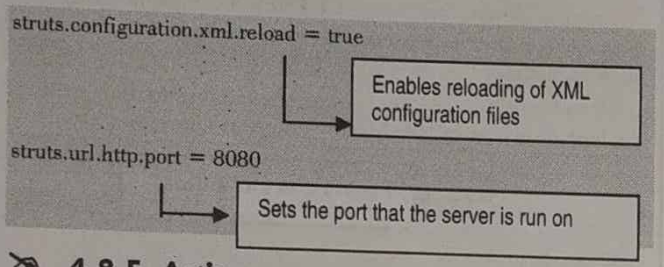
```
   debug="3"
   maxFileSize="1.618M"
   locale="true"
   nocache="true"/>
</struts-config>
```

Unit IV End Sem.

The struts.properties file

- You can modify the default behavior of the framework with the help of struts.properties file.
- Practically it is possible to configure all the properties present in struts.properties configuration file in the web.xml with the help of init-param as well as with the help of the constant tag which is present in configuration file (struts.xml).
- But if you want to store the things separate and more struts specific then you can create this file inside the folder **WEB-INF/classes**.
- Following are the some of the properties that you might change using **struts.properties** file:





4.8.5 Actions

GQ. Explain Actions in Struts with suitable example. **(8 Marks)**

- Actions in MVC framework can be considered as the core part of MVC. Whenever mapping from URL to particular action can be takes place then action provides the processing logic to process the user request.
- Actions have the two capabilities as follows:
  1. The action useful in transmitting the information from the request to the view, whether it's a JSP or other type of result.
  2. The action can be used to help out the framework in deciding which result should be given to the view that will be passed to the request as a response.

Create Action

- When we create action in strut2 it only requires one method which does not have any parameter and returns String or Result object. If this method is not declared in the class then execute() method can be used as default method.
- The Action interface is as follows:

```
public interface Action {
  public static final String successname = "success";
  public static final String noname = "none";
  public static final String ename = "error";
  public static final String inputname = "input";
  public static final String loginid= "login";
  public String execute() throws Exception;
}
```

Check the action method in welcome example:

```
package mypack;
public class welcome{
  private String sname;
  public String execute() throws Exception {
    return "success";
  }
  public String getName() {
    return sname;  }
  public void setName(String name) {
    this.sname = name;
  }
}
```

- Action method controls the view which is explained in the following example. To work out this we have to make changes in the execute method and class must extends the ActionSupport class.

Example

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
public class welcome extends ActionSupport{
  private String sname;
  public String execute() throws Exception {
    if ("INDIA".equals(sname))
    {
      return SUCCESS;
    }else{
      return ERROR;
    }
  }
  public String getName() {
    return sname;
  }
  public void setName(String name) {
    this.sname = name;
  }
}
```

- In the above example, the execute method contains sname attribute and the logic for checking whether the string sname is "INDIA". If it is true then it will return

"SUCCESS" otherwise returns "ERROR". Now, let us modify our struts.xml file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <constant name="struts.devMode" value="true" />
 <package name="mypack" extends="struts-default">
  <action name="welcome1"
   class="welcome "
   method="execute">
   <result name="success">/welcome.jsp</result>
   <result name="error">/error.jsp</result>
  </action>
 </package>
</struts>
```

Create a View

- Now we Create jsp file named as **Welcome.jsp** in the **WebContent** folder in the eclipse project. For this purpose, right click on the folder named **"WebContent"** in the project explorer and select **New → JSP File**.
- This file is used when the result is SUCCESS which is a String constant "success" as per the definition in Action interface.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
<title>Welcome</title>
</head>
<body>
 Welcome to my web page
</body>
</html>
```

- Also we have to create another jsp page for the action result ERROR which is a string constants "error" defined in Action interface. Following is the content of **error.jsp.**

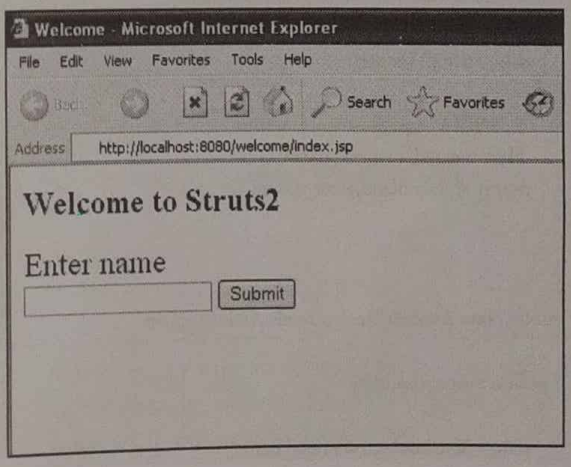
```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
<title>Access Denied</title>
</head>
<body>
 Access to this page is denied.
</body>
</html>
```

- Now we create **index.jsp** file which can be placed inside the WebContent folder. This file is used to provide the first action when the user fires a click event to instruct the Struts 2 framework to call the **execute()** method of the welcome class and render the welcome.jsp view.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Welcome</title>
</head>
<body>
 <h1>Welcome to Struts2</h1>
 <form action="welcome1">
  <label for="name">Enter name</label><br/>
  <input type="text" name="sname"/>
  <input type="submit" value="Ok"/>
 </form>
</body>
</html>
```

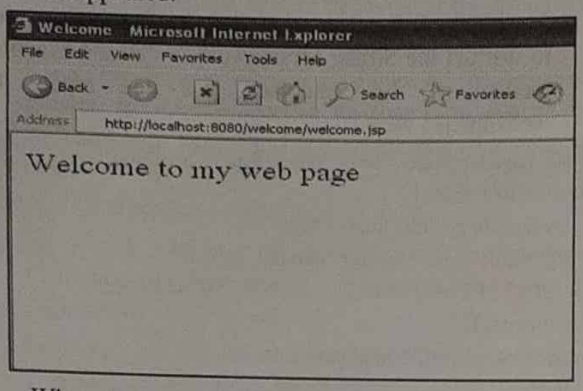
Execute the Application

- Now we have to create a War file to be stored in tomcat's webapps directory. To create a WAR file Right click on the project name and click **Export → WAR File**.
- Then store it in tomcat's webapps directory. Finally, start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp. This will give you following screen:

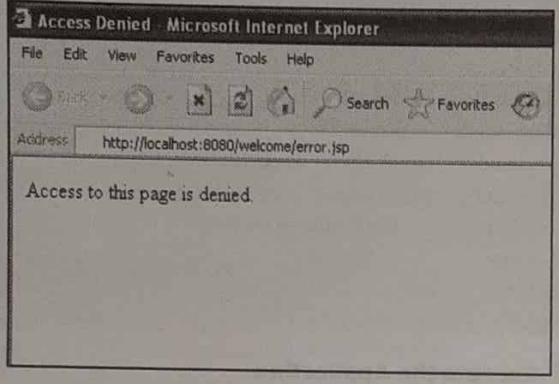


Unit IV End Sem.

- Let us enter a word as "INDIA" and following page will appeared:



When you enter word other than "INDIA" following page will be displayed.



☞ Create Multiple Actions

- You can also create multiple actions in a class to handle and process the different requests from the users and to give multiple URLs to them.

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
 class Mycommand extends ActionSupport{
   public static String success_name = SUCCESS;
   public static String error_name = ERROR;
 }
 public class welcome extends ActionSupport{
  ...
  public String execute()
  {
    if ("INDIA".equals(sname)) return
     Mycommand.success_name;
    return Mycommand.error_name;
  }
  ...
}
public class AnotherClass extends ActionSupport{
  ...
  public String execute()
  {
    return Mycommand.success_name;
  }
  ...
}
```

Now configure these actions in struts.xml file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
struts>
 <constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="welcome1"
     class="mypack.welcome "
     method="execute">
     <result name="success">/welcome.jsp</result>
     <result name="error">/error.jsp</result>
    </action>
    <action name="something"
     class="mypack.anotherClass"
     method="execute">
     <result name="success">/Something.jsp</result>
     <result name="error">/error.jsp</result>
    </action>
  </package>
</struts>
```

4.8.6 Interceptors

GQ. Explain Interceptors in Struts. **(8 Marks)**

☞ **Purpose :** The main aim of interceptors is to offer crosscutting functionality that can be kept at different location from the action method and also from the framework.

- Interceptors can be used for the purpose of execution of preprocessing and post-processing logic before and after action is called respectively and to perform alternate processing. It can handle the exceptions.
- In strut2 most of the features are implemented with the help of interceptors such as file uploading, exception handling etc.

☞ Struts 2 Framework Interceptors

- This framework offers many interceptors that are already configured and ready for use by the users. Following are the some of the interceptors used in strut2 framework.

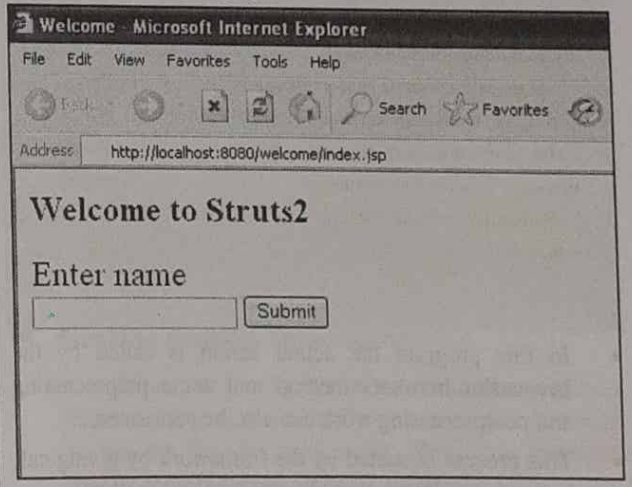
Sr. No.	Interceptor	Description
1.	Alias	Helps to set alias (nick names) to parameters across requests.
2.	checkbox	Handles checkboxes by assigning false value to unselected check boxes.
3.	conversionError	Stores data about error when converting strings to parameter types
4.	createSession	Creates HTTP session if not exists.
5.	debugging	Provides various types of debugging screens to the developer.
6.	execAndWait	Shifts the end-user to an intermediary waiting page when execution of action is happening in the background.
7.	Exception	Does the mapping of exceptions which are thrown from an action and result, facilitating automatic exception handling with the help of redirection.
8.	fileUpload	Helps for easy file uploading.
9.	i18n	Handles track of the selected locale in user session.
10.	Logger	Outputs name of the action to be executed to provide easy logging.
11.	Params	Sets the request parameters on the action.
12.	Prepare	Handles the pre-processing work, like establishing database connections.
13.	Profile	Log the simple profiling information for the actions.
14.	Scope	Save and access the action's state in the session or application scope.
15.	ServletConfig	Provides the action with access to diverse information regarding servlet.
16.	timer	Gives simple profiling information about the duration which action takes to execute.
17.	token	Verify the action for a valid token to avoid duplicate form submission.
18.	Validation	Provides support regarding validation for actions

How to use Interceptors?

- Now we will see how to use an interceptor in our "welcome" program. We use two interceptors ; **timer** interceptor to compute how much time it takes to execute an action method and **params** interceptor to transfer the request parameters to the action.
- We will keep welcome.java, web.xml, welcome.jsp and index.jsp files same as of the in previous example but let us update the **struts.xml** file to add an interceptor as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="welcome1"
      class="mypack.welcome"
      method="execute">
      <interceptor-ref name="params"/>
      <interceptor-ref name="timer" />
      <result name="success">/welcome.jsp</result>
    </action>
  </package>
</struts>
```

- To create a WAR file Right click on the project name and click **Export → WAR File**. Then store it in tomcat's webapps directory.
- At the end , start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp. following screen will be displayed :



- Now enter any word in the provided text box and click on OK button to execute the defined action. Now the generate log will show following text :

Unit IV End Sem.

```
INFO: Server startup in 3539 ms
27/07/2017 4:40:53 PM
com.opensymphony.xwork2.util.logging.commons.CommonsLogger
info
INFO: Executed action [//welcome!execute] took 109 ms.
```

☞ Create Custom Interceptors

- Creating custom interceptor in application is easy but you have to implement the Interceptor interface in the class. This can be illustrated as follows:

```
public interface Interceptor extends Serializable{
  void destroy();
  void init();
  String intercept(ActionInvocation invocation)
  throws Exception;
}
```

- The init() method is used to initialize the interceptor, and destroy() method is used to destroy the interceptor.
- To check whether the action has previously called or not the **ActionInvocation** object is used. This object offer access to the runtime environment.

☞ Create Interceptor Class

Let us create following Interceptor1.java in Java Resources → src folder:

```
package mypack;
import java.util.*;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
public class Interceptor1 extends AbstractInterceptor {
  public String intercept(ActionInvocation invocation)throws
Exception{
    /* let us do some pre-processing */
    String message = "Pre-Processing";
    System.out.println(message);
    /* let us call action or next interceptor */
    String res = invocation.invoke();
    /* let us do some post-processing */
    message = "Post-Processing";
    System.out.println(message);
    return res;
  }
}
```

- In this program the actual action is called by the **invocation.invoke()** method and some preprocessing and postprocessing work can also be performed.
- This process is started by the framework by giving call to the ActionInvocation object's invoke() method.
- Whenever the **invoke()** method is called, ActionInvocation consults its state and executes irrespective of the interceptor which comes next.
- After invocation of all the interceptors, the invoke() method executes the action.

☞ Create Action Class

Let us create a java file welcome.java under **Java Resources > src** with a package name **mypack** with the contents given below.

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
public class welcome extends ActionSupport{
  private String sname;
  public String execute() throws Exception {
    System.out.println("Inside action....");
    return "success";
  }
  public String getName() {
    return sname;
  }
  public void setName(String name) {
    this.sname = name;
  }
}
```

☞ **Create a View :** Let us create the below jsp file **welcome.jsp** in the WebContent folder in your eclipse project.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
<title>welcome </title>
</head>
<body>
Welcome to my webpage <s:property value="sname"/>
</body>
</html>
```

☞ Create main page

- In the WebContent directory, we have to create index.jsp .
- This index.jsp file will be work as the initial action URL where a user fire the click event to instruct the Struts 2 framework to invoke the defined method of the welcome class and make the welcome.jsp view.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Welcome</title>
```

```
</head>
<body>
 <h1>Welcome</h1>
 <form action="welcome1">
  <label for="name">Please enter your name</label><br/>
  <input type="text" name="sname"/>
  <input type="submit" value="Welcome"/>
 </form>
</body>
</html>
```

- The **welcome1** action defined in the above view file will be mapped to the welcome class and its execute method using struts.xml file.

Configuration Files

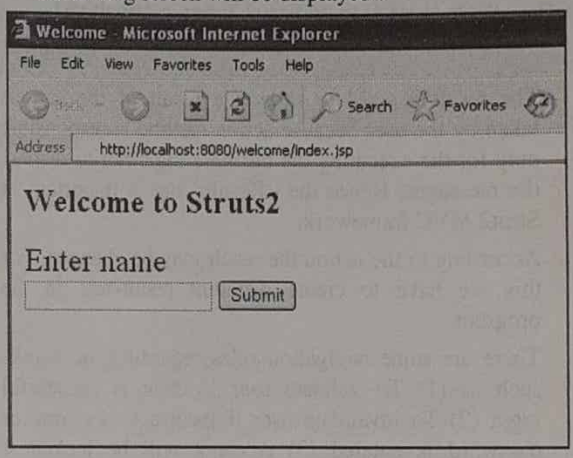
- To call interceptor we need to register it first. To register interceptor, the <interceptors>...</interceptors> tags are put inside the <package> tag in **struts.xml** file.
- We can define multiple interceptors in the same way as defined above and then we can call them as per the requirement.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <constant name="struts.devMode" value="true" />
 <package name="mypack" extends="struts-default">
  <interceptors>
   <interceptor name="Interceptor1"
    class="mypack.Interceptor1" />
  </interceptors>
  <action name="welcome1"
   class="mypack.welcome"
   method="execute">
   <interceptor-ref name="params"/>
   <interceptor-ref name="Interceptor1" />
   <result name="success">/welcome.jsp</result>
  </action>
 </package>
</struts>
```

- The web.xml file needs to be created under the WEB-INF folder under WebContent as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id="WebApp_ID" version="3.0">
 <display-name>Struts 2</display-name>
 <welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>
 <filter>
  <filter-name>struts2</filter-name>
  <filter-class>
   org.apache.struts2.dispatcher.FilterDispatcher
  </filter-class>
 </filter>
 <filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/*</url-pattern>
 </filter-mapping>
</web-app>
```

- To create a WAR file Right click on the project name and click **Export → WAR File**. Then save this file in webapps folder of tomcat.
- Now, start he service of Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp.
- Following screen will be displayed :



- Enter any text in the given text box and click on OK button to execute the defined action. Now if you will check the log generated, you will find following text at the end:

```
Pre-Processing
Inside action....
Post-Processing
```

Stacking multiple Interceptors

- It is difficult to configure each and every interceptor separately for every action. To overcome this overhead we can use the interceptor stack. Here is an example, directly from the struts-default.xml file:

```
<interceptor-stack name="InterceptorStack1">
 <interceptor-ref name="exception"/>
 <interceptor-ref name="servlet-config"/>
```

Unit IV End Sem.

```
<interceptor-ref name="prepare"/>
<interceptor-ref name="checkbox"/>
<interceptor-ref name="params"/>
<interceptor-ref name="conversionError"/>
</interceptor-stack>
```

- **The InterceptorStack1** can be used in your configuration as shown below.
- It can be placed inside the <package .../> node. All the <interceptor-ref .../> tags reference either an interceptor or an interceptor stack which has been configured prior the current interceptor stack. It is must to ensure that the interceptor name is distinct in all interceptors and interceptor stack configurations when configuring the first interceptors and interceptor stacks.
- The process of applying interceptor stacks is similar to the process of applying interceptor onto the action. Actually, we use exactly the same tag:

```
<action name="welcome1" class="mypack.Mycommand">
  <interceptor-ref name="InterceptorStack1"/>
  <result>view.jsp</result>
</action
```

4.8.7 Result Types

GQ. Write short note on Result Types in AngularJS? **(2 Marks)**

- The result is used to show the result of the actions taken by the user because action method is responsible only for the executing the business logic not to display the messages. Hence the <Result> tag is important in Strut2 MVC framework.
- According to the action the result may be changed. For this we have to create different result-sets in the program.
- There are some navigation rules regarding the results such as (1) To validate user If there is successful login (2) To invalidate user if incorrect username or password is entered (3) Account will be locked if multiple attempts are tried.
- For this situation we have to create three different views to give these outcomes to users.
- Struts have the number of types of result. Few of them are explained briefly in below:

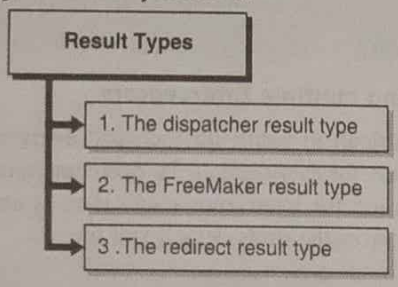


Fig. 4.8.3 : Result types

1. The dispatcher result type

Dispatcher result type is the default result type that means in the program if there is no any result type defined then it can take dispatcher result type. It can be used to send the result to Servlet, HTML page, JSP etc. which are running on the server. For doing this it will uses the *RequestDispatcher.forward()* method.

Example

```
<result name="success">
  /HelloWorld.jsp
</result>
```

Inside the <result...> element we can indicate the JSP file using a <param name="location"> tag as follows:

```
<result name="success" type="dispatcher">
  <param name="location">
    /welcome.jsp
  </param >
</result>
```

2. The FreeMaker result type

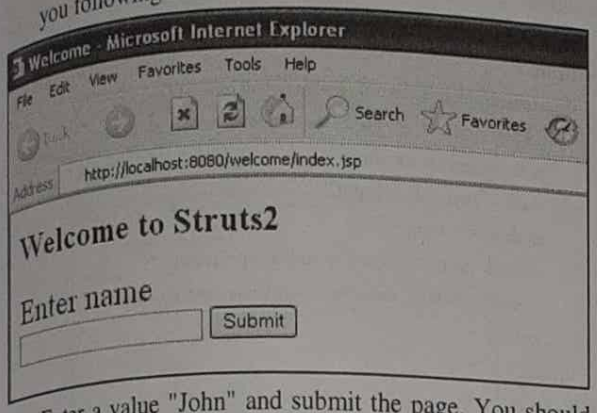
- Following is an example in which we are going to see how to use **FreeMaker** as the view technology. Freemaker is one of the famous templating engine which is used to produce output with the help of predefined templates. Let us create a Freemaker template file called **welcome.fm** with the following contents:

```
Welocme ${sname}
```

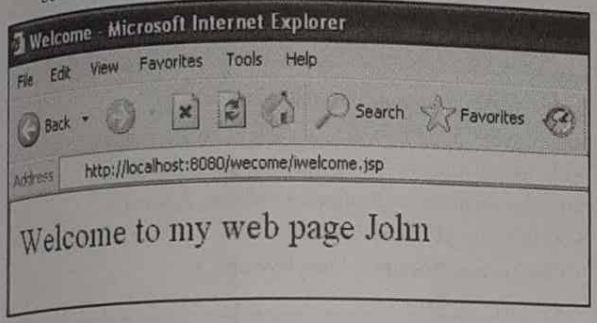
- Here above file is a template where **sname** is a parameter which will be passed from outside using the defined action. You will keep this file in your CLASSPATH. Next, let us modify the **struts.xml** to specify the result as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="welcome1"
      class="mypack.welcome"
      method="execute">
      <result name="success" type="freemarker">
        <param name="location">/welcome.fm</param>
      </result>
    </action>
      </package>
</struts>
```

- Other files such as welcome.java, welcome.jsp and index.jsp are taken from the above explained example. Now we have to create a War file to be stored in tomcat's webapps directory. To create a WAR file Right click on the project name and click **Export → WAR File**.
- Then store it in tomcat's webapps directory. Finally, start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp. This will give you following screen:



- Enter a value "John" and submit the page. You should see the next page



3. The redirect result type

- For generating a new request to such a location which is provided by the user the **redirect** result type uses the response.sendRedirect *()* method.
- We can give the location in the body of the <result...> element or as a <param name="location"> element. Redirect also supports the **parse** parameter. Here's an example configured using XML:

```
<action name="welcome1"
 class="mypack.welcome"
 method="execute">
 <result name="success" type="redirect">
   <param name="location">
    /grandwelcome.jsp
   </param >
 </result>
</action>
```

4.8.8 Validation

GQ. Explain Validation in AngularJS? **(4 Marks)**

- Validation is used to assure validity of source code by applying set of rules on it before an action takes place.
- Validation can be done either at the client side or at the server side. Client side validation is generally achieved using JavaScript. But one should not rely upon client side validation alone. Best practice suggests that the validation should be introduced at all levels of your application framework.
- Here we will take an example of **student** whose name and number would be saved using a simple page and we will put two validations to make sure that user always enters a name and PId in between 1 and 10. So let us start with the main JSP page as follows:

☞ Create main page

- Let us write main page JSP file **index.jsp**, which will be used to collect student related information mentioned above.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Student Form</title>
</head>
<body>
  <s:form action="studinfo" method="post">
    <s:textfield name="name" label="Name" size="20" />
    <s:textfield name="PId" label="PID" size="20" />
    <s:submit name="submit" label="Submit" align="center" />
  </s:form>
</body>
</html>
```

☞ Create Views

- Now we create a JSP file named as **success.jsp** which can be called if action returns SUCCESS.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

Unit IV End Sem.

```
<head>
<title>Success</title>
</head>
<body>
 Student Information is saved successfully.
</body>
</html>
```

☞ Create Action

- Create class **student**, and then add a method called **validate()** as shown below in **student.java** file. Student class must extend the **ActionSupport** class. If it does not extend then validate method will not be executed.

```
package com.tutorialspoint.struts2;
import com.opensymphony.xwork2.ActionSupport;
public class student extends ActionSupport{
  private String sname;
  private int  PId;

  public String execute()
  {
     return SUCCESS;
  }
  public String getSname() {
     return sname;
  }
  public void setSname(String name) {
     this.sname = name;
  }
  public int getPID() {
     return PId;
  }
  public void setPID(int number) {
     this.PId = number;
  }
  public void validate()
  {
    if (name == null || name.trim().equals(""))
    {
      addFieldError("name","The name is required");
    }
    if (PId < 11 || PId > 10)
    {
      addFieldError("PId "," PID  must be in between 1 and 10");
    }
  }
}
```

- As shown in the above example, the validation method checks the 'Name' field has a value or not. If value has not given, we add a field error for the 'Name' field with a custom error message and also check if entered value for 'PId' field is in between 1 and 10 or not, if this condition does not satisfy we add an error above the validated field.

☞ Configuration Files

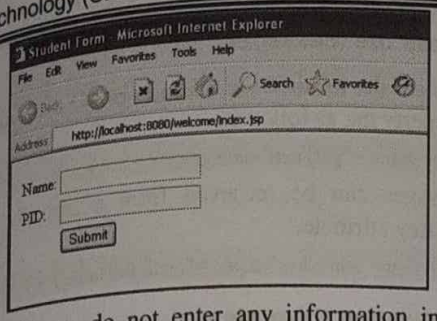
- Now let us keep all these files together using the **struts.xml** configuration file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="studinfo"
      class="mypack.student"
      method="execute">
      <result name="input">/index.jsp</result>
      <result name="success">/success.jsp</result>
    </action>
  </package>
</struts>
```

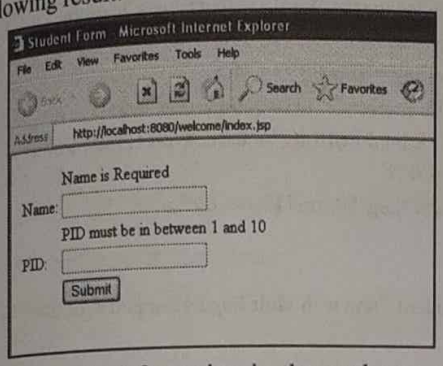
Following is the content of **web.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id="WebApp_ID" version="3.0">
  <display-name>Struts 2</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

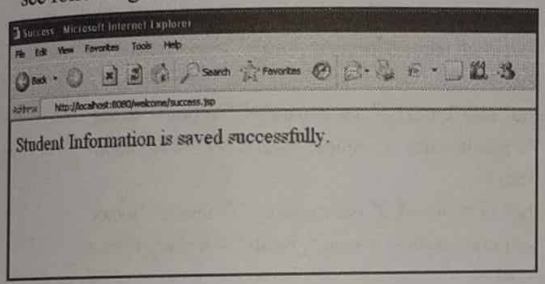
- To create a WAR file Right click on the project name and click **Export > WAR File**. Then store it in tomcat's webapps directory. Finally, start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp. This will give you following screen :



- Now if you do not enter any information inside the textboxes and click on Submit button, you will see following result:



- Now enter the information in the textboxes, you will see following result:



How this validation works?

- During execution of program when the users clicks on submit button then strut2 calls validate() method and if one of statements given inside the validate method becomes true then addFieldError() method is called. If the validation gets failed then it will redisplay the index.jsp file.
- The addFieldError () method takes two parameters. The first parameter is the name of the form field on which the error can be placed and the second parameter is the error message to the users screen.

```
addFieldError("sname","The name is required");
```

- To manage the return value of **input** we have to insert the below result to our action node in **struts.xml**.

```
<result name="input">/index.jsp</result>
```

XML Based Validation

- Another way of performing validation is to place the xml file after the action class. The XML based validation offers multiple kinds of validations such as range validation, required field validation, string length validation, regex validation etc.
- In the validation xml file can be stored with the name as '[action-class]'-validation.xml. Hence we create a file called student-validation.xml as given below:

```
<!DOCTYPE validators PUBLIC
"-//OpenSymphony Group//XWork Validator 1.0.2//EN"
"http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
  <field name="sname">
    <field-validator type="required">
      <message>
        The name is required.
      </message>
    </field-validator>
  </field>
  <field name="PId">
    <field-validator type="int">
      <param name="min">1</param>
      <param name="max">10</param>
      <message>
        PID must be in between 1 and 10
      </message>
    </field-validator>
  </field>
</validators>
```

- Consider student class as given below which does not contain **validate** () method:

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
public class student extends ActionSupport{
  private String sname;
  private int PId;

  public String execute()
  {
    return SUCCESS;
  }
  public String getSname() {
    return sname;
  }
  public void setSname(String name) {
    this.sname = name;
  }
  public int getPId() {
    return PId;
  }
  public void setPId (int number) {
    this.PId = number;
  }
}
```

Unit IV End Sem.

- When you will run this application, it will produce same result as we received in our previous example.
- We have to separate the application code and the validation code to make it easy for understanding .We can use the xml files to store this configuration which will be beneficial for us.

4.8.9 Localization

GQ. Explain Localization in AngularJS. **(4 Marks)**

❒ **Definition :** Localization is the process of planning and implementing products and services in a way that can be easily recognized by any specific local language. It is also called as Internationalization (i18n).

- The word "i18n" is a short hand in which I indicates start letter ,"n" indicated the last letter and digit 18 indicated the excluding first and last letter there are 18 characters in between.

☞ Resource Bundles

- While writing an application to be used in different languages we do not require writing it in different forms for each language because resource bundles provide the facility to make application available for users in different languages.
- The resource bundles will include titles for application, messages for users, and other data in the language regarding user. Resource bundles are the files that contain the key/value pairs for the default language of application.
- The simplest naming format for a resource file is:

```
bundlename_language_country.properties
```

- Here **bundlename** could be ActionClass, Interface, SuperClass, Model, Package, Global resource properties. **language_country** indicates the country locale for example Spanish locale is denoted by es_ES and English (United States) locale is denoted by en_US etc. Here the optional country part can be skipped.
- To support numerous languages your application must have to preserve more than one property files analogous to that languages/locale and describe all the content in terms of key/value pairs.

(i) **global.properties :** By default English (United States) will be applied

(ii) **global_fr.properties :** This will be used for French locale.

(iii) **global_es.properties :** This will be used for Spanish locale.

☞ Access the messages

To show the localization text, **getText()** method is used in the property tag as follows:

```
<s:property value="getText('some.key')" />
```

Messages can be retrieved from a resource bundle using the **key** attribute.

```
<s:textfield key="some.key" name="textfieldName"/>
```

☞ Localization Example

Create **index.jsp** in multiple languages as given below :

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>student Form with Multilingual Support</title>
</head>
<body>
  <h1><s:text name="global.heading"/></h1>
  <s:url id="indexEN" namespace="/" action="locale" >
    <s:param name="request_locale" >en</s:param>
  </s:url>
  <s:url id="indexES" namespace="/" action="locale" >
    <s:param name="request_locale" >es</s:param>
  </s:url>
  <s:url id="indexFR" namespace="/" action="locale" >
    <s:param name="request_locale" >fr</s:param>
  </s:url>
  <s:a href="%{indexEN}" >English</s:a>
  <s:a href="%{indexES}" >Spanish</s:a>
  <s:a href="%{indexFR}" >France</s:a>
  <s:form action="studinfo" method="post" namespace="/">
    <s:textfield name="name" key="global.name" size="20" />
    <s:textfield name="PID" key="global.PID" size="20" />
    <s:submit name="submit" key="global.submit" />
  </s:form>
</body>
</html>
```

- When action declared in a class returns SUCCESS then success.jsp file is called. Following code is written inside the success.jsp file.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<title>Success</title>
</head>
<body>
 <s:property value="getText('global.success')" />
</body>
</html>
```

Action to take care of Locale

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
public class Locale extends ActionSupport{
 public String execute()
 {
   return SUCCESS;
 }
}
```

Action to submit the form

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
public class student extends ActionSupport{
 private String sname;
 private int PId;

 public String execute()
 {
   return SUCCESS;
 }

 public String getSname() {
   return sname;
 }
 public void setSname(String name) {
   this.sname = name;
 }
 public int getPId() {
   return PID;
 }
 public void setPId(int number) {
   this.PId = number;
 }
}
```

- Now we create following three **global.properties** files and place it inside the CLASSPATH:

global.properties

```
global.name = sname
global.PId = number
global.submit = Submit
global.heading = Select Locale
global.success = Successfully authenticated
```

global_fr.properties

```
global.name = Noms
global.PId=P identite
global.submit = Soumettre des
global.heading = Sé lectionnez Local
global.success = Authentifi  é avec success
```

global_es.properties

```
global.name = Nombre
global.PId = la pla identification
global.submit = Presentar
global.heading = seleccionar la configuracion regional
global.success = Autenticado correctamente
```

Now create **struts.xml** having two actions as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <constant name="struts.devMode" value="true" />
 <constant name="struts.custom.i18n.resources" value="global"
/>
 <package name="mypack" extends="struts-default"
namespace="/">
  <action name="studinfo" class="mypack.student"
method="execute">
   <result name="input">/index.jsp</result>
   <result name="success">/success.jsp</result>
  </action>
    <action name="locale"
   class="mypack.Locale"
   method="execute">
   <result name="success">/index.jsp</result>
  </action>
 </package>
</struts>
```

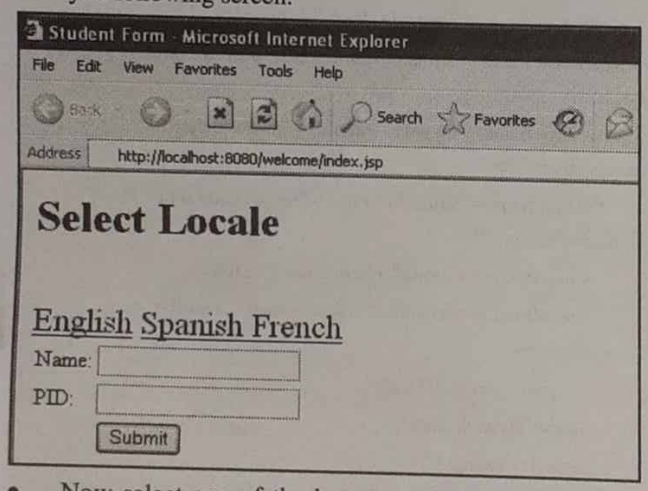
The web.xml file contents are as follow:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id="WebApp_ID" version="3.0">
 <display-name>Struts 2</display-name>
```

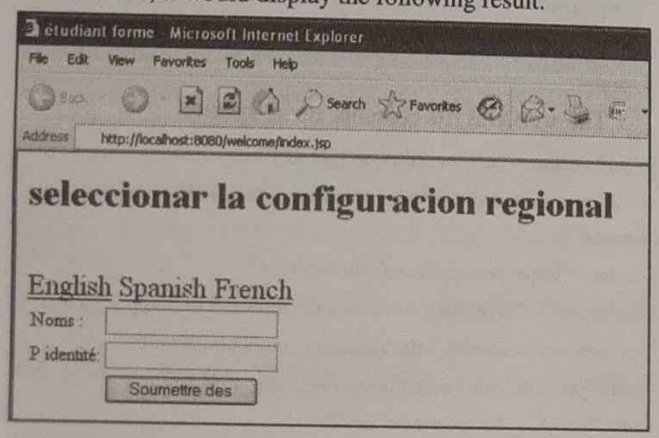
Unit IV End Sem.

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<filter>
  <filter-name>struts2</filter-name>
  <filter-class>
    org.apache.struts2.dispatcher.FilterDispatcher
  </filter-class>
</filter>
<filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
</web-app>
```

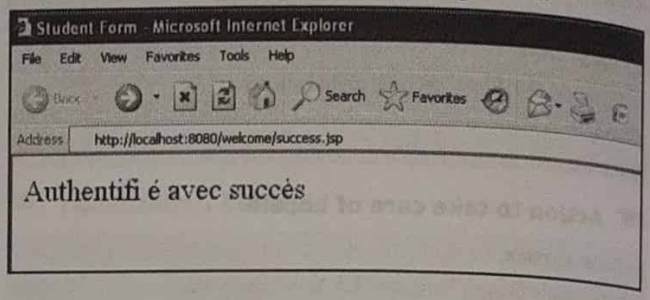
- To create a WAR file Right click on the project name and click **Export → WAR File**. Then store it in tomcat's webapps directory.
- Finally, start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp. This will give you following screen:



- Now select any of the languages, let us say we select **Frence**, it would display the following result:



- You can try with Spanish language as well. Finally, let us try to click **Submit** button when we are in France Locale, it would display the following screen:



4.8.10 Exception Handling

GQ. Write short note on Exception Handling in AngularJS ? **(2 Marks)**

- In struts you can handle exception very easily because struts provide mechanism to catch exception and according to exception users will be redirected to the corresponding error pages.
- For handling the exceptions struts uses "exception interceptor". You do not required to take any extra efforts to configure the exception interceptor as it is present inside the default stack, you can directly use it. Now consider our welcome example with some changes in welcome.java file. We are going to introduce the NullPointer exception in our java file.

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
public class welcome extends ActionSupport{
  private String sname;
  public String execute(){
    String a = null;
    a= a.substring(0);
    return SUCCESS;
  }

  public String getSname() {
    return sname;
  }
  public void setSname(String name) {
    this.sname = name;
  }
}
```

welcome.jsp contain the following content:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
<title>Welcome</title>
</head>
<body>
```

```
Welcome to my web page <s:property value="sname"/>
</body>
</html>
```

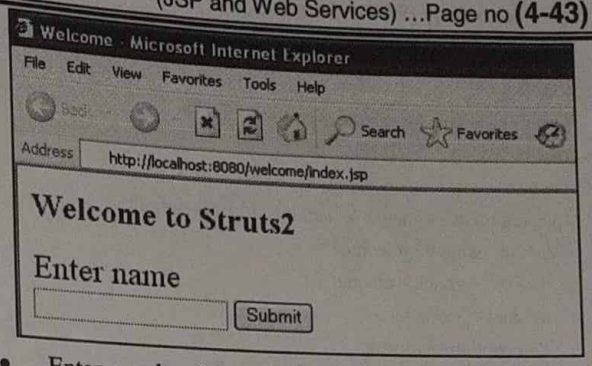
The **index.jsp** contains the following contents:

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Welcome</title>
</head>
<body>
  <h1>Welcome to Struts2</h1>
  <form action="welcome1">
    <label for="name">Enter name</label><br/>
    <input type="text" name="name"/>
    <input type="submit" value="Submit"/>
  </form>
</body>
</html>
```

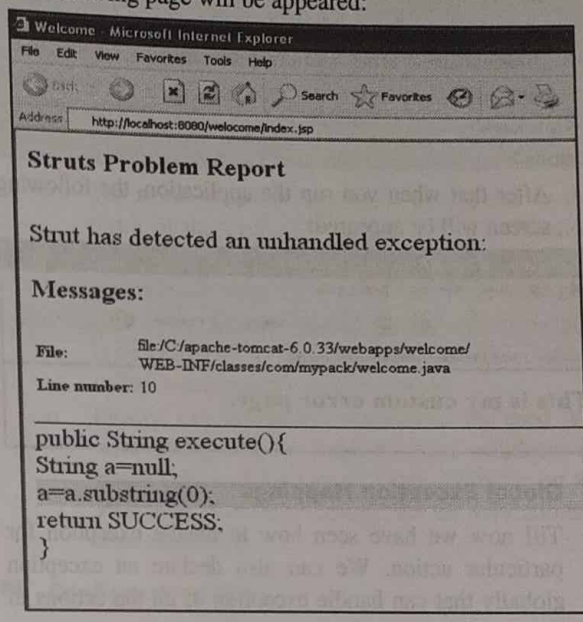
The **struts.xml** contains the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="welcome1"
      class="mypack.welcome"
      method="execute">
      <result name="success">/welcome.jsp</result>
    </action>
  </package>
</struts>
```

- To create a WAR file Right click on the project name and click **Export > WAR File**. Then store it in tomcat's webapps directory.
- Finally, start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp. This will give you following screen:



- Enter a value "welcome" and click on submit button. Following page will be appeared:



- Now our job is to create an error page for our Exception. So create a file **Error.jsp** which consist the following contents:

Unit IV End Sem.

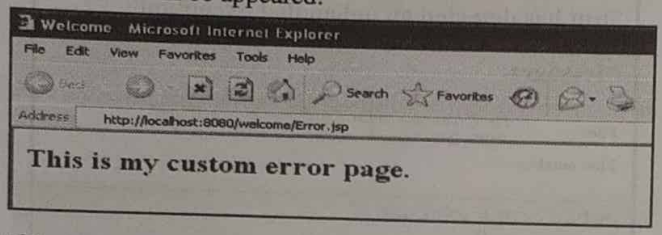
```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<Title>Error</title>
</head>
<body>
  This is my custom error page
</body>
</html>
```

To use this error page when exception occurs we have to configure it in struts.xml file. Implement following changes in order to configure struts.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
```

```
  "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <action name="welcome1"
      class="mypack.welcome"
      method="execute">
      <exception-mapping
exception="java.lang.NullPointerException"
      result="error" />
      <result name="success">/welcome.jsp</result>
      <result name="error">/Error.jsp</result>
    </action>
  </package>
</struts>
```

- After that when you run the application, the following screen will be appeared:



Global Exception Mappings

- Till now we have seen how to handle exception for particular action. We can also declare an exception globally that can handle exception of all the actions in the program.
- For example, to catch the NullPointerException exception, we could insert **<global-exception-mappings...>** tag in<package...> tag and its <result...> tag should be inserted within the <action...> tag in struts.xml file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.devMode" value="true" />
  <package name="mypack" extends="struts-default">
    <global-exception-mappings>
      <exception-mapping
exception="java.lang.NullPointerException"
      result="error" />
    </global-exception-mappings>
    <action name="welcome1"
      class="mypack.welcome "
      method="execute">
      <result name="success">/welcome.jsp</result>
      <result name="error">/Error.jsp</result>
    </action>
  </package>
</struts>
```

4.8.11 Annotations

GQ. What are Annotations in AngularJS? **(4 Marks)**

- In struts there are two ways for configuration. First is to configure the strus.xml file and the second way is to use the annotations feature. Use of this feature in struts achieves the Zero Configuration.
- To start using annotations in your project, copy all the following files from downloaded struts 2 lib folder C:\struts-2.2.3\lib to our project's WEB-INF\lib folder.
  (1) commons-fileupload-x.y.z.jar
  (2) commons-io-x.y.z.jar
  (3) commons-lang-x.y.jar
  (4) commons-logging-x.y.z.jar
  (5) commons-logging-api-x.y.jar
  (6) freemarker-x.y.z.jar
  (7) javassist-.xy.z.GA
  (8) ognl-x.y.z.jar
  (9) struts2-core-x.y.z.jar
  (10) xwork-core.x.y.z.jar
  (11) struts2-convention-plugin-x.y.z.jar
  (12) asm-x.y.jar
  (13) antlr-x.y.z.jar
- To understand the use of Annotation, we would have to reconsider our validation example.
- Here we will take an example of **student** whose name and **PId** would be saved using a simple page and we will put two validations to make sure that users always enters a name and PId between 1 to 10. So let us start with the main JSP page.

Create main page

Create a JSP file named **index.jsp**, which contain the student related information.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Student Form</title>
</head>
<body>
  <s:form action="studinfo" method="post">
```

```
<s:textfield name="sname" label="Name" size="20" />
<s:textfield name="PId" label="PID" size="20" />
<s:submit name="submit" label="Submit" align="center" />
</s:form>
</body>
</html>
```

Create Views

- If action return SUCCESS then we have use following JSP file named **success.jsp**.

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Success</title>
</head>
<body>
Student Information is saved successfully.
</body>
</html>
```

Create Action

- Create class **student**, and then add a method called **validate()** as shown below in **student.java** file. Student class must extend the **ActionSupport** class. If it does not extend then validate method will not be executed. In this class we are going to use the annotation.

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
import org.apache.struts2.convention.annotation.Action;
import org.apache.struts2.convention.annotation.Result;
import org.apache.struts2.convention.annotation.Results;
import com.opensymphony.xwork2.validator.annotations.*;
@Results({
  @Result(name="success", location="/success.jsp"),
  @Result(name="input", location="/index.jsp")
})
public class student extends ActionSupport{
  private String sname;
  private int PId;
  @Action(value="/studinfo")
  public String execute()   {
    return SUCCESS;
  }
  @RequiredFieldValidator( message = "The name is required" )
  public String getSname() {
    return sname;
  }
  public void setSname(String name) {
    this.sname = name;
  }
  @IntRangeFieldValidator(message = "PId must be in between 1
and 10",
                          min = "1", max = "10")
  public int getPId () {
    return PId;
  }
  public void setPId (int number) {
    this.PId = number;
  }
}
```

- The annotations used in this example are explained below:
  - Here the **Results** annotation is used which a set of results. It contains two result annotations.
  - The name of result annotation correspond to the result of the method execute(). There is location available to which a view should be provided analogous to value returned by execute() method.
  - Next is the **Action** annotation. The action method is decorated by it. The URL on which the action is called is accepted by Action method.
  - Finally, two **validation** annotations are used. We have configured the required field validator on **name** field and the integer range validator on the **PID** field. A custom message for the validations is also mentioned.

Configuration Files

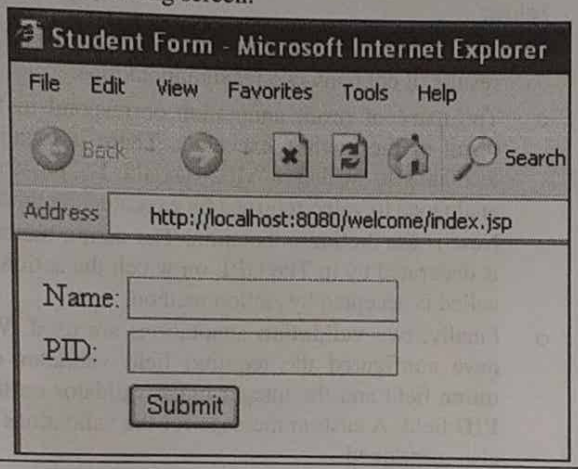
- Check the content of **web.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id="WebApp_ID" version="3.0">
  <display-name>Struts 2</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
    <init-param>
      <param-name>struts.devMode</param-name>
      <param-value>true</param-value>
```

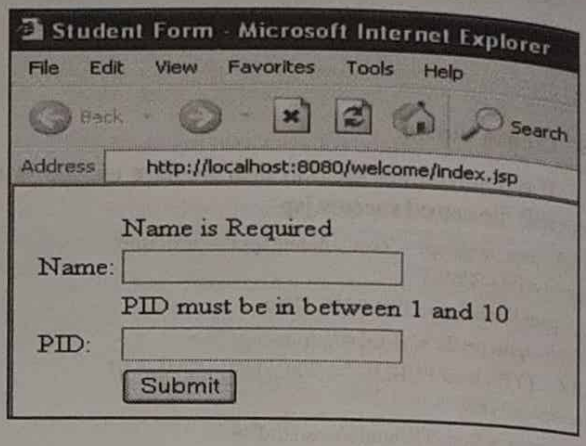
Unit IV End Sem.

```
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

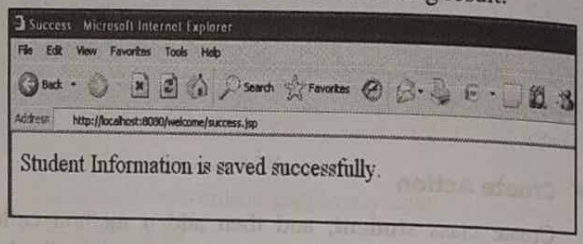
- To create a WAR file Right click on the project name and click **Export → WAR File**. Then store it in tomcat's webapps directory.
- Finally, start Tomcat server and try to access URL http://localhost:8080/welcome/index.jsp. This will give you following screen:



- Now do not enter any information inside the textbox, just click on **Submit** button. You will see following result:



- Now enter the correct information in the textbox, let us say name as "Arnav" and PId as 5, and finally click on **Submit** button. You will see following result:



...Chapter Ends

❑❑❑