# WEB TECHNOLOGY

## UNIT- 1

**web Technologies:-**

It is a term used to refer a group of languages to develop web applications or websites.

ex:- some of the languages are

HTML, CSS, Javascript, PHP, ASP, JSP, XML, servlets.

PHP, ASP, JSP → server side validation.

**website:**

→ A group of related webpages identified by a common domain name (used to refer name of particular website).

→ Every website has their own URL (Uniform Resource Locator).

structure:- http://www.vardhaman.org/structure.php

- http → protocol
- www.vardhaman.org → domain name
- structure.php → file name

http refers to Hyper text transfer protocol.

- .com = commercial
- .org = organization
- .gov = government
- .mil = military
- .gov.in = Indian government.

web page:-

It is single hyper text document connected to world wide web.

→ A Hypertext contains graphics, videos, sounds, text and text within text and so on.

→ They are classified into 2 types:

1. static web page.
2. Dynamic web page

1. static web page:-

shows the same content for every execution and user is unable to interact with static web page.

→ To create a static web page we use HTML, CSS (cascading style sheet)

2. Dynamic web page:

It shows different content for any execution. To interact with a static web page we need to add dynamic nature to web page.

→ By using Dynamic webpages we can perform some calculations, validations (ex. login page)

→ HTML is a interpreted language (compile line by line)

HTML:

HTML stands for Hyper Text Mark up Language.

→ It is used for creating static Webpages

Hypertext: The content placed on a web page like text, animations, images, videos, tables, text within text means whenever user clicks on some text the control moves to the next webpage.

→ It is not a programming language. we cannot perform any operations.

Markup language: It is used to design web page in more attractive manner and it is used to convert textual information into image, table etc.

→ HTML is invented by TimBeinerslee in 1989 but first version of HTML is released in 1991 and the current version of HTML is HTML 5 released in 2014.

→ A webpage contains set of HTML elements

→ All the HTML elements can be represented using tags

→ A tag can be enclosed in angular brackets

Syntax: <tag-name> content </tag-name>

→ In HTML some tags not requires end tags those are called Singleton tags

Syntax: <tag-name>

Structure of HTML document

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

An <html> is a root tag for every HTML document

<head> is used to specify meta information about the document like title name, styles, scripts and linking to external files.

→ To add title name to a webpage use a tag

<title> webpage </title>

→ <body> is used to whatever the content including in <body> that will be displayed on web page.

Write a HTML program to display a simple message

```
<html>
  <head>
  <title> welcome </title>
  </head>
  <body>
    welcome to HTML prog
  </body>
</html>
```

HTML is a interpreted Language.

→ saved with extension first.html/run on any web browser

Common tags in HTML:-

1.  :- It is used to define a paragraph

Syntax:-  content 

if we add any text in  the browser will add some space before and after the paragraph automatically

eg:-

```
<p>
  To,
  vardhaman,
  kacharam,
  hyderabad,
</p>
```

o/p:-

To vardhaman kacharam hyderabad

→ At the time of displaying the browser will ignore all new lines, spaces based on window adjustments.

Attributes of body tag :-

Attributes are useful for to add some functionality to the specified html element

<body> tag contains 3 attributes.

1. bgcolor:

It is used to assign some background color.

Syntax: <body bgcolor = "red">

<body bgcolor = "#44aa99">

2. background: It is used to assign some image as background

Syntax: <body background = "file name"> (or)

<body background = "url">

3. text: It is used to modify or change the change text color.

Syntax: <body text = "red">

<body bgcolor = "green" text = "pink">

Attributes of  :

It contains only one attribute i.e allign.

allign:- It is used to specify allignment position of a text in a paragraph

Syntax: 

default allignment is left.

2. <br>: br means line break the control moves to next line.

→ It is singleton tag.

3. : It converts the given text into bold

Syntax:  content 

4. : It converts the given text into italic form

5. <u>: It add underline to given text

eg:-   <u> vardhaman </u>  

6. <hr>:- It is also a singleton tag it is used to draw a horizontal line

Heading tags:-

In HTML to represent heading 6 tags are available ranges from $h_1$ to $h_6$.

<h1> $h_1$ is the largest heading size

$h_6$ is the smallest heading size

<h1> html </h1>

<h2> html </h2>

Attributes of heading tag:

align: It is used to specify position:

<h1 align = "right"> html </h1>

<h2 align = "justify"> html </h2>

<h3 align = "center"> html </h3>

: It is used to display everything in the center of the webpage.

Syntax:  content 

<pre>: It is used to display pre-formatted text.

eg:-
```
<pre>
hi
hello
how
are
</pre>
```

o/p:-
```
hi
hello
how
are
```

In case of <pre> it preserves both blankspaces and line breaks but <p> will ignore.

<font>: It is used to change the font size, font style and color to a particular text.

<font size = "20" color = "#bb2244"> welcome </font>

Spa face: It is used to change the type of the font

<font face = "font name">

<sub>: It is used to define subscript

ex: $x_{2+y_2}$ → $x$ <sub> 2 </sub> + $y$ <sub> 2 </sub>

$x_2$ $y_2$

<sup> :- It is used to add superscript

eg:- $x^2+y^2$

x <sup> 2 </sup> + y <sup> 2 </sup>

o/p: $x^2+y^2$

<big> :- To convert text into bigger size:

(or)

display text in bigger size

eg:- hi <big> hello </big>

<small> :- It will display the text in smaller size.

eg:- hello <small> welcome </small> o/p: hello welcome

<mark> : It is used to highlight some text with some background color. the default background color is yellow.

eg:- hi <mark> html </mark> programming

 :- It is used for emphasize the given text means it converts into italic form

eg:- hi  rushitha 
↓
will be in italic form.

 :- It is used to display the text in bold form

eg:-  note  content.

Hyper link :-

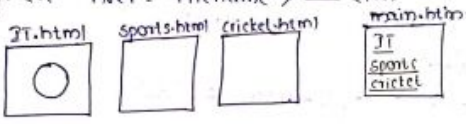
Hyper link is used to navigate from one web page to another webpage.

To create hyperlink use athor tag <a>. it contain two atributes

1) href 2) target → specify where to open the link doc tag

href :- It is a hyper link reference need to specify filename(or) url

eg:- <a href = "filename"> — </a>




```
<body>
<a href="IT.html"> IT </a><br>
<a href="sports.html"> sports </a><br>
<a href="cricket.html"> cricket </a><br>
</body>
```

In case of hyperlink all unvisited hyperlinks will be represented in blue

all active hyperlink in red

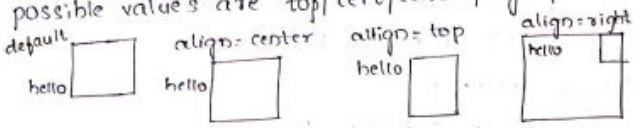
all visited hyperlink → purple

<img> : It used to place a image on the webpage. It is also a singleton tag.

Attributes of <img> :

1. src : It is used to specify the source file name.
2. width : pixels the
3. height : To specify the size of a particular image in pixels.
4. align :- It is used to specify the position based on the adjacent elements

The possible values are top/left/center/right/bottom




eg:- hello <img src="flower.jpg">

5. alt : It specify alternate text if the image is not available.
6. hspace : It is used to add the space horizontally in b/w image and text
7. Vspace :- It is used to add the space vertically in b/w image and txt.

write a html program to add image on a webpage

```
<html>
<head>
<title> Image </title>
</head>
<body>
<body background = "flower.jpg">
<img src = "f1.jpg" width="50" height = "50">
</body>
</html>
```

~~tare~~

<q>: It is used to add quotation marks to a specific text.

eg:- <q> html </q>

"html"

<del> :- It is used to delete a particular text (or) Strike off particular text

Syntax:- hi <del> hello </del> it

hi ~~hello~~ it

<ins>: It is used to insert a text

eg:- I like <del> red </del> <ins> blue </ins>.

o/p: I like ~~red~~ blue

Lists in HTML:

A list is a set of elements

Lists are classified into 3 types, they are

1. unordered list
2. Ordered list
3. Description list

1. Unordered list:

It contains a set of elements not placed in a specific order.

To create unordered list use a tag <ul>

to add set of elements use <li>

eg:-
```
<ul>
    <li> it </li>
    <li> cse </li>
    <li> ece </li>
</ul>
```

o/p:

- it
- cse
- ece

default style is disc

we have 4 styles • ○ ■ ↓

disc circle square none

→ The default bullet style for unordered list is disc.

To change bullet style use an attribute type

eg:- <ul type = "disc|square|circle|none">

Write a HTML program to display the given unordered

List

fruits → $h_1$

- mango
- orange
- banana

flowers → $h_2$

- rose
- lotus
- jasmine

```
<html>
<head>
<title> List </title>
</head>
<body>
<h1> fruits </h1>
<ul type = "square">
    <li> mango </li>
    <li> orange </li>
    <li> banana </li>
</ul>
<h2> flowers </h2>
<ul type = "circle">
    <li> rose </li>
    <li> lotus </li>
    <li> jasmine </li>
</ul>
</body>
```

Ordereder list:

It contains set of elements represented in sequential ordered

To create ordered list use a tag <ol>

To add set of elements use a tag <li>

eg:-
```
<ol>
  <li>red</li>
  <li>green</li>
  <li>blue</li>
</ol>
```
o/p:
1. red
2. green
3. blue

attributes of <ol>:

1. type:

It is used to specify the sequence type

eg:- <ol type = "1/a/A/I/i"> → lowercase roman letters.
(1 → number, a → lowercase, A → uppercase, I/i → uppercase roman)

2. start: always start value is integer value

→ It is an integer value specify the starting position of a particular sequence.

eg:- <ol type = "1" start = "56"> 56 57 58

<ol type = "A" start = "6"> F G H

write a HTML program to create ordered list

```
<body>
<h1> branches </h1>
<ol type = 'i'>
  <li> it </li>
  <li> cse </li>
  <li> ece </li>
</ol>
<h2> cricket players </h2>
```

Braches
i.
ii.
iii.

Cricket players
10.
11.
12.

```
<ol type = "1" start = "10">
  <li> dhoni </li>
  <li> sachin </li>
  <li> yuvraj </li>
</ol>
</body>
```

Nested List:-

A list within the list

Write a HTML program to create the given Nested list

1. Item1
   - Sub1
   - Sub2
   - Sub3
2. Item2
   i. Sub4
   ii. Sub5
   iii. Sub6
3. Item3
   - Sub7
   - Sub8

```
<body>
<ol type = "1">
  <li> Item1 </li>
  <ul>
    <li> Sub1 </li>
    <li> Sub2 </li>
    <li> Sub3 </li>
  </ul>
  <li> Item2 </li>
  <ol type = "i">
    <li> Sub4 </li>
    <li> Sub5 </li>
    <li> Sub6 </li>
  </ol>
  <li> Item3 </li>
  <ul type = "square">
    <li> Sub7 </li>
    <li> Sub8 </li>
  </ul>
</ol>
</body>
```

Description list:-

It is used to provide description about various notations (or) terms.

<dl> is used to define description list. it contains two subtags

1. <dt> 2. <dd>

<dt> : It is used to represent description term

<dd> : It is used to provide description about term.

Write a html program to create discription list

```
<body>
<dl>
  <dt>HTML</dt>
  <dd>Hypertext markup language</dd>
  <dt>HTTP</dt>
  <dd>Hypertext transfer protocol</dd>
  <dt>URL</dt>
  <dd>Uniform resource locator</dd>
  <dt>WWW</dt>
  <dd>world wide web</dd>
</dl>
</body>
```

HTML _____

HTTP _____

URL _____

WWW _____

Tables in HTML:

To create a table on the webpage use a <table> tag in HTML

Attributes of <table> tag:

1) bgcolor : It is used to add background color to the particular table.

eg:- <table bgcolor = "blue">

<table bgcolor = "#1.2233bb">

ii) border : It is used to specify the width of a borders the default border size is 1

eg:- <table border = "0">

o/p: the table will displayed without any borders

iii) align :- It is used to specify the position of a table on the webpage

eg:- <table align = "left/right/center">

by default deft. align

iv) cell padding :- It is used to specify the no. of spaces between cell borders and content.

eg:- <table cellpadding = "pixels">

v) cell spacing :- It is used to specify the no. of spaces between adjacent cells.

eg :- <table cell spacing = "pixels">

vi) width :- It is used to increase (or) change the size of a table.

eg :- <table width = "%">

<tr> :-

It is used to add a new row in the table.

<th> :-

It is used to define (or) add column heading displayed in bold form and center alignment

<td> :- To add table contents and default alignment is left

<caption> :- It is used to specify (or) add caption to a particular table.

Write a HTML program to create table

```
<html>
<body><head>
<title> Table </title>
</head>
<body>
<table bgcolor="yellow">
<caption> Books </caption>
<tr>
    <th> SNo </th>
    <th> book name </th>
    <th> author </th>
    <th> price </th>
</tr>
<tr>
    <td> 1 </td>
    <td> C prog </td>
    <td> Balguru swamy </td>
    <td> 500 </td>
</tr>
<tr>
    <td> 2 </td>
    <td> java </td>
    <td> jones </td>
    <td> 450 </td>
</tr>
<tr>
    <td> 3 </td>
    <td> WT </td>
    <td> Black book </td>
    <td> 600 </td>
</tr>
</table>
```

Books

| SNo | book name | author | price |
|---|---|---|---|
| 1. | C lang | Balguru Swamy | 500 |
| 2. | java | jones | 400 |
| 3. | WT | Black book | 600 |

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

Attributes of <tr> tag:

i) bgcolor :- It is used to assign background color to a particular row.

Syntax: <tr bgcolor = "#aabbcc">

ii) align :- It is used to align the content in a particular row.

Syntax: <tr align="left|right|center|justify">

default align is left

iii) valign :-

It is used to align cell content vertically

Syntax :- <tr valign="top|bottom|center">

By default valign is center

Attributes of <th> tag and <td> tag:

i) bgcolor ;

ii) align

iii) valign

iv) row span :- It is used to merge speficied no. of rows into single cell.

Syntax : <th rowspan="number">

v) colspan :- It is used to merge specified no. of columns into single cell.

Syntax: <th colspan = "number">

→ In case of <td> the default alignment is left and displayed in normal form.

width :- It is used to increase width of a particular cell.

Syntax: <th width="%">

height :- It is used to increase height of a particular cell

Syntax: <th height="%">

write a HTML program to display the following table

| SNo | Name | Des | Sal |
|---|---|---|---|
| 1 | abc | manager | 80,000 |
| 2 | xyz | prof | 50,000 |
| | por | Asstprof | 1,000 |

```
<html>
<head>
<title> Table </title>
</head>
<body>
<table>
<tr>
<th>SNo</th>
<th>name</th>
<th>Designation</th>
<th>salary</th>
</tr>
<tr>
<td>1</td>
<td>abc</td>
<td colspan="2">manager 80,000</td>
</tr>
<tr>
<td rowspan="2">2</td>
<td>xyz</td>
<td>professor</td>
<td>50,000</td>
</tr>
<tr>
<td>por</td>
<td>Assistant professor</td>
<td>1,00,000</td>
</tr>
</table>
</body>
</html>
```

Forms in HTML:

A form contains a set of elements is used for taking some data from the user.

To create a form use a tag <form>

we can place various form elements like textfield, labels list, checkbox, buttons soon.

1. Creating Textfields: (→ singleton)

→ To create any Form element use <input> tag.

→ In the <input> tag type attributes specifies type of form element

→ To create a textfield use <input type="text">

→ Attributes of <input> tag

1. name: it is used to provide some name to a particular textfield
2. value: It is used to provide some default text
3. size: no. of characters to be shown.

2. Button:

To create buttons on web page use the following forms

<input type="submit" value="message">

It is used to create submit button whenever user clicks on submit button it moves to next page.

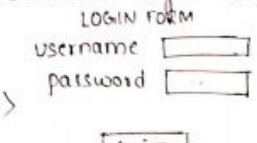
<input type="reset" value="msg">

It is used to create a reset button. It removes the content on page

To create normal clickable button use

<input type="button" value="msg">

write a HTML program to create login form.




```
<body>
<h <center>
    <h1>login form</h1>
<form>
    username : <input type="text" name="t1"><br>
    password : <input type="password" name="t2"><br>
    <input type="submit" value="login">
</form>
</center>
</body>
```

3. checkbox :-

To create a checkbox use the following form

<input type="checkbox" name="msg" value=" "> content

☐ it

The value will be written at the time of selection

eg:- <input type="checkbox" name="c1" value="1">It<br>
<input type="checkbox" name="c2">blue <br>
<input type="checkbox" name="c3"> red<br>

o/p ☐ blue
☐ red

To change the status of a checkbox in checked manner by default

<input type="checkbox" name=" " checked>

o/p: ☑

Radio Buttons:

To create the radiobutton use

eg:- <input type="radio" name="name"> O

eg:- <input type="radio" name="gender"> Male

o/p :- O male

→ To select only one radio button provide same name for all radio buttons

eg:- <input type="radio" name="gender">male <br>
<input type="radio" name="gender">female

o/p: O male
O female

in this we can select only one

→ To provide default status use checked attribute

Syntax: <input type="radio" name="a" checked>

password Field:

→ To create a password text field give

syntax: <input type="password" name="t1">

default is •

place holder:

It is used to provides some default text to understand (or) recognise by the user.

Combobox :-

To create a combobox use a tag <select>

→ To add elements into combobox use a tag <option>

eg:- For maintaining 4 branches

```
<select>
    <option>It</option>
    <option>cse</option>
    <option>eee</option>
</select><option>ece</option>
```

→ The user can select only one item from combobox

→ To select multiple items use multiple attribute in <select> tag

eg:- <select multiple>
<option> it </option>
</select>

→ To display specified no. of elements as visible in a particular list use size attribute

eg:- <select multiple size="4">
4 elements are visible to the user.

**Text area :-**

To provide multiple lines of text use <textarea> tag.

Syntax:- <textarea rows=" " cols=" ">

rows ⇒ how many rows are visible (no. of lines are visible)

cols ⇒ no. of characters are visible in a every row

**Comment tag :**

To provide any ~~tags~~ comments in HTML document use

<!-- content -->

White a HTML program to create registration form. for a website.

```
<html>
<head>
<title> Form </title>
</head>
<body>
<form> <center>
<h3> registration Form </h3>
</center>
```

Registration form

Name:

father's name:

Gender: o male o female

DoB: D M Y

Email id:

Phone no:

```
<form>
name: <input type="text" name="t1"> <br>
father's name: <input type="text" name="t2"> <br>
Gender: <input type="radio" name="gender">
male female <br>
DOB : <select size="4">
<option> 1 </option>
<option> 2 </option>
<option> 3 </option>
<option> 4 </option>
</select>
<select size="4">
<option> jan </option>
<option> feb </option>
<option> mar </option>
<option> Apr </option>
</select>
<select> size="4">
<select -si
<option> 1998 </option>
<option> 1999 </option>
<option> 2000 </option>
<option> 2001 </option>
</select> <br>
email id : <input type="text" name="t3"> <br>
phone no : <input type="text" name="t4"> <br>
Hobbies : <input type="checkbox" name="c1" > reading <br>
<input type="checkbox" name="c2"> dancing <br>
<input type="checkbox" name="c3"> singing <br>
```

Hobbies:

Address:

Register clear

<input type="checkbox" name="c4">

Address: <text area>

## Frames in HTML:

Frames are used for placing multiple webpages in the single web browser.

→ To divide a webpage into multiple sessions use <frameset> tag.

it contains two attributes

1. rows 2. cols

rows: it is used to divide webpage into multiple rows.

eg: <frameset rows="50%,50%">

cols: it is used to divide webpage into multiple columns.

eg: <frameset cols="25%,25%,25%,25%">

frameborder: It is used to display frame borders the default width of frame border is 1. If frameborder=0 there is no frame borders.

→ To assign a webpage to a particular session use <frame> tag

It contains two attributes

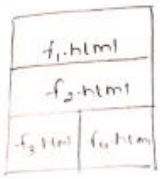
1. src 2. name



src: source file name

name: name assign to a frame.

eg: create a frameset for the given format




```
<head>
 <frameset rows="33%,33%,*">
  <frame src="f1.html" name="f1">
  <frame src="f2.html" name="f2">
  <frameset cols="50%,50%">
   <frame src="f3.html" name="f3">
   <frame src="f4.html" name="f4">
  </frameset>
 </frameset>
</head>
```

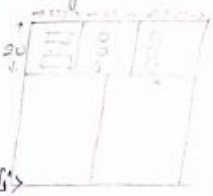
eg: Create frameset for the given format

```
<head>
 <frameset rows="20%,80%">
  <frame src="f1.html" name="f1">
  <frameset cols="15%,85%">
   <frame src="f2.html" name="f2">
   <frameset rows="26%,26%,*">
    <frame src="f3.html" name="f3">
    <frame src="f4.html" name="f4">
    <frame src="f4.html" name="f4">
   </frameset>
  </frameset>
 </frameset>
</head>
```

Create a html document that has 5 frames there must be two rows of frames the first with 3 frames and the other with 2 frames all the frames in the first row have equal width of display each frame contains separate html document the 1st frame contains 2 textfields for representing name and address of 30 characters display and 2nd frame contains 3 radiobuttons represents various colors and 3rd frame contains 4 checkboxes for representing car accessories. bottom 2 frames contains 2 images as background. and the first row contains 20% of total height

Ma

```
<head>
<frameset rows="20%,80%">
<frameset cols="33%,33%,*">
<frame src="f1.html" name="f1">
<frame src="f2.html" name="f2">
<frame src="f3.html" name="f3">
</frameset>
<frameset cols="50%,50%">
<frame src="f4.html" name="f4">
<frame src="f4.html" name="f5">
</frameset>
</head>
```




f1.html

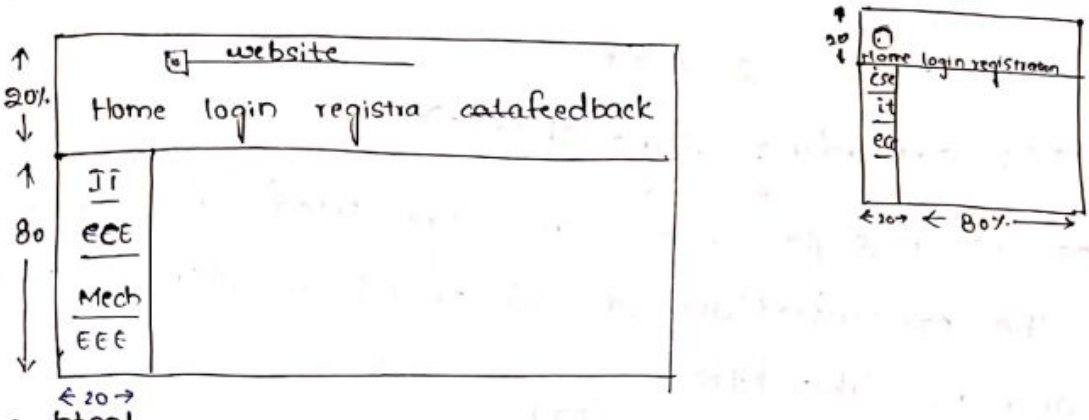
```
<html>
<head>
<title> f1 </title>
</head>
<body>
<form>
name:<input type="text" size="30"><br>
address:<input type="text" size="30"><br>
</form>
</body>
</html>
```

f2.html

```
<html>
<head>
<title> f2 </title>
</head>
<body>
<
```

create an html document for representing online book store website home page.

i) A web page divided into 3 frames top, left, right




Home.html

```
<html>
<head>
<frameset rows="20%,80%">
  <frame src="top.html" name="f1">
<frameset cols="20%,80%">
  <frame src="left.html" name="f2">
  <frame src="right.html" name="f3">
</frameset>
</frameset>
</head>
</html>
```

left.html

```
<html>
<head>
<title> Hyperlink </title>
</head>
<body>
<a href="it.html" target="f3">it </a> <br>
<a href="ece.html" target="f3"> ece </a><br>
<a href="Mech.html" target="f3"> Mech </a><br>
```

<a href="eee.html" target="_blank">

</body>

</html>

## 2. CSS

CSS: (Cascading Style Sheets).

→ CSS is a style sheet language used for describing the presentation of document written in markup language like html

(or)

→ It is used to add styles (colors, borders, alignments) to a webpage.

(or)

→ It is used to design a web pages in more attractive manner by adding some styles.

→ To provide some styles to various html elements we need to design style rules.

Syntax:-

→ To design a style rule use the following syntax.

Syntax:

selector { declaration }

Selector: An html element (or) HTML tag

→ declaration contains two parts property: value;

→ property defines color, fontsize, text-align, background color...

→ property: It specifies the type of property style

eg:- h1 { color: green; }

p { text-align: center; color: blue; }

Types of selectors:

Selectors are classified into 5 types

1. normal selector
2. group selector
3. universal selector
4. Id selector
5. class selector.

Normal selector (or) element selector

→ It is used to define styles for single HTML element

eg:- h6 { color: pink; }

ul { color: red; }

p { text-align: center; color: red; }

Group selector

→ It is used to define (or) assign same styles for multiple HTML tags, use group selector

eg:-

```
h1 { color: red; }
p { color: red; }
pre { color: red; }
```

⟹

```
h1, p, pre
{ color: red; }
```

Universal selector

→ It is used to assign styles for all html elements

eg:- * { ____ }

* { text-align: center; color: orange; }

Id selector:

→ It is used for defining set of style rules those are applicable for any html elements those contains id attribute.

→ To declare Id selector use the following syntax

Syntax: #id-name
{
Style rules
}

eg: #it
{
color: red;
fontsize: 50;
}

eg:- <h1> hello </h1> → The style is not applied.
Html  } Style is applied
 html </h1>
hello  → Style is not applied.

Class selector:
→ It is similar to ID selector it is also applicable for more than one html element
→ To declare class selector use the syntax

Syntax: .classname
{
rules
}

eg: .it
{
color: yellow;
}
.cse
{
color: blue;
}
<h1> hello </h1> → Style is not applied
<h1 class="it"> hello </h1> } Style (colon) is applied
<h1 class="cse"> hello </h1>

Types of style sheets:
Style sheets are classified into 3 types
1. Inline style sheet
2. Internal style sheet
3. External style sheet

1. Inline style sheets:
It is used for assigning a style to a single HTML element.
→ To assign styles use an attribute style
Syntax: <tag-name style="property: value; property: value...">
_____ </tag-name>

eg:- <h1 style="color: pink; fontSize: 10"> Hello </h1>

Write a HTML program to demonstrate Inline style sheet.

```
<html>
<head>
<title> Inline </title>
</head>
<body>
<h1> welcome </h1>
<h1 style="color: green; text-align: center;"> welcome </h1>
<p style="color: pink; font size: 50;">
hello
vardhaman
</p>
</body>
</html>
```

Internal style sheet:

→ It is used to apply a styles for all HTML elements present in a single webpage.

→ To create internal style sheet use <style> tag in the head session

<style> contains various selectors to define the rule.

Syntax:
```
<head>
  <style>

  </style>
</head>
```

Write a HTML program to demonstrate internal style sheet.

```
<html>
<head>
<style>
h1 color</h1 h1
        {
        color: green;
        }
    h3
    {
    color: #9999333;
    }
    p
    {
    color font size: 15;
    }
</style>
</head>
<body>
<h1>hello</h1>
<h1> welcome </h1>
<h3> vardhaman</h3>
<h2> it </cse>
<p>
hello
hi
vardhaman
</p>
<p>
internal style sheet </p>
</body>
</html>
```

Write a HTML program to define styles using ID and class selector.

```
<html>
<head>
<style>
<h1 #it
    {
    color: red;
    }
    #.cse
    {
    fontsize: 20;
    }
</style>
</head>
<body>
<h1 id="it"> hello </h1>
<p class="cse">
vardhaman it
cs branch </p>
</body>
</html>
```

External style sheet (CSS):

→ It is a separate CSS document contains a set of rules.

→ To apply styles to a particular HTML document we need to link CSS page using <link> tag

<link> tag :- It is used to link an external document

<link> tag contains 3 attributes

rel :- it specifies the relation b/w html document and linking document.

rel = "stylesheet"

type : it specifies type of linking document to link CSS document we need to give type = "text/css"

href : it specifies the URL of linking document.

Syntax:

<link rel="stylesheet" type="text/css" href = "file_name">

eg:- Write a HTML program to demonstrate external Style sheet

ext.css document

```
h1
{
color : green;
font size : 30;
}
p{
color : green;
font size : 20;
}
```

⇒

```
<html>
<head>
<title> Style sheet </title>
<link rel="Stylesheets" type="text/css" href="ext.css">
</head>
<body>
hello
<h1> welcome </h1>
<p> webprogramming </p>
</body>
</html>
```

CSS properties:

i) Background properties

i) Background-color : It is used to assign for a particular HTML element.

Syntax: background-color : red ; (hexa value :)

ii) eg background-image : It is used to assign image as background for a particular HTML element.

syntax: background-image : url ('file-name'); (url also)

iii) background-repeat : It is used to control the repitition of background image.

no-repeat : it will display image for once (top left)

repeat-x : it will repeat horizontally

repeat-y : it will repeat vertically.

iv) background-position : It is used to change the position of a background image

Syntax: background-position : top/right/center/bottom/left ;

default position is top left

eg:- background-position: center ;

background-position: bottom right ;

Write a HTML program to demonstrate background properties.

```
<html>
<head>
<title> Style sheet </title>
<style>
body
{
background-image: url('xml.jpg');
background-repeat: no-repeat;
background-position: center;
}
</style>
</head>
<body>
<h2 style="color:red;">IT</h2>
<h1> HTML </h1>
<h2> CSS </h2>
<h1 style ="color:green;font-size:40;">web programming</h1>
<p> web design </p>
</body>
</html>
```

2. Text properties:

i) color: It is used to change the foreground text color.

ii) text-align: It is used to specify the position.

Syntax: text-align: center/left/Right/justify;

iii) text-transform: It is used to convert the text into specified format.

we have 4 options none/lowercase/uppercase/capitalize

iv) text-decoration: it is used to add decoration to a particular text.

It is also used for removing line from hyperlinks

we have 4 options none/underline/overline/linethrough

Syntax: text-decoration: none/underline/overline/line-through

write a HTML program to add text properties.

```
<html>
<head>
<style>
<!p
{
color: green:
text-transform: uppercase;
text-decoration: line-through;
} text-align: center;
}
a
{
text-decoration: overline;
}
</style>
</head>
<body>
hello
<p>hi how are you </p>
<a href ='hello.html'>Ii </a>
</body>
</html>
```

Border properties:

→ It is used to add borders to a particular HTML elements.

1. border-style: It is used to assign a border style options are none | dotted | dashed | solid | double | groove | ridge | inside | outside.

Syntax: border-style: dotted;

o/p: for all borders the same style will assigned

2) border-style: dotted double;

o/p: top and bottom are with dotted double is applicable for right and left

3) border-style: dotted solid dashed

for top, Right & left, bottom

4) border-style: dotted solid dashed none

top, Right, bottom, left

2. border-color: It is used to assign color to a border

3. border-width: It is used to change the width to a border

Syntax: border-width: length;

pixels (px) | points (pt)

Write a program to demonstrate border properties

```
<html>
<head>
<style>
 p
 {
   border-style : dotted ; dashed ;
   border-color : green ; red;
   border-width: 10px;
   padding : 25px 50px.
 }
 h1
 {
   border-style : dashed ;
   border-color : red ;
 }
</style>
</head>
<body>
 <p> hi hello vardhaman </p>
 <h1> welcome </h1>
</body>
</html>
```

o/p:

hi hello vardhaman

Welcome

4. Padding: properties:

It is used to add space b/w borders and content.

We have 4 properties

i) padding-top:

ii) padding-right

iii) padding-bottom

iv) padding-left

By using padding property we can set for all borders

Syntax: padding: length; → px (or) pt

eg: padding: 10px; → for all borders.

5. Width and height: It is used to increase the width and height of a particular element

Syntax: width: length%.

height: length%.

eg:- width : 25%
height : 50%

6: Margins property:

→ It is used to add space between window borders and HTML element.

i) Syntax: Margin: length;
(px) (or) (pt)

we can assign separately as following

i) margin-top
ii) margin-right
iii) margin-bottom
iv) margin-left

7. Cursor:

→ It is used to change the cursor style

Syntax: cursor: help | pointer | wait | crosshair | zoom-in | zoom-out

```
program:
<html>
<head>
<style>
.xlink
{
 cursor: crosshair;
}
.hlink
 {
  cursor: help;
 }
.wlink
 {
  cursor: wait;
 }
.zlink
 {
  cursor: zoom-in;
 }
</style>
</head>
<body>
<b class="zlink">This is normal | bold </b> <br><br><br>
<a href="css1.html"
   class="xlink">CROSSLINK </a><br><br><br>
<a href="css1.html" class="hlink">HELPLINK </a>
  <h1 class="wlink">welcome </h1>
</body>
</html>.
```

Font properties:

→ It is used to assign (or) apply some styles related to fonts.

i) Font-size: it is used to change the font-size

ii) font-style: it is used to assign the font style

for Syntax: font-style: normal | italic | oblique

iii) font-family: it is used to specify the font name.

Syntax: font-family: verdana | Roman...

iv) font-weight: To increase the boldness (or) weight of the font

Syntax: font-weight: normal | bold | bolder | lighter;

```
program:
<html>
<head>
<style>
 p
 {
  color: green;
  font-size: 50;
  font-style: oblique;
```

```
font-weight : bolder;
font-family: verdana;
}
</style>
</head>
<body>
 hello
<p>
 hi how are you </p>
</body>
</html>
```

Related to hyperlinks:

→It is used to assign some colors based on the state of a hyperlink

it has 4 state.

i) a:link → Incase of unvisited hyperlink.

ii) a:visited → visited hyperlink

iii) a:active → about to visit

iv) a:hover → whenever mouse cursor moves to hyperlink.

```
eg:- a:link
     {
      color: red;
      background-color : green;
     }
```

prog Write an HTML document contains 3 hyperlink. if those are unvisited assign background color hyperlink color to red. once they are visited change the background color to pink and text color to green.

```
<html>
<head>
<style>
 a:link
 {
  background-color: yellow;
       color: red;
 }
 a:visited
 {
  background-color: pink;
       color: green;
 }
</style>
</head>
<body>
 <a href="IT.html"> hello </a>
 <a href="cse.html"> cse </a>
 <a href="ECE.html"> ECE </a>
</body>
</html>
```

float:

→It is used for positioning (or) placing the content adjacent to the particular HTML element

syntax: float: left/right/none;

Write a HTML program to apply some styles for displaying lists

```
<html>
<head>
<style>
 ol
 { border-style: dotted;
```

```
background-color: pink;
    padding: 25;
}
ol li
{
  background-color: green;
}
ul
{
  border-style: solid;
  background-color: dark blue;
}
ul li
{
  border-style: solid;
  background-color: light blue;
}
</style>
</head>
<body>
 <ol>
   <li> coffee </li>
   <li> tea </li>
   <li> coco cola </li>
 </ol>
 <ul>
   <li> coffee </li>
   <li> tea </li>
   <li> coco cola </li>
 </ul>
</body>
</html>
```

<Marquee> tag is used for scrolling a text (or) image on the web page.

Syntax: <marquee> hello </marquee>
text/image

eg:- <marquee> welcome </marquee>

Attributes of <marquee> tag:

1. direction:
   It is used to specify direction of the scrolling text.
   The possible options are left/right/up/down

2. behavior: It is used to specify the type of scroll.
   possible options: scroll | slide | alternate
   ↓ default

3. scrollamount: It is used to increase the speed of scrolling text

4. bgcolor : it is used to assign some background color for scrollable area

5. width & height : it is used to increase (or) decrease width and height of scrollable area

6. loop : by default the text will be scrolled for infinite times to reduce the number of iterations use loop attributes.

* Program

```
<html>
 <head>
 </head>
 <body>
  <marquee direction="up", behaviour="alternate",
      bgcolor="green" loop="5"> welcome to
  html programming
```

</body>
</html>

## position & z-index properties:

position: It specifies the type of positioning method used for an HTML element

It contains 4 values

1. static
2. relative
3. absolute
4. fixed

→ while giving position property we need to assign top, left, right, bottom properties for a particular HTML element

1. static:
   it is the default positioning method/element
→ static elements are positioned according to normal flow of page.
→ In case of static position it will ignore the top, left, right, bottom properties.

2. relative: It is used to place an element based on the previous elements position.

3. absolute: It will display on the exact position

4. fixed: It means it always stay on the same position at the time of scrolling web page.

Write a HTML program to work with position attributes

```
<html>
<head>
<style>
h1
{ top:20px;
  left:40px;
  position:static;
  border-color:red;
  width:40%;
}
h2
{ top:20px;
  left:40px;
  position:static;
  border-color:green;
  width:40%;
}
</style>
</head>
<body>
<h1> Welcome to vardhaman </h1>
<h2> Hello </h2>
</body>
</html>
```

z-index:- It specifies stack order of a particular HTML element

→ Always higher z-index value will be available on top.

program:

```
<html>
<head>
<style>
h1
{ top:20px;
  left:40px;
  position:absolute;
  border-color:red;
  width:40%;
  z-index:3;
}
```

```
h2
{
  top:20px;
  left:40px;
  position:absolute;
  border-color:green;
  width:40%
  z-index:1;
}
</style>
</head>
<body>
<h1> welcome to HTML </h1>
<h2> style sheets </h2>
</body>
</html>
```

 tag:

It is used to divide a web page into a divisions (or) sessions.

→ In the different sections we can place different content like images, text etc.

Syntax: 
any content

it contains only one attribute align

align: It is used to specify the alignment position of a content in the division.

* Java Script

Java script is mainly used to add dynamic nature to the static webpages

(or)

It is used to increase the interactions of a user with webpages.

→ java script is a client side scripting Language. Usually, it will be integrated and communicating with HTML. To process java script code we require any web browser.

→ Initial name of java script is 'Live script'

→ java script is a object based language. it follows the syntax, constructs of C language.

→ A java script code must be included in <script> tag

Syntax: <script>
//code (or)
</script>

<script type="text/javascript">
//code
</script>

Ex: The <script> tag must be included in the head section.

To display a message on the webpage, use document.write

Syntax: document.write("msg");

eg: document.write("hello");
document.write("<h1>hello</h1>");
document.write("hello <br> welcome");

Write a java script program to display a message on webpage

```
<html>
<head>
<script type="text/javascript">
document.write("welcome to javascript <br>");
document.write("hello");
</script>
</body></head>
</html>
```

External javascript file:

Create a separate javascript file and saved with the extension .js

→ To link a .js file to a html document use <script> tag.

Syntax: `<script type="text/javascript" src="url">`
`</script>`

eg:- ext.js

```
document.write("hello");
document.write("welcome to js");
```

HTML program

```
<html>
<head>
<script type="text/javascript" src="ext.js">
</script>
</head>
</html>
```

Comments in java script :-

A single line comments are represented as

//

Multi lines comments are represented as

/* - - - - - */

A javascript is a interpreted language.

→ javascript is a loosely typed language. means there are no keywords are available to represent the datatypes.

→ In java script to declare a variable use a keyword "var"

Syntax: var variable-name;

eg: var a = 5; (it will store integer value)
var b = it; (it will acts as string)
var c = true;
var d = 1.56;
var d = new Date();

Rule:-

→ In java script every variable starts with a letter (or) underscode (or) dollar.

→ It is a case-sensitive language.

→ we cannot use special words as variables.

Write a java script program to perform addition.

```
<html>
<head>
<script type="text/javascript">
var a=5;
var b=6;
var c=a+b;
document.write("c is"+c);
document.write("type "+typeof(c));
document.write(a**b);
</script>
</head>
</html>
```

→ operators

** : x**y
↓
$x^y$

typeof( ) : It will return which type of data stored in a variable.

Functions in java script :-

To define a function use the following syntax.

```
function func-name(args-list)
{
  //statement
}
```

1) function display( )
```
{
 document.write("hello");
}
```

2) function with parameters
```
function sum(n1, n2)
{
 var c = n1+n2;
 document.write(c);
}
```

3) function with return value
```
function cube(n)
{
 return n*n*3;
}
```

~~4)~~ function ~~no~~

Write a javascript program to demonstrate functions
```
<html>
<head>
<script type="text/javascript">
 function square(n)
 {
  ~~return~~ var c = n*n;
  document.write("c is"+c);
 }
 function cube(n)
 {
  return n*n*3;
 }
 document.write("hello"<br>");
  square(6);
var d = cube(3);
 document.write("<br> d is"+d);
</script>
</head>
</html>
```

Write a java script program to display square and cube of a 9 specified integers ranges from 1 to 5 and output for will be in tabular form.

| Num | square | cube |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

```
<html>
<head>
<script type="text/javascript">
document.write("<table border='1'>");
document.write("<tr><th>number</th>
        <th>square</th> <th>cube</th> </tr>");
 var i;
 for(i=1;i<=5;i++)
 {
document.write("<tr><td>"+i+"</td> <td>"+i*i+"</td> <td>
        "+i*i*3+"</td> </tr>");
 }
document.write("</table>");
</script>
</head>
</html>
```

Objects in javascript:

A javascript is a object based language it contains some built in objects

Some of the built-in objects are

1) window
2) Date
3) String
4) Number
5) Math
6) Array

Window: It is an object it refers current working window

→ it has give built-in functions

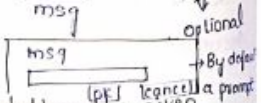
1) prompt()
2) alert()
3) confirm()
4) open()
5) close()

prompt(): It is used to take input from the user.

Syntax: Window.prompt("enter a value", "default value");

(or)

prompt("msg");




optional

By default a prompt box will return data in string form. To perform some operations we need to convert into equivalent form

parseInt()
parseFloat()

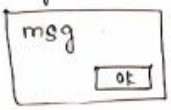
Write a js program to check whether the given number is even (or) odd.

```
<html>
<head>
<script type="text/javascript">
var n = Window.prompt("enter a value");
if (n%2==0)
{
 alert("even number");
}
else
{
 alert("odd number");
}
</script>
</head>
</html>
```

alert(): It is used to display alert messages (or) warning messages

Syntax: alert("msg");

Window.alert("msg");

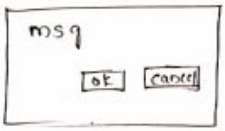



confirm(): It is used to display confirm window on web page.

Syntax: confirm("msg");

(or)

Window.confirm("msg");




program:

```
<html>
<head>
<script type="text/javascript">
var n= confirm("Doyou want to continue");
 if(n==true)
 {
  document.write("welcome");
 }
 else
 {
  document.write("bye");
 }
</script>
</head>
</html>
```

Open():- It is used to open a new window.

Syntax: window.open("url");

(or)

open("url");

eg:- window.open("login.html");

close():- It is used to close the current working window.

Syntax:- window.close();
close();

eg:- var x = confirm("Do you want to open new window");
if(x==true)
{
window.open("f1.html");
}
else
{
window.close();
}

Math: It is used to perform some mathematical operations.

1. sqrt()
2. sin()
3. cos()
4. tan()
5. log()
6. abs()
7. pow(x,y) = $x^y$
8. round() → round(4.7) = 5
(4.4) ⇒ 4
9. ceil() → ceil(4.5) = 5
10. floor() → floor(4.5) = 4

→ min(): it returns the min element from the set of elements

max(): it returns the max element from the set of elements

random(): it generates random value with between 0 to 1

String:- A string is a collection of characters. In javascript a string can be enclosed in either ' ' (or) " "

eg:- var x = "hello";

To find the length of the string use a property length.

eg:- x.length - 5.

Methods:

1. concat(): It is used to join multiple string.

eg:- var x = "hello";
var y = "it";
x.concat(y);

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| h | e | l | l | o |
| -5 | -4 | -3 | -2 | -1 |

o/p: hellohit

x.concat(y,"hi","hello");

o/p: hellohit hihello.

2) indexOf(): It returns first occurence index position of a particular character (or) a particular string.

Syntax:- indexOf('string'); (or) indexOf("string",start);

eg:- var x = "hi hello how are you hello";
var y = x.indexOf("hello") ⇒ o/p: 4
var y = x.indexOf("hello",10) ⇒

3) LastIndexOf(): It returns last occurence index position of a particular character (or) a particular string.

4. charAt():- It returns a character at a particular index position.

Syntax:- charAt("index");

eg:- var x = "hello";
var y = x.charAt("3");
var x = x.charAt(-2);

5) slice(): It is used to extract a set of characters from the specified position.

Syntax: slice(start,[end]);
↓ optional
start to end-1

(or)

slice(start)
↳ start to end.

eg:- var x = "hi hello how are you";

var y = x.slice(2,9); - hello-b

var y = x.slice(5); llo how are you

var y = x.slice(-12,-3); o - how are -

var y = x.slice(-8); w are you

substr(): It is used to extract substring from given string

syntax: substr(start [length]); (optional)

eg:- var x = "vardhaman college";

var y = x.substr(2,4) o/p: rdha

var z = x.substr(7); o/p: an-college

var z = x.substr(-8,-5) n-col

toLowerCase():- It is used to convert the given string into lower case

toUpperCase():- it is used to convert the given string into upper case

split(): it is used to divide the given string into set of strings based on some special character

eg:- var x = "welcome to js";

var y = x.split(" "); o/p: welcome to js based on spaces it can't be split

① var y = "Apple, banana, mango, orange";

var z = y.split(",",3);

it can be split upto only 3 values.

Date:- It is used to perform some operations on date and time.

To create a date object use a following syntax

Syntax: var d = new Date();

document.write(d);

↳ it contains current system date and time

var d = new Date(year, month, date, hours, minutes, seconds, milliseconds)

0-11 1-31 0-23 0-59 0-59 0-999

var d = new Date(milliseconds)

↳ from jan 1st 1970 12 am (10101234)

var d = new Date("String")

Methods:-

1. getFullYear(): It returns the year in 4 digits format

Syntax: var d = new Date();

document.write(d.getFullYear());

2. getMonth(): It returns the month (0-11)

3. getDate(): It returns the date (1 to 31)

4. getDay(): It returns week day from particular doeek (0-6)

5. getHours(): It returns the time in hours (0-23)

6. getMinutes(): It returns the time in minutes (0-59)

7. getMilliseconds(): (0-999)

8. getSeconds()

9. getTime(): It returns the time in milliseconds.

10. SetFullYear():

document.write(d.SetFullYear(2020));

document.write(d.setMonth(11));

document.write(d.setMinutes(30));

Write a js program to demonstrate date object

```
<html>
<head>
<script type="text/javascript">
var d = new Date();
document.write(d+"<br>");
document.write(d.getFullYear()+"<br>");
document.write(d.setMonth(6));
document.write(d.getMonth()+"<br>");
document.write
```

```
document.write('d.getTime());
</script>
</head>
</html>
```

**Array:-**

An Array is a collection of homogenous data elements.

In javascript to create an Array use following forms.

1) var array-name = [ele1, ele2, ....elen];

eg:- var a = ["mon", "tue", "wed"];

↳ a.length = 3. (It returns no. of elements present in array)

2) var array-name = new Array(Size);

(or)

var array-name = new Array(ele1, ele2, ele3....);

eg:- var a = new Array(5);

```
a[0] = "monday";
a[1] = "tue";
a[2] = "wed";
a[3] = "Thur";
a[4] = "fri";
```

In case javascript the array size will be dynamically increased.

eg:- var a = new Array("mon", "Tue", "wed", "Thur");

Write a javascript program to read an array elements and display the sum of array elements.

```
<html>
<head>
<script type="text/javascript">
var n = parseInt(prompt("enter n value"));
var a = new Array(n);
var i, sum=0;
for(i=0; i<a.length; i++)
{
a[i] = parseInt(prompt("enter array element"));
sum = sum+a[i];
}
alert("sum is "+sum);
</script>
</head>
</html>
```

Methods of Array object:

1. reverse(): It returns the elements in reverse order

eg:- var a = [10,20,30,40];

var b = a.reverse();

2. sort(): It is used to arrange the elements in ascending order.

3. push(): It is used to add one (or) more elements at end of the array.

eg:- a.push(55);

a.push(60,70,80,90);

4. pop(): It is used to removes and returns last element from the given array.

eg:- x = a.pop();

document.write(a); o/p: 10,20,30

document.write(x); o/p: 40

5) unshift(): It is used to add one (or) elements at the beginning of array

eg:- var a = [10,20,30,40];

a.unshift(50,60,70);

6) shift():- It returns and removes first element of the given array.

7. IndexOf():- It returns the first occurence position of a particular element.

eg:- var a = [10,20,30,40]
x = a.IndexOf(20);

lastIndexOf():- It returns last occurent index position of a particular element

includes():- It used to check whether the particular element available (or) not. if it is available it returns true.

eg:- var a = ['mon', 'tue', 'wed'];
var b = a.includes('fri');
o/p: false

Slice(): It is used to extract a subset of element from the given array

Syntax: slice(start,end)
↓
start to end-1
slice(start)
↓
start to last index

eg:- var x = [10,20,30,40,50,60];
var y = x.slice(2,5);
var z = x.slice(2);
document.write(x); o/p: 10,20,30,40,50,60
document.write(y); o/p: 30,40,50
document.write(z); o/p: 30,40,50,60

Splice() :- It is used to add (or) remove the elements from the existing array

Syntax: splice(start, delete, ele1, ele2 .... elen)
↓ start: Starting index position
↓ delete: optional, it specifies how many ele has to delete
ele1, ele2 .... elen: optional, elements to insert

splice function returns removed elements. it modifies the existing array

eg: var x = ["mon", "Tue", "Thurs", "Fri"];
var y = x.splice(2);
o/p: x => mon, Tue
y => Thurs, fri

2) var x = ["mon", "Tue", "Thurs", "Fri"];
var y = x.splice(2,0,"wed");
o/p: x => mon, tue, thurs, fri

3) ~~var~~ var y = x.splice(2,1,"sat","wed");
o/p: x => Mon, Tue, sat, wed, fri
y => Thurs

x.splice(0,1); → it deletes the ele at 1st index (position)
x.splice(x.length-1,1); → it deletes the last element.

Two-dimensional array:-

To create an array in javascript 1st allocate the row size

syn:- var array-name = new Array[row-size];

Then allocate size for every column separately

3x3 matrix:

```
var a = new Array(3);
a[0] = new Array(3);
a[1] = new Array(3);
a[2] = new Array(3); (or)
for (i=0; i<a.length; i++)
{
  a[i] = new Array(3);
}
```

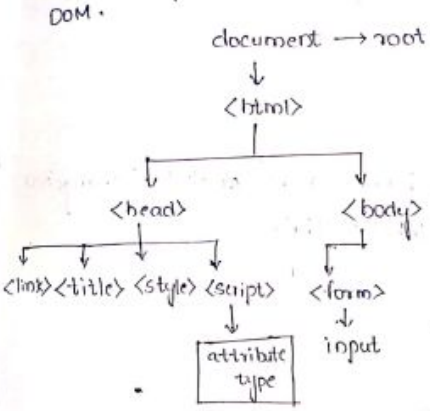
Write a java script program to read a matrix and display.

```
<html>
<head>
<script type="text/javascript">
 var r = parseInt (prompt("enter row size"));
 var c = parseInt (prompt ("enter colo size"));
  var a = new Array(r);
   for(i=0;i<a.length;i++)
    {
      a[i]= new Array(c);
    }
  for (i=0; i<a.length ; i++)
  {
   for(j=0;j<a[i].length ;j++)
    {
      a[i][j] = parseInt (prompt ('enter array elements'));
    }
  }
  for(i=0; document.write ("matrix elements <br>");
    for(i=0;i<a.length;i++)
     {
       for (j=0; j<a[i].length;j++)
       {
        document.write (a[i][j]+"  ")
       }
      document.write('<br>');
     }
</script>
</head>
</html>.
```

Dynamic HTML:-

It is used to add the dynamic nature to the static web pages using Java script and DOM (Document object Model)

→ whenever a web page is loaded into the web browser automatically a tree structure will be created using DOM.




Events in javascript:

Event is an action performed by the user.

To perform some operation in the javascript we need to handle the events properly.

⇒ onclick: when it generates whenever user clicks on some HTML element like image, table, button soon

getElementById(): It is used to return the html element based on id.

innerHTML: It is used to retrieve (or) modify the content of html element.

Syntax:

```
document.getElementById("id value").innerHTML
                                    ="content";
```

eg:-

```
<html>
<head>
<script>
function changeText()
{
 document.getElementById('it').innerHTML="oops!";
}
</script>
</head>
<body>
<h1 style="background-color:red;" onclick="changeText(
          id="it">hello</h1>
</body>
</html>
```

eg:-

```
<body>
 <form>name="f1">
 Enter a number:<input type="text" name="t1">
    result:]
<html>
<head
<script>
function factor()
{
  var n= parseInt(document.f1.t1.value);
    var fact=1;
   for(i=1; i<=n; i++)
    { fact = fact*i;
    }
   document.f1.t2.value = fact;
}
</script>
</head>
<body>
<form name=f1>
enter n value: <input type="text" name="t1"> <br>
  result: <input type="text" name="t2"><br>
  <input type="button" value="find" onclick="factor
                                          ()">
  </form>
  </body>
  </html>
```

2. onfocus:- it generates whenever a particular HTML element gains the focus.

3. onblur: it generates whenever a particular HTML element lost his focus.

eg:-

```
<head>
<html>
<script>
function Myfunction()
{
 document.getElementById("t1").style.background=
                                    "yellow";
}
function myfunction1()
{
 document.getElementById("t1").style.background=
                                    "green";
}
</script>
</head>
```

```
<body>
enter your name: <input type="text" id="t1"
        onfocus="myfunction()" onblur="MyFunction1()">
</body>
</html>
```

4) onsubmit: it generates when a user clicks on the submit button to submit the form contents.

Syntax: `<form name="f1" onsubmit="script-code" action="file-name">`

5. Onload: It generates when a web page is loaded into the web browser. It has to used with `<body>` tag.

eg:-

```
<html>
<head>
<script type="text/javascript">
 function display()
 {
  alert('welcome');
 }
</script>
</head>
<body onload="display()">
  <h1> hello </h1>
</body>
</html>
```

6. onmouseover :- It generates whenever user moves to the particular html element.

7. onmouseout :- It generates whenever user (or) cursor left from the particular html element.

8. onmouseup :- when we release click on the mouse.

9. onmousedown : when we click on the mouse

10. onchange: whenever user select an item in the radiobutton, (or) checkbox (or) in choice list then event is eb generated.

Write a javascript program to handle mouse events.

```
<html>
<head>
<script>
function bigImg()
{
 document.getElementById("i").style.height = "64px";
 document.getElementById("i").style.width= "64px";
}
function normalImg()
 {
  document.getElementById("i").style.height = "32px";
  document.getElementById("i").style.height = "32px";
 }
</script>
<head>
<body>
<img onmouseover="bigImg()" onmouseout="normalImg()"
   id="i" src="XML.jpg" alt="Smiley" width="32" height=
                                        "32">
```

</body>
</html>

eg:-

```
<html>
<head>
<script>
function mouseDown()
{
document.getElementById("myp").style.color="yellow";
}
function mouseup()
{
document.getElementById("myp").style.color="red";
}
</script>
</head>
<body>
<p id="myp" onmouseup="mouseUp()" onmousedown="mouseDown()">
mouse events
</p>
</body>
</html>
```

Write a javascript program to handle onchange event

```
<html>
<head>
<script>
function Mychange()
{
var x=document.getElementById("myselect").value;
document.getElementById('demo').innerHTML="you selected:"+x;
}
</script>
</head>
<body>
<p>select a car from the list</p>
<select id="myselect" onchange="Mychange()">
<option value="Audi">Audi
<option value="BMW">BWM
<option value="mercedes">mercedes
<option value="volvo">volvo
</select>
<p id="demo" style="color:red;"></p>
</body>
</html>
```

Regular Expression in javaScript:

→ Regular expressions are useful for defining the patterns.
→ It is used in pattern matching
→ It also used in searching and replacing
→ In javascript to create a regular expression use "RegExp" object.

Syntax: var exp = new RegExp("pattern", "flags");
(or) optional.
var exp = /pattern/flags;

eg: var exp = /it/;
var str = "hello It hello it";
o/p: It is used to check pattern 'it' is present in string (or) not.

Methods:

1. search(): It is used to search whether a particular pattern matches with the given string. if it is matches it returns first occurent index position

eg:- var str = "hello It hello it";
var exp = /it/;
var x = str.search(exp); → it returns only one value.
o/p: x is 15

→ if we use a flag value as i (it will act as case insensitive);
* If the given string not matches with the pattern it returns -1.
var e = /it/i;
var y = str.search(e);
o/p: y value is 6.

2. match(): It is also similar to search but it returns matched string from the given string and it also returns array of multiple matches if we use flag value as g.

eg:- var str = "hello It hello it";
var exp = /it/ig; → It will allow multiple matches.
var y = str.match(exp);
o/p: It, it

3. replace(): It is used to perform search and replace operations.

Syntax: replace(pattern, 'new-string');

4. test(): It is used to test whether a particular string matches with the pattern (or) not. if it is matches it returns true otherwise false

→ To design some patterns use the following rules:

i) To define ranges use [ ]

[abc] = {a,b,c} any one characters is selected
[a-z] = {a,b, .... z}
[a-zA-z] = {a,b,...z,A,B ...z}
[0-9] = {0,1,2..,9} only one digit

ii) Quantifiers:

In javascript we use the following Quantifiers.

1) [0-9] ⇒ only one
2) [0-9]* ⇒ zero (or) more
3) [0-9]+ ⇒ one (or) more
4) [0-9]? ⇒ zero (or) one
5) [0-9]{specific length}
eg: [0-9]{2} ⇒ 00, 01, ... 99.
6) [0-9]{start, end} ⇒ it will accept range from start to end
eg: [0-9]{3,5} ⇒ 3digits to 5digits.
7) [0-9]{start, } ⇒ greater than length everything is accepted.

iii) To represent a Regular Expression by starting with some character use cap (^) symbol.

^[A-z][0-9]{3}
↳ every thing starts with capital letter followed by with 3 digits.
o/p: A011, A112,

iv) To design a regular expression by specifying with ends with use dollar ($) symbol.

[0-9]$
↓ ends with digit

* c a javascript program to accept a string starts with a upper case followed by letters (or) digits.

```
<html>
<head>
<Script>
  var x = new RegExp("^[A-Z][a-zA-Z0-9]*");
  var y = prompt("enter a string");
   if(x.test(y))
   {
     alert("valid");
   }
   else
   {
     alert("invalid");
   }
</Script>
</head>
</html>
```

* Design an HTML document contains two form element by accepting name and phone number by satisfying the given constraints.

i) A name starts with a capital letter followed by a set of small letters ends with 2 digit

ii) A phone number must contains 10 digits.

```
<html>
<head>
<Script> function valid()
var x = new RegExp("^[A-Z][a-z]*[0-9]{2}$");
var y = new RegExp("[0-9]{10}");
 var name = document.f1.t1.value ;
   var phone = document.f1.t2.value;
   if(x.test(name))
   {
     if(y.test(phone))
     {
       alert("
         alert("valid");
     }
     else
     {
       alert("invalid phone number");
     }
   }
   else
   {
     alert("invalid name");
   }
}
</Script>
</head>
<body>
<form f1 name="f1">
Enter a name: <input type="tex" name="t1"> <br>
Enter a phone number: <input type="text" name="t2"> <br>
<input type="button" value="validate" onclick="valid()">
</form>
</body>
</html>
```

window.location: It is used for redirecting to next page (or) navigate to next web page.

Week-4 :-

Write a javascript program to validate the various fields in registration form.

1) username must be starts with small letter and followed by capital letters minimum length of 8 characters

2) password contains any uppercase (or) lower case (or) digits with width of 8-15

3) An email must be in specific form
example@domain.(com/in)

4) phone number must contains 10 digits

5) pin code must contains 6 digits.

UNIT-II

9/03/2019

PHP

→ PHP is used for adding dynamic content to a static webpage

→ A PHP is a server side scripting language.

→ PHP is invented by Rasmus Lerdorf in 1994 for monitoring is online resume and personnal information.

→ PHP is development in C language.

→ Initial name of PHP is personal home page and later abbrievated as hypertext pre processor.

→ The current version PHP is PHP 7.0

→ It is a interpreted language and it supports platform independent nature

→ PHP script always embedded in a HTML code.

→ All PHP files are saved with extension .php.

→ It is a case sensitive language.

→ To process any PHP application we require a server.

1) WAMP: window, Apache, MySQL, PHP
↓
operating system.

2) LAMP: Linux, Apache, MySQL, PHP it supports in Linux operating system.

3. MAMP: Mac, Apache, MySQL, PHP → it support in mac operating system

4. XAMPP: cross platform, Apache, MySQL, PHP, perl

Creation of a PHP script:

To include a PHP code use the following syntax

Syntax: <?php
= code
?>

comments in php:

1. Single line comment: // ——— (or) #
2. multi line comment: /* ——— */

Displaying a message:

To display a message using php use either echo (or) print.

eg:- i) echo "hello";

echo "<h1> hello </h1>";

echo "<br>";

print: it is also used for displaying a message but the difference is a print returns '1' as value. but echo doesn't return anything.

variables To Write a php script to display a simple message

```
<html>
<head> <?php
        echo "hello<br>";
        print "welcome";
      ?>
</html>
```

variables in php:

→ A variable is a name given to a memory location that holds some data.

→ In php all the variables must be declared using dollar sign.

Rules:

1) every variable must be starts with the letter (or) underscore (_).

→ A variable doesn't contain any special character except -

→ It is a case sensitive language

→ It is loosely typed language there is no need to specify the datatypes at the time of declaration

syntax: $var-name = value;

eg:- $x=5;
$_z=5.67;
$a = new Sample();

ii)
```
<html>
<?php
    $x=50;
    $y=20;
    $sum=$x+$y;
    echo "sum is $sum";
?>
```

127.0.0.1

→ If the server is available on the same system then we need to use local host (or) ip address 127.0.0.1 to access the server.

→ If the server is available at remote system we need to use IP address of the system.

→ on the browser we need to give local host then server home page will be displayed.

Constants in php:

In php to define a constant use two approaches

1. const keyword
2. define function

1. const: It is used to define case-sensitive constants.

Syntax: const const-name = "value";

example: const pi = 3.14;
const MAX = 50;
echo pi;
echo MAX;

There is no need of dollar sign in const keyword.

2. define : It is also used to define constants.
it will be treated as case-insensitive

Syntax: define ("const-name", "value", "case-insensitive");
↓ optional and boolean value

case-insensitive is a boolean operand the default value is false. it will be act as case-sensitive.

example: define ("sum", "50", true);

eg: * case-insensitive
echo sum;
echo SUM;
echo Sum;
echo SUM;

* To make it as case-insensitive give boolean operand as true.

③ define ("MAX", "50");
↳ echo MAX; case-sensitive
echo Max → not possible.

Datatypes in PHP:

In PHP we can store different types of data in a variable.

Datatypes are classified into 8 types.

1) Integer ⇒ $x = 50; $z = -567;
2) Float (or) Double ⇒ $x = 5.67; we can store decimals
3) Boolean ⇒ $x = true (or) false;
4) String ⇒ $x = "Hello" and $y = 'hello';
5) Array ⇒ $x = array ("red", "blue", "green");
6) NULL ⇒ $x = null
7) ⑥ Resource ⇒ In a particular variable we can store the database (or) file as resource.
8) Object ⇒ It is used to store object type date.
$x = new Date();
$x = new @Sample();

Scalar: integer, float, boolean, string → single value

Compound: Array, object ⇒ multiple values.

Special: Resoure, null

Write a PHP script for creating a simple class.

```
<html>
<?php
  class Sample
  {
    function display()
    {
      echo "hello <br>";
    }
  }
  $x = new Sample();
  $x.display();
?>
</html>
```

Functions in PHP:

1) var_dump(): it returns the datatype and value and it displays on screen

Syntax: var_dump(var-name);

example $x = 50;

echo $x;

var_dump($x); → int 50

2) gettype(): It is used to retrieve the datatype of a particular variable

Syntax: gettype(var-name);

3) is_integer(): It is used to check whether a given variable is integer (or) not. If it is true it return 1

Syntax: is_integer(var-name);

eg:- $x=100;

$y= is_integer($x);

Echo $y; → 1

4) is_double():

5) is_string():

6) is_boolean()

7) is_array()

8) is_object()

Operators in PHP:

In PHP operators are classified into 7 types

1. Arithmetic operators
2. Assignment operators
3. Comparision operators
4) increment/decrement
5) logical operators
6) string operators
7) array operators

Arithmetic ⇒ +, -, *, /, %, ** ⇒ x**y ⇒ $x^y$

Assignment ⇒ =, +=, -=, *=, /=, %=

Comparision ⇒ <, <=, >, >=, ==, ===, !=(or)<>, !==

$x = 100;

$y = "100";

if($x == $y) o/p: true

↳ it returns true because the value is same, irrespective of datatype

if($x === $y) o/p: false

↳ it returns true if both value and datatype are similar

!= : it returns true if both values are different.

!== : it returns true if values are same and datatypes are different.

increment/decrement;

$x++, ++$x;

5) logical Operators:

1. logical AND : &&, and
2. logical OR : ||, or
3. logical NOT : !
4. XOR : if both are same ⇒ false (or) true.

6) String operators:

1. . : concatination operator

.= : concatination operator assignment operator.

eg:- $x="it";

$y="hello";

$x= $x.$y;

echo $x; ⇒ ithello

```
echo "x : $x";
echo "x is $x";

$x .= "it";
  ↓
$x = $x."it";
  ↳ o/p: it it
```

array operators:

+ : union of two arrays

== : it returns true if the both arrays contain same set of element.

=== : it returns true if both arrays contain same set of elements in same order

!=/<> : it returns true if both arrays contains different elements.

!== :

Control structures in PHP:

In PHP control statements are classified into 3 types

1) selection
2) looping
3) Branching

Selection statement:

1. if
2) switch

if → Simple if
- If else
- Nested if
- else-if ladder

Simple if :
```
if (cond)
{

}
```

if-else :

Syntax:
```
if (cond)
{

}
else
{

}
```

else if ladder :

Syntax:
```
if (cond)
{

}
elseif (cond)
{

}
else
{

}
```

2) Looping statements:

In PHP loop statements are classified into 4 type

1. while loop 2. do while loop 3. for loop 4. foreach loop

foreach loop: It is used to access the array elements

Syntax:
```
foreach ($array-name as $var-name)
{

}
```

Example:

```
$x = array(10,20,30,40,50);
```

```
for($i=0; $i<$x.length; $i++)
{
  echo $x[$i];
}
```

```
foreach ($x as $value)
{
  echo $value."<br>";
}
```

o/p: 10 20 30 40 50

For array sum

```
$sum=0;
foreach($x as $i)
{
  $sum=$sum+$i;
}
```

1). Explain different types of web architecture

a) two-~~trial~~ b) three-~~trial~~ tier

Functions in PHP:-

A block of statements is used to perform a specific task is called function. All the function names in PHP are case insensitive.

Syntax: function func-name(args-list)
{
//statements
}

→ All func-name should start with letter (or)_.

eg:- i) function display()
{
echo "hi";
}
→ without arguments.

ii) function add($x,$y)
{
$z=$x+$y;
echo $z;
}
→ with arguments

iii) function add($x,$y)
{
return $x+$y;
}
→ with return value.

iv) syntax:- function fun-name($var=value)
{
}
→ with default arguments

```
function display($x=50)
{
 echo $x;
}
 display(60);
 display("it");
 display();
```
o/p: 50
by default all functions store same value.

write a php script to find factorial of given number using recursion.

```
<html>
<?php
 function factor($n)
 {
  if($n==0)
  {
   return 1;
  }
  else
  {
   return $n*factor($n-1);
  }
 $res=factor(6);
  echo "$res";
 }
?>
</html>
```

Arrays in PHP:-

→ An array is a collection of homogenous data element

→ In PHP arrays are classified into 3 types

1. Indexed array
2. Associative array
3. Multidimensional array

Syntax: $arr_name = array(ele1, ele2, ....)

eg: $x = array("it", "cse", "ece");

(or)

$x[0] = "it";

$x[1] = "cse";

$x[2] = "ece";

count():-

It returns the number of elements available in given array.

```
for($i=0; $i<count($x); $i++)
{
echo $x[$i];
}
```

2. Associative array:

In associate array every element will be represented as key/value pair. In case of associate array an element can be accessed based on its key value.

All key names must be unique.

Syntax: $array_name = array("key1"=>"value" "key2"=>"value2");

eg: $sal = array("abc"=>50,000 "xyz"=>60,000,....)

(or)

$sal["abc"] = 50,000;

$sal["xyz"] = 60,000;

To access associate array elements use following syntax

```
foreach($arr_name as $key => $value)
{
}
```

eg: write a php script to demonstrate associate array.

```
<html>
<?php
$sal = array("abc"=>40,1000 value, "pqr"=> 50000, "xyz"=>60000);
foreach($sal as $k=>$v)
{
echo $v;
}
?>
</html>
```

Array methods in PHP:

1) sort(): It is used to arrange the elements in sorted order (ascending)

eg: $x = array(5,10,23,7,2);

sort($x);

↳ sort the elements and store in same array.

2) rsort():- It is used to display array elements in descending order.

3) ksort():- It will display elements in ascending order based on key value (It is used with associative array)

4) krsort():- It will display elements in descending based on key value.

5) asort():- display elements in descending order based on value.

6) arsort():- display elements in ascending order based on value.

7) print_r():- It will display array elements (instead of loops)

eg:- print_r($x)

o/p:- Array([0]=>5, [1]=>10, ...)

8) array_reverse():-
It returns an array that contains array elements in reverse order

```
$x = array(7,13,12,14);
$y = array_reverse($x);
print_r($y) ⇒ ([0]⇒14,[1]⇒12, ....)
```

9) array_search():-
It is used to search the specified elements and it returns either index or key name if it is value.

syntax: array_search("key element", array_name);

```
$x = array(7,13,12,14,19)
$y = array_search(14,$x)
echo $y:-
    ↳ o/p: 3
```

10) array_intersect(): It returns an array of common elements available in two array

```
array_intersect(array1,array2);
eg:- $x = array(10,20,30,40);
     $y = array(5,10,15,20);
     $z = array_intersect($z,$y);
     echo $z;
        ↳ 10,20
```

11) array_sum(): It returns sum of array elements.

```
syntax: array_sum(array_name);
        $x = array(5,10,15);
        $y = array_sum($x);
        echo $y;
```

12) array_product():
It returns product of all array elements

13) array_push():
used to add one (or) more elements at end of the array.

syntax: array_push(array_name, ele1, ele2, ...)

```
array_push($x,20,25,30);
print_r($x);
```

14) array_pop():- It is used to delete last element from the array

15) shuffle():-
It is used to shuffle the array elements randomly

```
eg:- shuffle(arr_name);
```

16) array_unique():-
It is used to remove duplicate elements in the given array and returns unique elements of the array

```
eg:- array_unique($x);
```

17) array_shift():
used to remove the 1st element from given array and returns deleted element

```
$x = array(5,10,23,7,2);
$y = array_shift($x);
   echo $y;
      ↳5
print_r($x);
      ↳ 10,23,7,2
```

1) array_combine():-

used to combine 2 indexed arrays into an associative array.

Eg:- $x = (1961, 1962, 1963);

$y = ("abc", "xyz", "pqr");

$z = array_combine($x, $y)

$x is key

$y is value

19) array_slice():

20) array_chunk():

used to divide the array into several chunks or paths

array_chunk(array_name, size);

3) Multi-Dimension Array:-

To create 2D array use following syntax

2D:- $array_name = array

(

array(-,-,-)

array(-,-,-)

array(-,-,-)

);

3D

$x = array

(array

(array(10,20,30);

array(40,5,10);

));

Forms And Database:

Retrieving data from forms:

→ To retrieve the data from form (or) to validate the form contents we require 3 attributes in <form> tag

1. name → used to specify the name of a form
2. action – used to specify target script url
3. Method – It is a http method used to submit form contents to web server

There are 2 types of http methods.

1) GET 2) POST

→ In case of GET method after submitting the form contents the data is visible in the address bar in terms of query string. some of the special characters are not allowed

→ By using g. GET method we can send max of 2000 characters

→ In case of POST method the data will be transferred in a secured manner and we can send any amount of data.

→ In PHP to retrieve the form contents use suitable global variable based on the http method

if method is GET use $_GET["name"]

if method is POST use $_POST["name"]

$_REQUEST → It is used for both ge GET and POST.

write a PHP script to retrieve data from form and display

f1.html

```
<html>
<body>
<form method="post" action="f1.php">
Enter a name: <input type="text" name="t1">
<input type="submit" value="submit">
</form>
</body>
</html>
```

in query string

file-name.php?content

f1.php

```
<html>
<?php
$y = $_POST["t1"];
echo "welcome to".$y;
(or)
echo "welcome to".$_POST["t1"];
?>
</html>
```

Note:-

History object is used to maintain list of url's visited by user.

It contains 2 methods

1) back - it is used to retrieve previous visited page
2) go - it is used to retrieve specified visited page

eg:- go(-1) → previous visited page

go(-2) → 2 pages back

To validate the form initially we need to check whether the field contains some data (or) not.

Syntax: if($_POST["name"])

if data is entered it returns true otherwise, it returns false.

write a PHP script to validate the form contents

f1.

f1.html

```
<html>
<body>
<form method="post" action="form_valid.php">
Enter a name: <input type="text" name="userid"><br>
password : <input type="text" name="pass"> <br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

form_valid.php

```
<?php
$flag=1;
$msg=" ";
if($_POST['userid'])
{
 if($_POST["pass"])
 {
  $flag=1;
 }
 else
 {
  $msg="password not entered";
  $flag=0;
 }
}
else
{
 $msg="username not entered";
 $flag=0;
}
if($flag==0)
{
 echo'<h1>$msg</h1>';
 echo"<input type="button" value="back" onclick=
                                "history.back()">";
}
else
{
 echo"<h1>all are valid</h1>";
}
?>
```

procedure to establish a connectivity between php and MYSQL database:

→ To establish a connectivity between php application and MySQL database use the following steps

1) establish a connection with MYSQL database
2) select a database
3) prepare a SQL Query and execute.
4) process the results
5) close the connection

Step1: establish a connection

To establish a connection from php application use a built-in function → mysqli_connect()

Syntax:- $conn = mysqli_connect ("host name", "username", "password", "database name");

If MySQL database is available on the local system use local host (or) 127.0.0.1 as host name

* database-name is an optional. It specifies on which database we are currently working.

→ If connection is not establish it returns NULL

→ To display error message related to connections call the built-in function mysqli_connect_error();

Note:-

die(): It is used to display the error messages and terminate from the php application.

Syntax:- die("unable to connect");

(or)

die("mysqli_connect_error()");

↳ with reason

Step2: selecting a database

To select a database use a builtin function

mysqli_connect_db()

Syntax:- mysqli_connect_db(conn,"db-name");

eg:- mysqli_connect_db($conn,"it-b");

while selecting a database if any error occurs use a builtin function

mysqli_error($conn);

Step3:- prepare a sql query and execute

To prepare and process an sql query use a built-in function mysqli_query();

Syntax:- mysqli_query($conn,"query");

If Query is related to select (or) describe then it returns result set contain some set of values.

→ if Query is related to insert, delete, update (or) create then it returns true (or) false.

eg:- $x= mysqli_query($conn,"select x from emp");

$q="insert into emp values(' ')";

$x= mysqli_query($conn, $q);

Step4:- process the results

1) mysqli_num_rows():- It returns number of rows available in result set.

Syntax:- mysqli_num_rows(result_set);

eg:- mysqli_num_rows($x);

2) mysqli_num_fields():- It returns the number of columns available in the result set.

3) mysqli_fetch_row():- It is used to fetch the single row from the result set interms of indexed array

syntax:- $row= mysqli_fetch_row(result_set);

$x

echo $row[0]; ⟹ First column

echo $row[1]; ⟹ Second column.

4) mysqli_fetch_assoc():- It returns single row from the result set interms of Associative array.

syntax:- $row= mysqli_fetch_assoc(result_set);

$x

echo $row["ename"];

echo $row["sal"];

5) mysqli_fetch_array():- It returns single row from the result set interms of either indexed array (or) Associative array

Syntax:- $row= mysqli_fetch_array(result_set);

↓

it can access both any type of arrays.

(or)

$row= mysqli_fetch_array(result_set, type);

↓ optional

↓

MYSQLI_ASSOC

MYSQLI_NUM

MYSQLI_BOTH

Step5:- close the connection

To close the connection use the following function

mysqli_close(conn_var);

# XML

22/01/19

XML stands for eXtensible Markup language

→ It is used to store and transport the data over a network.

→ It is a platform independent language is used for providing self description of a particular data.

→ It is invented by W3C (world wide web consortium) in 1998 ↓ organisation

Differences between HTML and XML

| HTML | XML |
|---|---|
| → It is used to display data and main focus on how data looks | → It is used to describe the data and main focus on what data is |
| → It contains predefined tags to design a HTML document | → It contains user defined tag to construct a XML document. |
| → HTML is a case-insensitive language | → XML is a case-sensitive language |
| → It will not check the syntax | → It will check the syntax |
| → In HTML, some of the elements not required the ending tags | → In XML, every XML elements required the ending tags |
| → It does not preserves coblde spaces | → It preserves white spaces. |

Every XML document contains XML elements and attributes associated with it.

Well-formed XML document :-

If a XML document follows the syntax rules of XML language then it is set to be a well-formed XML document

Syntax rules:

1) every XML document must contain one root element

eg: <root>
|
</root>

2) All XML elements must have closing tag

3) All XML elements are case-sensitive

4) All XML elements are properly Nested

5) All XM attribute values must be enclosed in either ' ' (or) " "

6) To display special characters use various entity symbols

| | |
|---|---|
| < < | &quotes; " " |
| > > | ' ' ' |
| & & | |

Comments:

In XML comments can be included by using

<!-- comments -->

XML elements :-

XML elements are building blocks of XML document to design any XML elements we need to follow some naming rules.

naming rules:

1) every XML elements starts with letter (or) _

2) An XML element may contains letters, digits, _'s, -'s and periods (.)

eg:- <root>
</root>
<book-name>
<book-name>
<book.name>

<book123>

3> All XML elements are case-sensitive

4> we can use any word except XML

5> It preserves white spaces (or) XML elements cannot contain spaces

write an XML document to maintain sigle student

```
<?xml version = "1.0" ?>  → optional  (information)
<student>
  <name> rushitha </name>
  <rollno> 17881A1940 </rollno>
  <age> 19 </age>
  <branch> it </branch>
</student>
```

•XML extension

write an XML document to maintain books catalogue each book contains child elements like Author, title, publisher and cost.

```
<catalogue>
  <book>
    <author> Berners'lee </author>
    <title> html </title>
    <publisher> ABC </publisher>
    <cost> 500 </cost>
  </book>
  <book>
    <author> Dennis ritche </author>
    <title> c language </title>
    <publisher> xyz </publisher>
    <cost> 1000 </cost>
  </book>
  <book>
    <author> williams </author>
    <title> machines </title>
    <publisher> POR </publisher>
    <cost> 1500 </cost>
  </book>
</catalogue>
```

Valid XML document :-

→ If a XML document follows the rules given by Document Type Definition (DTD) (or) XML Schema Definition (XSD) then we conclude the given document is valid XML document.

Document Type Definition (DTD):-

→ DTD is used to define structure of XML document It provides rules for defining XML elements and attributes

DTD For defining XML elements :-

1> To define root element use the following syntax

Syntax :- <!ELEMENT root_name (child1, child2, ....)>

eg:- <!ELEMENT Student (rollno, name, age, branch)>

<!ELEMENT catalogue (book)+> (one or more)

<!ELEMENT book (author, title, publisher, cost)>

2> To define child elements use the following syntax

Syntax:- <!ELEMENT ele-name (datatype)>

In DTD we can use of two types of datatypes

1> CDATA 2> PCDATA

1> CDATA : unpassed character data means the browser is not going to process any content

2> PCDATA : passed character data means the browser is going to transform all entity symbols

eg:

<!ELEMENT roll-no (#PCDATA)>

<!ELEMENT name (#PCDATA)>

To validate XML document we need to include DTD in the XML document

It can be done in 2 ways

1> Internal DTD 2> External DTD

1> Internal DTD:

If we include DTD rules in the XML document then it is said to be Internal DTD.

Syntax: <!DOCTYPE root-name[

// Element declaration

]>

Eg: Create internal DTD to maintain student information contains the following child elements name, roll-no. DoB, blood group and branch.

```
<!DOCTYPE college[
  <!ELEMENT college (student)*>
  <!ELEMENT student (name, roll-no, dob,
                     blood-group, branch)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT roll-no (#PCDATA)>
  <!ELEMENT dob (#PCDATA)>
  <!ELEMENT blood-group (#PCDATA)>
  <!ELEMENT branch (#PCDATA)>]>
<college>
<student>
<name> sushitha </name>
<roll-no> 1201 </roll-no>
<dob> 13-02-2001 </dob>
<blood-group> AB+ </blood-group>
<branch> IT </branch>
</student>
<student>
<name> ABC </name>
<roll-no> 1205 </roll-no>
<dob> 23-06-2000 </dob>
<blood-group> O+ </blood-group>
<branch> IT </branch>
</student>
</college>
```

2. External DTD:

Create an external document contains various DTD rules and saved with extension '.DTD'.

→ link external DTD file with XML using the given syntax

Syntax:

<!DOCTYPE root-name SYSTEM url>

eg: <!DOCTYPE root-name SYSTEM SJ.dtd>
college

Create external DTD for maintaining employee information contains various child elements like empid, empname, designation, salary and address. Address contains various child elements like HNO, Street name, locality, pincode.

emp.dtd

```
<!ELEMENT employee (empid, ename, desg, sal,
<!ELEMENT address (HNo, S_name, locality, pin)>
<!ELEMENT empid (#PCDATA)>
```

```
<!ELEMENT ename (#PCDATA)>
<!ELEMENT desg (#PCDATA)>
<!ELEMENT sal (#PCDATA)>
<!ELEMENT HNO (#PCDATA)>
<!ELEMENT S.name (#PCDATA)>
<!ELEMENT locality (#PCDATA)>
<!ELEMENT pin(#PCDATA)>]>
```

emp.XML

```
<!DOCTYPE employee SYSTEM emp.dtd>
<employee>
<empid> 1201 </empid>
<ename> Gundu </ename>
<desg> Manager </desg>
<sal> 10000 </sal>
<address>
<HNO> 1-4 </HNO>
<S.name> adarshnagar </S.name>
<locality> korutla </locality>
<pinNo 505306 </pin>
</address>
</employee>
```

plugin → plug-admin → XML Tools → click and install
↳ validate_now

Defining Entities in DTD:

To define entity in the DTD use the following Syntax

Syntax:- <!ENTITY ent_name "ent_value">

eg:- <!ENTITY it "Information Technology">

<branch> ⁢ </branch>

Defining Attributes in DTD:

attributes are useful for providing additional information for a particular XML element.

eg:- <book id="501">
|
</book>

To define attributes in the DTD use the following Syntax

Syntax:- <!ATTLIST ele_name att_name att_type att_value>

ele_name ↓ name of element; att_name ↓ name of attribute

att_type : type of attributes specifies

1. CDATA → character data
2. enumeration → set of enumerated values
3. ID → (It must be a unique id)

att_value : It has 4 types

1. value - it will be treated as default value
2. #REQUIRED - in the XML document an attributes must be required
3. #IMPLIED - In the XML document may (or) may not include attribute.
4. FIXED value - The attribute value is fixed.

eg: Create Internal DTD for the given XML Document

```
<book id="2">
 <title> ____ </title>
 <author> ____ </author>
 <pages> ____ </pages>
 <price> ____ </price>
 </book>
```

```
<!DOCTYPE book[
 <!ELEMENT book (title,author,pages,price)>
 <!ATTLIST book id ID #REQUIRED>
 <!ELEMENT title (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT pages (#PCDATA)>
 <!ELEMENT price (#PCDATA)>
 ]>
```

eg: Create an Internal DTD for a catalog of four stroke motor bikes where each motor bike as following child elements like make, model, year, colour, engine, chasis number and accessories. The engine element has child elements like engine number, number of cylinders and type of fuel The accessories elements has attributes like disc brake, auto start and radio, each of which is required and possible values like yes or no. Entities must be declared for names of popular bike names

```
<!DOCTYPE catalog[
<!ELEMENT catalog (bike)+>
<!ELEMENT bike (make,model,year,colour,engine,
        chasis-number,accessories)>
<!ELEMENT engine (eng-num,no-of-cylin,type-of-fuel)>
<!ATTLIST accessories disc-brake enumeration(yes|no)
                #REQUIRED>
<!ATTLIST accessories auto-start enumeration(yes|no)
                #REQUIRED>
<!ATTLIST accessories radio enumeration(yes|no)
                #REQUIRED>
<!ENTITY p "Bajaj pulsar">
<!ENTITY b "bike">
<!ELEMENT make(#PCDATA)>
<!ELEMENT model(#PCDATA)>
<!ELEMENT year(#PCDATA)>
<!ELEMENT colour(#PCDATA)>
<!ELEMENT chasis-number(#PCDATA)>
<!ELEMENT accessories(#PCDATA)>
<!ELEMENT eng-num(#PCDATA)>
<!ELEMENT no-of-cylinder<#PCDATA)>
<!ELEMENT type-of-fuel(#PCDATA)>
]>
<catalog>
<bike>
 <make>duke</make>
 <model>&p</model>
 <year>2018</year>
 <colour>black</colour>
 <engine>
 <eng-num>123</eng-num>
 <no-of-cylin>2</no-of-cylinder>
 <type-of-fuel>die petrol</type-of-fuel>
```

```
<lengine>
<chasis-number>os</chasis-number>
<accessories disc-break = "yes" radio='yes' autostart =
                                        "yes"/>
</bike>
</catalog>
```

Steps: Create an XML document and provide some reference to .css file

Syntax: <?xml-stylesheet type="text/css" href="file-name"?>

Create an XML document contains book catalogue each book contains title, author and price.

book.css

```
catalog
{
  border-style : dashed;
}
book
{
  border-style : dotted;
  border-color : pink;
}
title
{
  color: blue;
}
author
{
  font-size: 50;
  text-transform : uppercase;
}
price
{
  font-family : verdana;
  font-weight : bolder;
}
```

Book.xml

```
<catalog>
  <book>
    <title> Java </title>
    <author> james </author>
    <price> 500 </price>
  </book>
  <book>
    <title> HTML </title>
    <author> bond </author>
    <price> 1000 </price>
  </book>
</catalog>
```

## XML Schema (XSD):

It is used to describe the structure of XML document and it is also referred as XML Schema definition language (XSD)

Differences b/w DTD and XSD

| | DTD | XSD |
|---|---|---|
| 1> | DTD not follows the syntax of XML | XSD follows the same syntax as XML |
| 2> | DTD not supports any datatypes it contains CDATA and PCDATA | XSD supports various datatypes like string, data, boolean... |
| 3> | In DTD we cannot impose restrictions over the data | In XSD we can impose restrictions over data. |
| 4> | In DTD name spaces are not allowed | In XSD name spaces support must be required to avoid naming collisions. |
| 5> | In the DTD it XML document we can define the elements in any order and the all elements in DTD must be required. | In XSD we can impose the restrictions in over the order of sequence. |

→ To create XML Schema use a root element (schema)

To use various XSD elements and various XSD datatypes we need to create a name space.

Syntax:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/
                    XMLSchema">

</xs:schema>
```

XSD elements:-

To create an element using following element

<xs:element>

→ In XSD elements are classified into two types

1> Simple type element

2> complex type element

Simple type element :-

An element contains only value and it does n't contain any child elements or attribute

eg:- <age> 30 </age>

<price> 500 </price>

to create simple type element use <xs:simpleType>

Complex type element :

An element can contain some child elements and attributes.

eg:- <book>

<author> James </author>

<price> 500 </price>

</book>

To create complex type elements use <xs:complexType>

→ To create a XSD element use the following syntax.

Syntax:-

<xs:element-name = " " type="datatype" />

↳ xs:string

eg:- <xs:element_name="age" type="xs:integer"/>xs:integer

xs:decimal

<xs:element-name="name" type="xs:string"/> xs:boolean

xs:date

xs:time

XSD indicators:-

XSD indicators are useful for to control how the elements can be used in XML document.

We have 2 types of indicators

1> Ordered indicators 2>Occurence indicators

Order indicators :-

Order indicators are used to control the order of child elements.

We have 3 order indicators.

<xs:all> → all elements must be required in any order

<xs:sequence> all elements must be required in sequence order

<xs:choice> either we use any elements

Occurence indicators:

We have 2 occurence indicator

1) min occurs

2) max occurs

for both default values are 1

maxOccurs = unbounded

↓

any no.of elements are included.

For '*': minOccurs = "0" and maxOccurs = "unbounded"

For '+': minOccurs = "1" and maxOccurs = "unbounded"

For '?': minOccurs = "0" and maxOccurs = "1".

eg:-

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="note">
      <xs:complexType>
      <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
      </xs:sequence>
      </xs:complexType>
      </xs:element>
      </xs:schema>
```

Save as .xsd

note.xml:

```
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="note.xsd">
  <to>sushitha</to>
  <from>raju</from>
  <heading>reminder</heading>
  <body>hi</body>
  </note>
```

eg:-

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="addresses">
<xs:complexType>
<xs:sequence>
<xs:element ref="address" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="address">
<xs:complexType>
<xs:sequence>
<xs:element name="name" type="xs:string"/>
<xs:element name="street" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

address.xml

```
<addresses xmlns:xsi="http://www.w3.org/2001/XMLSchema-
        instance" xsi:schemaLocation="address.xsd">
  <address>
  <name>ruchitha</name>
  <street> xyz </street>
  </address>
  <address>
   <name>
   </street>
  </address>
  </addresses>
```

XSD restrictions:-

To impose some restrictions over the data use a tag <xs:restriction>. it contains only one attribute i.e base. here base specifies type of data.

1) range of values:

To limit (or) specify the range for a particular XML element use

```
<xs:minInclusive>
<xs:maxInclusive>
```

eg:-

```
<xs:element name="age">
<xs:simpleType>
<xs:restriction base="xs:integer">
<xs:minInclusive value="0" />
<xs:maxInclusive value="120" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

2. Enumeration:-

To define set of possible values use

```
<xs:enumeration>
```

eg:-

```
<xs:element name="car">
  <xs:simpleType>
  <xs:restriction base="xs:string">
  <xs:enumeration value="Audi" />
  <xs:enumeration value="BMW" />
  </xs:restriction>
  </xs:simpleType>
  </xs:element>
```

3. pattern:

To define patterns over the data use

```
<xs:pattern>
```

Syntax: `<xs:pattern value="[a-z]*" />`

"[A-Z][A-Z][A-Z]" → 3 upper case letters

"[xyz]" → any one letter from x,y,z

"([a-z])*"

4. Restrictions on length

To impose restriction over the length of a particular XML element

```
<xs:element name="password">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:length value="8" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

<xs:minLength> → specifies minimum number of characters required for particular XML element.

<xs:maxLength>

XSD attributes:

To specify attributes for any XML elements use <xs:attribute>

Syntax: <xs:attribute name="xxx" type="yyy"/>

eg: <xs:attribute name="lang" type="xs:string"/>

Default value:

A default value is automatically assigned to the attribute when no other value is specified.

To provide some default value use default attribute

Syntax: <xs:attribute name="lang" type="xs:string" default="EN"/>

Fixed value:

A fixed value is automatically assigned to the attribute when and you cannot specify another value

Syntax: <xs:attribute name="lang" type="xs:string" fixed="EN"/>

Optional and required attributed:

→ Attributes are optional by default. To specify that the attribute is required, use the "use" attribute.

Syntax:

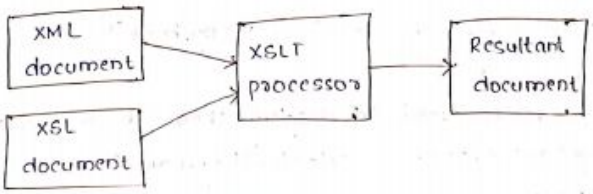
<xs:attribute name="lang" type="xs:string" use="required"

XSLT:-

XSLT → extensible stylesheet Language transformation

→ XSLT is used for transform the given XML document into some required format

→ by using XSLT we can convert XML to HTML docume




Creation of XSL document:

→ XSL is also developed by w3c

Every XSL document contains a root tag <stylesheet>

Syntax:

<xs:stylesheet version="1.0"

xmlns:xs="http://www.w3.org/1999/XSL/Transfo

→ To create some require formal we need to use various XSL elements.

1. <xs:template>:

it is used to create some template for formatting the given XML document and it contains the set of rules to access the content.

Syntax:

<xs:template match="tag-name"/> → for specific document

<xs:template match="/"> → is applicable for whole document

2. <xs:value-of>

It is used to retrieve the content of a particular XML element.

Syntax: <xs:value-of select="tag-name"/>

To retrieve the attribute value use the following syntax

Syntax: <xs:value-of select="@attri-name"/>

3) <xs:for-each>
It is used to repeat the same rules for various tags

Syntax: <xs:for-each select="catalog/book">

4. <xs:sort> :
It is used to display them in sorted order.

Syntax: <xs:sort select="tag-name"/>

5. <xs:if> :
it is used to check the condition

Syntax: <xs:if test="condition">

6. <xs:choose> :
To impose multiple conditions use <xs:choose>
<xs:choose

write a xsl program to display employee in tabular form.

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
    <body>
      <h2>Employee</h2>
      <table border="1">
      <tr bgcolor="pink">
        <th>Id</th>
        <th>firstname</th>
        <th>lastname</th>
        <th>nick name</th>
        <th>Salary</th>
      </tr>
  <xsl:for-each select="class/Employee">
    <tr>
    <td><xsl:value-of select="@id"/></td>
    <td><xsl:value-of select="firstname"/></td>
    <td><xsl:value-of select="lastname"/></td>
    <td><xsl:value-of select="nickname"/></td>
    <td><xsl:value-of select="salary"/></td>
    </tr>
  </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

XML

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="employee.xsl"?>
<class>
 <employee id="001">
  <firstname>Aryan</firstname>
  <lastname>Gupta</lastname>
  <nickname>Raju</nickname>
  <salary>30000</salary>
 </employee>
```

```
<xs:choose>
  Syntax: <xs:choose>
            <xs:when test="cond">
            #data
          </xs:when>
          <xs:when test="cond2">
          #data
          </xs:when>
```

```
<xs:otherwise>
  ||
</xs:otherwise>
</xs:choose>
```

XML parsers :-

A parser is a word used in compiler.

→ A parser is a program (or) a software library is used to check the given document is valid (or) not.

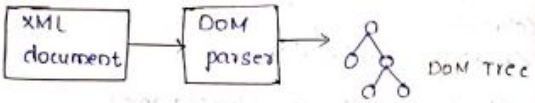
→ If it is valid an XML document will be converted into some memory representation to access the content.

In XML parsers are classified into 2 types

1) Document object Model (DOM)
2) Simple Api for XML (SAX)

DOM parser :-

By using DOM parser an XML document can be converted into tree structure. it contains a root node named as "document"



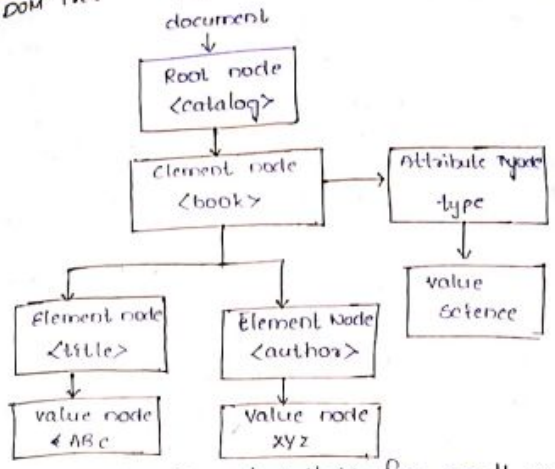

A DOM Tree contains various types of nodes.

1. Root node 2. Element node 3. Text node
4. Attribute node
5. value node

example:

```
<catalog>
  <book type='science'>
  <title> ABC </title>
  <author> XYZ </author>
</book>
</catalog>
```

DOM Tree




→ DOM parser is only suitable for small XML files. in case of large files it requires more memory to store the tree structure.

SAX parser :-

A SAX parser is not going to create any intermediate representation.

→ It is a event based parser for every a SAX parser will generates some event

→ To handle those events we need to design suitable methods

SA 1) start Document()
2) endDocument()
3) startElement()
4) endElement()
5) characters()

| DOM | SAX |
|---|---|
| 1> A DOM is a tree based parser | 1> A SAX is a event based parser |
| 2> In case of DOM an XML document can be converted into tree structure | 2> A SAX parser not creates any internal structure |
| 3> more memory is required to store tree structure | 3> It we require less memory |
| 4> DOM not suitable for large XML files | 4> it is suitable for XML files |
| 5> In DOM we can perform both read and write operations | 5> In SAX only reading is allowed. |
| 6> In case of DOM we can traverse in both backward and forward manner and we can also access randomly. | 6> In SAX only in forward traverse is allowed. |

Unit - 5

11/02/2019

## JDBC

→ JDBC stands for java Database connectivity

→ It is used to connect with database from java Application we require JDBC API

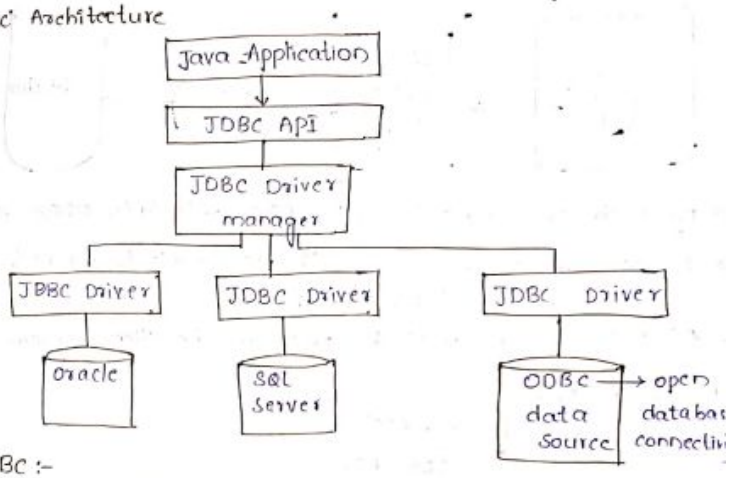
→ To use (or) access JDBC API we need to import sql package.

```
import java.sql.*;
```

JDBC Architecture:

→ It is used to establish a communication between java application & database.

JDBC Architecture




ODBC :-

→ ODBC stands for open Database connectivity

→ it is developed by microsoft but it is only suitable for windows platform

→ all the ODBC drivers are automatically installed at the time of installation of os.

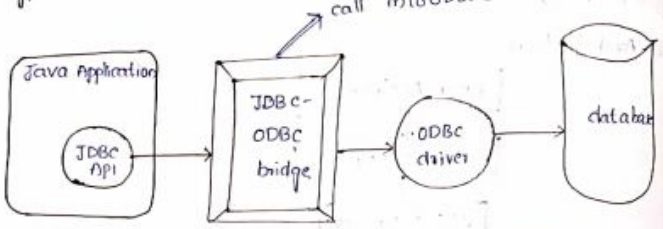
→ For every database vendor there is a separate ODBC driver is available.

Types of Drivers:

JDBC drivers are classified into 4 types-

- Type 1 (-JDBC-ODBC bridge driver)
- Type 2
  - Native API, partially Java
- Type 3
  - JDBC Network Driver, partially Java
- Type 4
  - 100% Java (pure java driver)

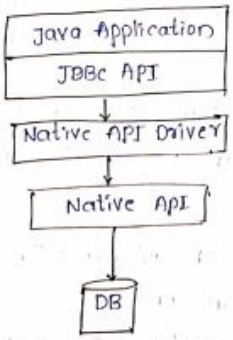
Type 1 Driver




1) Type 1 driver translates all JDBC calls into ODBC calls
2) A ODBC driver process all the ODBC calls to interact with a suitable database

→ All ODBC drivers must be exist on the client system

Type 2 driver :-




In case of Type 2 driver JDBC calls are converted into Native API calls usually Native API calls are written in either C (or) C++.

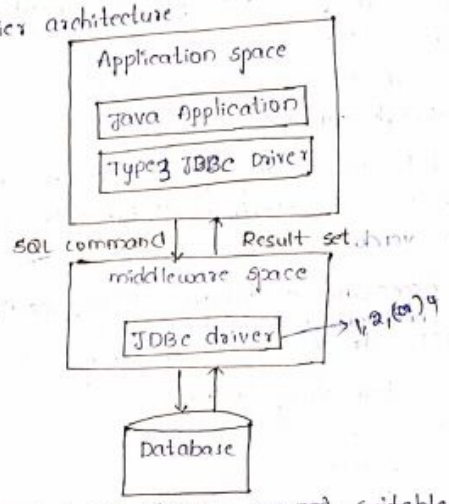
→ Type 2 drivers are provided by database vendors and it must be installed on client machine.

→ for every database there is a separate vendor specific native API driver is available.

→ compared to Type 1 it is fast because it eliminates the ODBC conversion.

Type 3 Driver :-

It is also referred as JBBC network driver, it follows 3-tier architecture




→ Type 1 and Type 2 drivers are not suitable for accessing a remote database

→ by using Type 3 driver all JDBC calls are converted into network calls supported by a suitable middleware server.

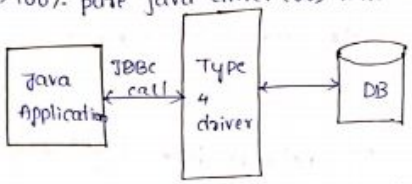
→ Then a server converts all network calls into suitable database calls using any JDBC driver.

→ In case of Type 3 driver there is no need to install database server on client system.

→ It is too expensive because we need to maintain an additional middle ware server.

Type 4 Driver:

→ 100% pure java driver (or) thin driver




→ In a Type 4 driver a JDBC calls directly converts into suitable database calls.

→ A Type 4 driver can communicate directly with any database through network connection

→ we don't need to install any special software on the client system. It is so flexible. All type4 drivers usually provided by the vendor itself.

procedure for accessing a database from java Application

To establish a connection with database use following steps

1. Registering a JDBC driver with Driver manager
2. Establish a connection with database
3. create a SQL statement
4. Execute a SQL statement
5. process the result
6. close the connection.

Step1: Registering a JDBC driver with Driver manager.

→ before using any driver we need to register a driver with driver manager

→ all the drivers contains some built-in class name.

→ TO register a driver we need to load a driver into client application.

→ To load a class name use Class.forName() method.

Syntax: Class.forName("Driver class name");

Ex:- Incase of Type1

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

The class name same for any database

In case of Type4 a class names will be differed based on database.

ex: class.forName('oracle.jdbc.driver.OracleDriver');

Step2: Establish a connection with database.

→ After loading a driver a driver manager as responsible to establish a connection with database.

To establish a connection use getConnection() method.

Syntax: Connection con =

DriverManager.getConnection(databaseurl, username, password);

username and password are optional

optional optional.

database url specifies the location of database

url format: jdbc:sub-protocol:datasource-name.

In case of

Type 1 - jdbc:odbc:it-b

Type 4 - jbbc:Oracle:thin:@host-name:port:db-name

oracle port number - 1521

oracle is 'XE'

Step3:- Creation of SQL statement:

→ To perform any operation on the database we need to create suitable sql statement.

→ Sql package contains 3 types of statement:

1) Statement
2) Prepared Statement

3) Callable Statement

Ex:- Statement st = con.createStatement();

PreparedStatement st = con.preparedStatement();

step4: execute an sql statement

To execute an sql statement use the following functions

1. executeQuery() → it is used only with select command it returns the result as result set contains various records.

ex:- ResultSet rs = st.executeQuery("select * from emp");

2) executeUpdate():

It is used with non-select commands like insert, delete, update (or) create. it returns an integer value i.e number of rows are affected.

Syntax: int x = st.executeUpdate("insert into emp values(1201, "xyz")");

iii) execute(): It is used with all sql commands. it retur is a boolean function

→ It returns true for select command otherwise it returns false.

Syntax: boolean x = st.execute(query);

→ To retrieve the result set use a built-in function getResultSet()

5) steps: process the results

To retrieve the results from result set use various built-in functions.

i) next():- It is a boolean function it returns true if the next record is available

ii) previous():

iii) first()

iv) last()

To access particular field use getter functions.

Syntax:- getInt("field-name");

To retrive the string information use getString()

Syntax: getString("field_name");

eg:- ResultSet rs = st.executeQuery("select * from emp");

```
while(rs.next())
  {
    s.o.p(rs.getInt("roll.no");
    s.o.p(rs.getString("name");
  }
```

steps: close the connection

To close the connection use a method close();

eg:- con.close();
st.close();
rs.close();

Write a JDBC program to create a table in the database

```
import java.sql.*;
class Create
{
public static void main(String args[])
 {
  try
  {
   Class.forName("oracle.jdbc.driver.OracleDriver");
   Connection con = DriverManager.getConnection("jdbc:orack:
        thin:@localhost:1521:XE","System","System");
   Statement st = con.createStatement();
   st.executeUpdate("create table st.it(rollno number(10),
                    name varchar2(20), age number(3)");
   S.o.p("Table created");
   con.close();
  }
  catch(Exception e)
   {
    e.printStackTrace();
   }
 }
```

2. write a JDBC program to retrieve the data from database

```
import java.sql.*;
class Select
{
public static void main(String args[])
{
try
{
class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:
Oracle:thin:@localhost:1521:XE", "System", "System");
Statement st = con.CreateStatement();
ResultSet rs=St.executeQuery("Select * from studs");
while(rs.next())
{
System.out.println(rs.getString(2)+" "+rs.getInt(1)+
" "+rs.getInt(3));
}
rs.close();
st.close();
con.close();
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
```

3. write a JDBC program to perform update and delete commands on database.

Result set :-

Resultset object contains set of records in the tabular form.

ResultSet rs = st.executeQuery("select * from emp");

To access table contents use the following methods

1. boolean next(): if the next record is available it return true and moves to next row from current position

Syntax: rs.next()

2. boolean previous() :- it returns true if the previous row is available and it also used to move to previous record.

3. boolean first() :- It moves to the first row from the current position

4. last() :- It is used to move to last row from current position.

5. absolute() :- It is used to moves to the particular row number from the beginning.

Syntax :- absolute(int row)

6. relative() :- it is used to move to specified record from the current position.

Syntax :- absolute relative(int row)

↳ +ve → moves in forward

-ve → moves in backward

7. getString() :

It is used to retrieve the data from column

Syntax :-

String getString(column_name);

String getString(col_num);

8. getInt() :- It is used to retrieve data from table.

Syntax :- String getInt(col_name);

String getInt(col_num);

## ResultSetMetaData:-

MetaData means data about data means we can get some additional information about the data

ResultSetMetaData contains information about ResultSet like tablename, no.of columns, .etc.

To retrieve the MetaData use a builtin function getMetaData().

Syntax:- ResultSetMetaData rmd = rs.getMetaData();

**Methods:-**

1. getColumnCount(): It returns the number of columns available in the result set (Integer)
2. getTableName():- It returns the name of a table (String)
3. getColumnName():- It returns the name of the column.

   Syntax:- getColumnName(int);
4. getColumnTypeName():- It returns the column Type name (or) datatype of a particular column (String)

   Syntax:- getColumnTypeName(int)

Ex:- write a JDBC program to display meta information about result set

```
import java.sql.*;
class Meta
{
 public static void main(String args[])
  {
    try
     {
       Class.forName("oracle.jdbc.driver.oracle
                                    Driver");
   Connection con = DriverManager.getConnection
          ("jdbc:oracle:thin:@localhost:1521:
            XE","system","system");
   Statement st = con.createStatement();
   ResultSet rs = st.executeQuery("select * from emp");
   ResultSetMetaData rm = rs.getMetaData();
    int x = rm.getColumnCount();
    System.out.println("number of columns"+x);
    for(i=1;i<x;i++)
     {
       S.O.P(rm.getColumnName(i)+" "+rm.getColumn
             TypeName(i));
     }
    S.O.P("table name"+rm.getTableName());
    con.close();
  }
  catch(Exception e)
  {
    e.printStackTrace();
  }
 }
}
```

**Types of statements:-**

In JDBC we have 3 types of statements

i) Statement
ii) prepared Statement
iii) callable Statement

Statement: It is used to create static Queries. if we execute any no.of times the same data will be consider. To create a sql statement using statement class use a function CreateStatement()

Syntax:- Statement st = con.createStatement();

Statement contains 3 method

1. execute()
2. executeUpdate()

executeQuery()

2. preparedStatement() :-

It is used to prepare a sql query but the data will be provided at run time.

To prepare a statement use a built-in function prepareStatement()

Syntax :-

```
preparedStatement ps = con.prepareStatement("query");
```

```
preparedStatement ps = con.prepareStatement("insert into
        st-nag values(?,?)");
```

↳ place holders.

The values of place holder will able assigned at run time.

→ To assign some values to place holders use setter methods.

Syntax :- ~~set-xxx~~ setXXX()

```
setInt(placeholder_pos, value);
setString(placeholder_pos, value);
```

→ To execute a statement use execute update. executeUpdate()-

Write a JDBC program to perform insertion using preparedStatement

```
import java.sql.*;
class Insert
{
  psvm(String args[])
  {
    try
    {
      Class.forName("oracle.jdbc.driver.OracleDriver");
      Connection con = DriverManager.getConnection
("oracle:jdbc:oracle:thin:@localhost:1521:xe", "system",
"system");
      preparedStatement ps = con.prepareStatement("insert
        into st-nag values(?,?)");
      Scanner s = new Scanner(System.in);
      S.O.p("enter roll number");
      String rn = s.next();
      S.O.p("ehter name");
      String n = s.next();
      ps.setString(1,rn);
      ps.setString(2,n);
      int i = ps.executeUpdate();
      con.close(); S.O.p(i+" record inserted");
    }
    catch(Exception e)
    {
      S.O.p(e);
    }
  }
}
```

3. Callable Statements :

Callable statements are useful for executing stored procedures and functions available in the Database.

→ A procedure and function can contains some statement is used to perform some specific task.

The difference b/w procedure & function is a procedure cann-ot return any value but function can return some value.

Procedure creation:

Syntax :

```
create (or) replace procedure procedure_name(x in number, y in
        number, z out number) is
```

```
begin
x:= x+y;
end;
/
```

function creation:

Syntax: create (or) replace function func_name (x in number, y in number) return number is

```
begin
return x+y;
end;
/
```

→ To execute procedures (or) functions use a function prepareCall();

Syntax:- CallableStatement cs = con.prepareCall("{call nag(?,?,?)}");

CallableStatement cs = con.prepareCall("{?=call nag-f(?,?)}");
↳ for function.

→ assign some data for placeholders using setter methods. setXXX()

→ register output parameter to know which type of data stored in the place holder. to register use a builtin function
Syntax:- cs.registerOutparameter(placeholdernumber, data-type)
example: cs.registerOutparameter(3,Types.INTEGER)

Types.DOUBLE
Types.VARCHAR
for long strings ← Types.LONGVARCHAR

→ execute an sql statement using execute() method

→ To retrieve the output use getter methods.
Syntax: getInt(placehold-number);

```
import java.sql.*;
class callpro
{
  psvm(String args[])
  {
    try
    {
      class.forName("oracle.jdbc.driver.OracleDriver");
  Connection con = DriverManager.getConnection("jdbc:
    oracle:thin:@localhost:1521:xE", "System", "system");
```

```
CallableStatement cs = con.prepareCall("{call nag(?,?,?)}");
Scanner s = new Scanner(System.In);
 S.o.p("enter num1");
 int a = s.nextInt();
 S.o.p("enter num2");
 int b = s.nextInt();
 cs.registerOutparameter(3,Types.INTEGER);
     cs.setInt(1,a);
     cs.setInt(2,b);
     cs.execute();
    S.o.p("sum is"+ cs.getInt(3));
    con.close();
   }
   catch(Exception e)
   {
     S.o.p(e);
   }
}
```

UNIT-5
part-2
AJAX

10/02/19

AJAX stands for Advanced Javascript and XML. (Asynchronous)

→ AJAX is useful for developing more interactive webbased applications.

→ AJAX is the combination of various technologies like HTML, CSS, and Javascript.

→ In case of AJAX, we can retrieve the data from server asynchronously without reloading a webpage.

Applications of AJAX

1. AJAX technology is widely used in various applications like
* youtube, twitter, facebook, Google map, instagram etc.

→ To apply asynchronous request AJAX follows "XMLHttp Request" object.

→ In previous technologies like javascript, PHP we are using Synchronous communication between client and server.

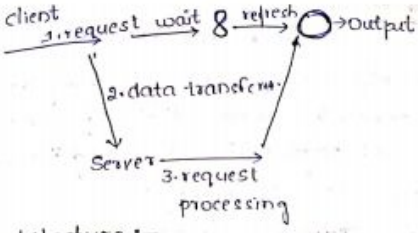
→ It follows following steps

1) client sends a request to the server
2) A server receives the request and process, at the time of processing client is in wait state.
3) Send response to client
4) receives the response and reload the webpage.

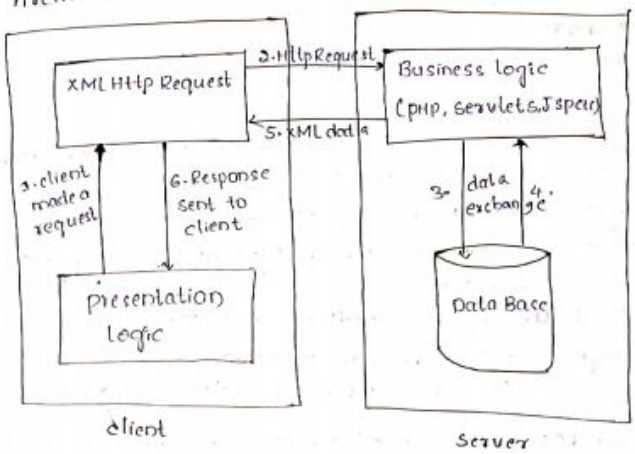
→ AJAX follows asynchronous communication it uses "XMLHttpRequest" object to transfer the data in asynchronous manner.

It performs ~~ver~~ 3 operations

i) client sends a request to the server at backend
ii) Server process the request
iii) receives the response and displaying without reloading a webpage.




AJAX Architecture :-




Steps:

1) client made a request to XMLHttpRequest object
2) Object send a HttpRequest to ~~by~~ a server at a backend
3)&4) data exchange can be done b/w database and server
5) client receives a respond from server in the form of XML data
6) object sends a request to client without reloading the webpage.

## XMLHttpRequest Object:

→ All modern browsers supports the XMLHttpRequest object

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

→ To create an object use following syntax

```
var xhttp = new XMLHttpRequest();
```

→ Methods :-

1. open()
2. send()

1. open() :- It is used to establish a communication(or) connection b/w client and server

Syntax: open(method, url, async);

method:- it represents which type of Http request
GET, POST

Url :- Url of the specific script file.

async :- it is a boolean variable
true - Asynchronous false - Synchronous.

2. send() : It is used to send a request to a server

eg:- xhttp.open(GET,"demo_post.php",true);
xhttp.send();

xhttp.open(post,"demo_post.php",true);
xhttp.send("fname="sushitha");

→ To receive the reponse we need to use various datamembers available in XMLHttpRequest.

→ To describe status of request object use readyState property it contains some integer value.

if (readyState == 0) → request not initialized
== 1 → server connection established
== 2 → request received
== 3 → processing request
== 4 → request finished and response is ready

responseText :- it returns the response in the text form.
responseXML :- it returns the response in XML document form.
onreadystatechange :- defines a function to be called when the readyState property changes.

```
<html>
<head>
<script>
function showUser(str)
{
  if(str == "")
  {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  else
  {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange= function()
    {
      if(this.readyState == 4)
      {
        document.getElementById("txtHint").innerHTML= this.response
                                                      Text;
      }
    };
    xmlhttp.open("GET","getuser.php?q="+str,true);
    xmlhttp.send();
  }
}
</script>
</head>
<body>
<form>
enter a name : <input type="text" onkeyup="showUser(this.value)">
</form>
<br>
<div id="txtHint"></div>
</body>
</html>
```

20/04/...

Unit-5

# Webservers & servelets

server :

A server is a program (or) device that provides services (or) fuctionalities for other programs (or) devices is called client

To develop web based applications a web browser will acts as client.

Types of servers :

1) file server :-

It is a program (or) a storage device dedicated to storing some files.

we can access those files from any client system with the help of ftp protocol.

eg:- microsoft sharepoint server

2) mail server :-

To receive any mails (or) To send any mails we require a mail server.

There are 2 types of mail servers

1) incoming mail server - eg:- POP (post office protocol)

2) Outgoing mail server - eg:- SMTP (simple mail transfer protocol)

3) print server :-

It is a system . it is used to manage one (or) more printers connected through network.

A print server accepts the request from client and assigned to suitable printer.

eg:- wireless print server → cisco q company

4) database server :- It provides various services related to datastorage and data manipulation

eg:- SQL server provided by microsoft

dv2 → IBM company

5) proxy server :-

It is a server and it will acts as a intermediate layer b/w client and other servers.

→ whenever a client sends a request a proxy receives the request and transfer to suitable server.

eg:- free proxy, WinGate

6) web server :-

It is used to run (or) develop web applications on the client system.

→ Some of the web servers are WAMP, APACHE Tomcat Server, internet information server (IIS), Google web server, Netware, BEA Weblogic server.

→ whenever client made a request á HTTP protocol transfers the request to webserver.

7) Application server :-

It is a server and it will provide all facilities for creating and maintaining web applications

→ It provides all services required for 3-tier architecture.

eg:- JBOSS, Glass fisb

Apache Tom cat web server :-

Tomcat web server is provided by apache software foundation. it is fully written in java it is used to process various web applications developed in java languages like servlets and JSP.

the current version of Tomcat is Tomcat 9 released in 2018

Installation process :

1) To install any Tomcat server first we need to install JDK and that must require a support of JSDK (Java Software Development Kit and JRE (Java Runtime Environment.

→ welcome to the setup wizard click on next button

2) Accept the license Aggreement click on I Agree

3) choose components (or) Features of Tomcat

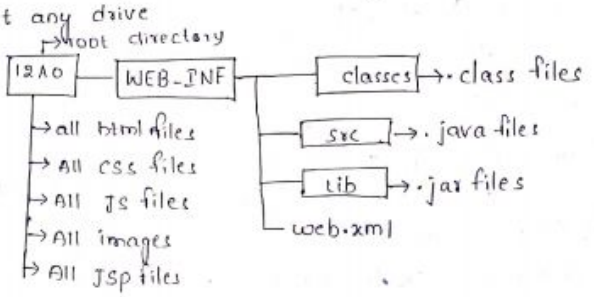
4) configure the Tomcat by assigning http port number and provide username and password

5) Select JVM path location c/programFiles/~~Apache~~/java/jre

6) choose installation location c/programFiles/Apachesoftware foundation/Tomcat 7.0

→ To run any web applications on Tomcat server the user must follow some directry structure.

→ Select any drive




Java Servlets :-

→ A servlet is a simple java program is used to add dynamic nature to static webpage.

→ It is a server side scripting languages.

→ To process a servlet we require a web server i.e Tomcat Server.

→ To develop servelet application we need to follow Servlet API

Common gateway Interface (CGI):-

→ Before servlets programmer used a standard method for developing dynamic web applications i.e, CGI.

→ In case of CGI, whenever client made a request then a new process will be created.

→ Entire CGI technology is developed in C or C++.

→ If no.of requests are increases we require more memory and more time to respond.

→ It is a platform dependent technology.

→ In case of servlets whenever a client made a request a new thread will be created instead of separate process

→ It is fully written in java so it supports platform independent nature.

→ It takes less time to respond a request by a user

→ To create any servlet application we need to follow require a servlet API

1. javax.servlet package
2. javax.servlet.http package

Creation of a Servlet :-

To create a servlet we need to follow 2 techniques.

1. extending GenericServlet class
2. extending HttpServlet class

In case of generic servlet is a protocol independent it supports all transfer protocol HTTP, SMTP, FTP etc.

→ But HTTP servlet class is a protocol dependent it supports only HTTP protocol.

So HTTP servlet is a subclass of GenericServlet.

The packages required for generic servlet is
javax.servlet

The packages is required for HTTP servlet is jav
javax.servlet.http.

Servlet life cycle :-

→ During the lifecycle of a servlet a servlet will be available at 3 stages.

i) initialization of a servlet (init())
ii) request process (service())
iii) destroying a servlet (destroy())

i) init() :-

It is used to initialize the servlet parameters it will be called only once when the servlet is loaded into main memory. It is used to initialize various variables, constants various database resources.

Syntax:- public void init() throws ServletException
{
}

ii) service() :-

It is like a main method. it is used to handle all the request by the user.

→ It contains a set of java statement is used to perform some specific task.

→ In the user servlet we must override the service(). but in case of HttpServlet we can select the method based on http requests [GET, POST]

1) doGet()
2) doPost()
} — service() in HttpServlet

→ if HTTP request is GET then use doGet()
→ if HTTP request is POST then use doPost()

Syntax :- public void service(ServletRequest req, ServletResponse res)
{ throws ServletException, IOException
}

```
public void doGet(HttpServletRequest req, HttpServletResponse
                  res) throws ServletException, IOException
```

iii> destroy() :-

It is used to destroy servlet from the main memory. we can place some statements like file closing, database connection close. It also called only once during its lifetime

syntax: public void destroy()
{
}

Skeleton of Servlet :-

```
import java.io.*;
import javax.servlet.*;
public class Architecture extends GenericServlet
{
  public void init() throws ServletException
  {
    // initialization of all parameters
  }
  public void service(ServletRequest req, ServletResponse res)
      throws ServletException, IOException
  {
    // Request nothing processing
  }
  public void destroy()
  {
    // do nothing
  }
}
```

* Write a servlet program to print welcome message

```
import javax.servlet.*;
import java.io.*;
public class Hello extends GenericServlet
{
  public void service(ServletRequest req, ServletResponse res)
      throws ServletException, IOException
  {
    PrintWriter out = req.getWriter();
    out.println("<h1> welcome to servlet programming </h1>");
  }
}
```

web.xml

```
<?xml version="1.0" ?>
<web-app>
 <servlet>
 <servlet-name> First </servlet-name>
 <servlet-class> Hello </servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name> First </servlet-name>
 <url-pattern> /it </url-pattern>
 </servlet mapping>
 </web-app>
```

→ web.xml will be ~~display~~ used as diployment descriptor for identifying various servlet file in servlet container

→ To recognize particular servlet file we need to specify url pattern for every class.

* Write a servlet program to print current system date & time.

```
import javax.servlet.*;
import java.util.*;
import java.io.*;
public class DateDemo extends GenericServlet
{
```

```
public void service(ServletRequest req, ServletResponse
        res) throws ServletException, IOException
{
  PrintWriter out = rs.getWriter();
  Date ob = new Date();
    out.println("Date is" +ob);
}
}
```

* Reading Servlet parameters (or) Retriving data from form:

→ To read data from HTML form use a built in function getparameter() available in ServletRequest or HttpServlet Request.

Syn:- String getparameter("s");

eg:- String s = req.getparameter("ta");

To transfer the contents from HTML to servlet use action attribute in <form>

Syntax:- <form action = "url-pattern">

<form action = "./_____">

Write a servlet program to retrieve name from form and display a message in the following form welcome to

```
<html>
<head> </head>
<body bgcolor = "red">
<form action = "act">
                  └→ Url pattern
enter a name: <input type = "text" name = "ta">
  <input type = "Submit" value = "submit">
  </form>
  </body>
  </html>
```

Servlet:-

```
import javax.servlet.*;
import java.io.*;
import java.util.*;
public class Retrieve extends GenericServlet
{
public void service (ServletRequest req, ServletResponse rs)
    throws ServletException, IOException
{
 PrintWriter out = rs.getWriter();
 String s = req.getparameter("ta");
  out.println("welcome to" +s);
}
}
```

* → WASp to find factorial of given number
→ WASp for login validation to check username & password
→ WASp to insert form contents into database.
→ WASp to implement change password form.

Reading Initialisation parameters & passing the parameters to Servlet:

To read initialisation parameters from the server we require two built-in classes

1. ServletConfig
2. ServletContext

1. ServletConfig :- It is used to provide complete description of a web.xml file.

A config object will be created at the time of a servlet is loaded into main memory

By using ServletConfig we can able to retrieve the parameters passed to a single servlet

To pass parameters we require a tag.

<init-param> :-
we need to write a tag in web.xml.
~~<init-param>~~ <init-program> tag must be included in <servlet> tag.
To send parameters use the following form

```
<init-param>
<param-name> branch </param-name>
<param-value> it </param-value>
:
</init-param>
```

Methods of ServletConfig:-
To retrieve ServletConfig information use a meth
i) getServletConfig();
Syntax:- ServletConfig c = getServletConfig();
ii) getInitParameter(): It is used to retrieve the value of a particular parameter
Syntax:- String s = c.getInitParameter("param-name");
eg:- String s = c.getInitParameter("branch");
iii) getInitParameterNames(): It returns all the initialized parameter names.
Syntax:-
Enumeration e = c.getInitParameterNames()
iv) getServletName():-
It returns the particular servlet name
Write a servlet program to retrieve parameters using ServletConfig

```
import java.servlet.*;
import java.io.*;
public class ConfigDemo extends GenericServlet
{
public void service(ServletRequest req, ServletResponse
res) throws ServletException, IOException
{ PrintWriter out = res.getWriter();
ServletConfig config = getServletConfig();
String s = config.getInitParameter("date");
out.print("date is:"+s);
String n = config.getServletName();
out.println("name is:"+n);
}
}
```

web.xml

```
<web-app>
<servlet>
<servlet-name> four </servlet-name>
<servlet-class> ConfigDemo </servlet-class>
<init-param>
<param-name> date </param-name>
<param-value> 22:03:22 </param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name> four </servlet-name>
<url-pattern> /config </url-pattern>
</servlet-mapping>
</web-app>
```

2) ServletContext:-
In case of ServletConflict we can able to send Parameters to single servlet but to pass same parameter to all the servlet in my application then use servlet Context.
ServletContext is an object contains description of web.xml file. To send parameters through servlet context use a tag <context-param> in <web-app> tag.

Syntax:- <web-app>
<context-param>
<param-name> —— </param-name>
<param-value> —— </param-value>
</context-param>
<servlet>
|
</servlet>
</web-app>

→ To retrieve servlet context use a method getServletContext()

Syntax:- ServletContext ob = getServletContext();

Methods of ServletContext class

1) getInitParameter():- It is used to retrieve the parameter value based on parameter name.

String s = ob.getInitParameter("name");

2) getInitParameterNames():- It returns all the parameter names.

3) setAttribute():- It is used to store information in the Servlet Context.

Syntax:- setAttribute(name, value);

4) getAttribute():- It is used to retrieve the information from ServletContext.

Syntax:- getAttribute(name);

Write a Servlet program to read parameters using ServletContext.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class context extends HttpServlet
{
public void doGet(HttpServletRequest request, HttpServlet
        Response response) throws ServletException,
                        IOException
{ PrintWriter out = response.getWriter();
  ServletContext con = getServletContext();
  String d = con.getInitParameter("branch");
   out.print("branch is:"+d);
     out.close();
}
}
```

web.xml

```
<servlet-mapping>
<servlet-name>FS</servlet-name>
<url-pattern>/fs</url-pattern>
</servlet-mapping>
  :
  :
<context-param>
<param-name>branch</param-name>
<param-value>it</param-value>
</context-param>
```

Session tracking (or) Session management

The time period btn login and logout will be treated as session. During that period, a user may visit multiple web pages.

→ Http protocol is used to maintain interaction b/w client & server. But Http protocol is a stateless protocol means it is not going to maintain state(data) of particular user. Because Http treats every request as a new request

→ So, to maintain state of particular user we need to maintain session information using session Tracking

→ To manage sessions 4 techniques are available

1. cookies
2. HttpSession
3. Hidden form fields
4. URL rewritting

1. cookies :- A cookie is a small piece of information stored on client system or web browser. Every cookie contains 2 things

i) cookie Name

ii) cookie value

→ cookies are of two types < Non-persistent cookie / persistent cookie.

Non-persistent cookie :- Once the browser is closed the data will be lost

eg:- banking

persistent cookie :- Once the browser is closed the data will not be lost

eg:- fb, Gmail.

Advantages:

→ It is easy to maintain

→ Cookies information is stored at client system.

Drawbacks:

→ It is allowed to store only textual information and max limit of 4kB data

→ If we disable the cookies in web browser it won't work properly.

Creation of cookie :-

To create a cookie use a built-in class Cookie in the javax.servlet.http package

constructors :-

i) Cookie() :- It is used to create a cookie

ii) Cookie(String name, String value) :- It is used to cookie creat a cookie with specified name & value.

eg:- Cookie c = new Cookie("nam", "hi");

Methods :-

i) setName() :- It is used to assign (or) modify name to a cookie

Syntax:- SetName(String name)

Cookie c = new Cookie();
c.setName("branch");

ii) setValue() :- It is used to assign (or) modify value of a cookie.

eg:- c.setValue("it");

iii) getName() :- It is used to return name of particular cookie return type string

Syntax:- c.getName();

iv) getValue() :- It is used to return value of particular cookie.

→ To store information on client system use a method addCookie() available in the HttpServletResponse class.

eg:- res.addCookie()

→ To retrieve cookie information from user browser use a method getCookies() available in HttpServletRequest class

Syntax:- Cookie c[] = req.getCookies();

write a servlet program to implement cookies

cook1.html

```
<html>
<body>
<form action="c1" method="post">
Name : <input type="text" name="name"/> <br>
<input type="submit" value="go" />
</form>
</body>
</html>
```

cook1.html


cook2.html



cook.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class cook1 extends HttpServlet
{
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
```

```
    {
     PrintWriter out = res.getWriter();
       String n = req.getparameter("Name");
        Cookie ck = new Cookie("uname",n);
          res.addCookie(ck);
          res.SendRedirect("cook2.html");
    }
  }
```

cook2.html

```
<html>
<body>
<form action="c2" method="post">
Enter branch: <input type="text" name="branch"/> <br>
 <input type="submit" value="go"/>
</form>
</body>
</html>
```

cook2.java

```
    PrintWriter out = res.getWriter();
    String n = getreq.getparameter("branch");
     out.println("Branch is"+n);
     Cookie c[] = req.getCookies();
      out.println("Cookie Name:" + c[0].getName());
      out.println("Cookie value:" + c[0].getValue());
```

Note:

SendRedirect(): It is used to move the control to the next webpage. it is available in ServletResponse and HttpServletResponse

Syntax:- res.SendRedirect("url");

*WASP to valid credentials.
*WASP to insert form contents into database.
*WASP to implement cookies
*WASP to implement session management concept.

→Http session:-

It is a class available in javax.servlet.http package

It is used to store entire session information of particular client.

→All the session objects is going to managed by server.

→Whenever a session starts automatically a server assigns a unique session id.

→To create a session use a class HttpSession

→To create a session use the following constructors.

To create a session use getSession() available in HttpServlet request class

i> HttpSession s = request.getSession();

it returns a session. if the session is already exist then it returns existing one otherwise it creates new one.

ii>HttpSession s = request.getSession(true);

It always returns a new session.

iii> HttpSession s= request.getSession(false);

→it returns a pre-existing one if it is not available it creates a new session.

Methods:-

1> getId(): It returns a session Id.

Syntax: String getId()

2> isNew():- It returns true if the session is new Otherwise it returns false.

iii) getCreationTime(): It returns the time when the session is created. it returns in milliseconds.

iv) getLastAccessedTime():- returns the last time the client sent a request to a session.

v) invalidate(): It is used to destroy the session

vi) SetMaxInactiveInterval():- It specifies the time limit in seconds. if a particular webpage inactive for a specified time automatically a destry from the memory

Syntax:- setMaxInactive_Interval(int interval)
↳ time

vii) getMaxInactiveInterval():- it returns the time interval in second.

WASP to implement or manage sessions using HttpSession.

```
<html>
<head>
</head>
<body>
<form action="sess1">
```

SetAttribute(): It is used to store information in the session

Syntax:-setAttribute(name, value)

getAttribute() :-It is used to retrieve the information from session

Syntax:-getAttribute(name)
↳ it return object type data

Program:-

sess.html

```
<html>
<head></head>
<body>
<form action="sess1">
Name: <input type="text" name="userName"/><br/>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Session1.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Session1 extends HttpServlet
{
public void doGet (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    PrintWriter out = res.getWriter();
    String n= req.getparameter("userName");
    HttpSession s= req.getSession();
        s.setAttribute("uname", n);
        res.sendRedirect("cook2.html");
    }
}
```

cook2.html

```
<html>
<body>
<form action="sess2" method="post">
```

```
Enter branch:<input type='text
<input type="submit" value="submit">
</form>
</body>
</html>
```

Session2.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Session2 extends HttpServlet
{
 public void doPost(HttpServletRequest req, HttpServletResponse res)
     throws ServletException, IOException
 {
   PrintWriter out = res.getWriter();
   HttpSession s = req.getSession(false);
   String b = req.getParameter("branch");
    out.println("branch"+b);
   String n = (String)s.getAttribute("uname");
      out.println("Hello"+n);
       out.close();
  }
 }
```

→ WASP to display a msg welcome for 1st visit and display a msg "thank you for visit again" for 2nd visit onwards.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class Visit extends HttpServlet
{
public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
{
 PrintWriter out = res.getWriter();
 HttpSession s = req.getSession();
 Integer accesscount = (Integer)s.getAttribute("xyz");
  if(accesscount == null)
  {
   accessCount = new Integer(1);
    out.println("welcome");
  }
 else
 {out.println("thank you for visit again");
 accessCount = new Integer(accessCount.intValue()+1);
 }
 s.q s.setAttribute("xyz",accessCount);
  s.setMaxInactiveInterval(60);
  out.println("session id"+s.getId());
  out.println("session creation time:"+new Date(s.getCreation
                    Time()));
   out.println("NO of visits made by you:"+accessCount);
  }
}
```

JavaBeans:-

→ A java bean is a software component that has been designed to reuse in different applications.

→ There is no restriction on a capability of a bean it may perform simple operation like spell check, (or) complex operation like weather forecasting

→ A bean may be visible to the user like a button, checkbox, etc

→ Sometimes a bean may be invisible to user like grammar check.

→ After creating a bean, a bean must be placed on the Bean Development kit (BDK) to reuse in another application

→ Some of the tools are java BDK, Net Beans, eclipse,

→ It supports platform independent nature because it is fully written in java.

→ To create a java bean we need to follow some rules

1) Every java bean must contain a zero-argument (empty) constructor.

2) A bean contains some properties (data members) and methods.

3) A bean contains only setter and getter methods to access properties.

4) Every java bean class must implement a Serializable interface

A Serializable interface not contains any methods just it is used to maintain (or) persist state of a particular object.

```
import java.io.*;
public class Employee implements Serializable
{
  int salary;
  String name;
  Employee()
  {
   salary=10000;
   name="ruchitha";
  }
 void setSalary(int s)
 {
  salary=s;
 }
 void setName(String n)
 {
  name=n;
 }
 int getSalary()
 {
  return salary;
 }
 String getName()
 {
  return name;
 }
}
```

→ Bean properties:-

In JavaBeans properties are classified into 4 types

1. Simple properties
2. Indexed properties
3. Bound properties
4. Constrained properties

Simple property:-

A simple property of a Bean is able to store only a single value like age, name, salary etc.

To design methods for simple properties

eg:- int age

```
void setAge(int age)
{
}
int getAge()
```

Indexed property:-

An Indexed property of a Bean is able to store multiple values.

eg:- int marks

```
void setMarks(int a[])
{
  for(int i=0; i<a.length; i++)
   {
     Marks[i]=a[i];
   }
}
int [ ] .getMarks( ) →it returns entire array.
 {
  return marks;
 }
```

In case of arrays, we can design the getter and setter methods based on its index position

eg:- 
```
void setMarks(int index, int value)
{
  marks[index] = value;
}
```

To retrieve the value based on index use

```
int getMarks(int index)
  {
    return marks[index];
  }
```

Bound properties:-

→ if any object want to modify the value of a property then propertyChangeEvent will occur.

→ To handle propertychangeEvents we require Propertychange listener.

→ To receive property change notifications a bean object must be register using addPropertyChangeListener( )

→ A propertychangeListener contain only one method propertyChanged( )

→ whenever a property value changes it will informed to all the registered Listeners

→ A listener receives the notification and accept the change.

4- Constrained property:-

→ Similar to bound property but the listener has a capability of veto the change

→ whenever a user trying to change a property value propertychangeEvent will occur to handle the event we need to implement vetoablechangeListener

→ It contains only one method vetoablechange( )

→ A vetoableChangeListener perform 3 operations

1) A listener has to register with a event
2) received a notification
3) A listener will accept (or) veto the change
   ↳ reject

→ To veto the change a Listener will through propertyVetoException

Syntax:- throw new propertyVetoException("msg");

→ If any listener vetos the change the old value will be reverted back.

UNIT - IV

18/03/19

## Java Server pages

JSP is a server side scripting language it is used to add dynamic content to a static webpage.

→ A Jsp is a extension of servlet. it has so many advantages

→ In the java servlets the presentation logic embedded in the business logic (java code)

out.println("<h1>hello</h1>");

if we done any modification at presentation logic we need to recompile and redeploy in the server.

To overcome that in the Jsp the business logic embedded in the presentation logic. if we done any modification it will not recompile.

→ unlike servlets a Jsp contains various implicit objects to perform some operations.

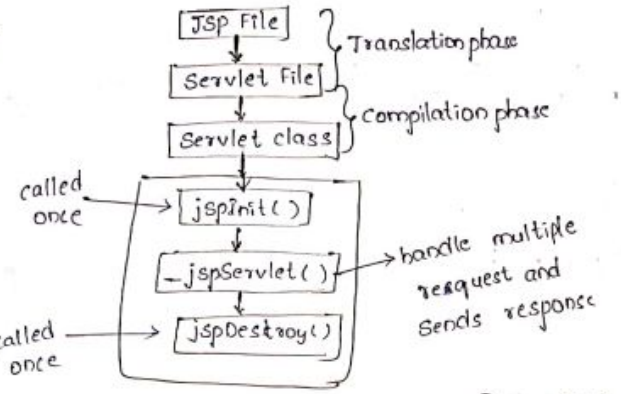
→ All Jsp files must be processed by using a web server i.e tomcat server.

→ life cycle of a Jsp page:-

During the lifecycle of a Jsp page it will be available at various stages

1) Translation phase
2) Compilation phase
3) loading a class file into memory
4) calling jspInit() method
5) calli executeJspService() method → request processing
6) call jspDestroy() method

## Jsp life cycle




Anatomy of Jsp page (or) components of Jsp page

→ Every Jsp page contains two components

1) tags required for presentation logic (html)
2) the tags required for develop jsp code (jsp elements)

→ Jsp elements are classified into 3 types.

1. Scripting elements
2. Action elements
3. Directive elements.

8, Jsp Scripting elements :-

→ Scripting elements are useful for to insert (or) add java code in Jsp page

1. Scriplet
2. Expression tag
3. Declaration tag

Scriplet :-

It is used to embed a java code in the Jsp progra

Syntax :- <%

code

%>

→ To a Jsp page we can include any number of scriplet tag.

* Write a Jsp program to print a simple message like "hello world".

```
<html>
<body>
<%
out.println("hi welcome all");
%>
<h1>
<%
out.println("welcome");
%>
</h1>
</body>
</html>
```

Expression tag:-

It is used to evaluate the expression. after evaluation it is directly passed to out.println

Syntax:- `<%= exp %>`

eg:- `<%= 3+4 %>`

Declaration tag:-

It will be act as declaration section. it may include variable declaration (or) method declaration.

Syntax:-
```
<%!
   declarations
%>
```

To import any packages use following syntax

`<%@ page import="java.util.* "%>`

Jsp Action Elements:-

Jsp action elements are useful for to perform specific task in Jsp page.

1. <jsp:include>:-

It is used to include the contents of another Jsp file (or) HTML file in the current page.

→ It includes the contents at run time means during translation it cannot include any content.

Syntax:- `<jsp:include page="filename/url" />`

eg:- `<jsp:include page="hello.jsp" />`

eg:- first
hello.html
```
<html>
<body>
<h2>welcome</h2>
</body>
</html>
```

first.jsp
```
<html>
<body>
hi how are you
<jsp:include page="first.html" />
hello
</body>
</html>
```

o/p:- hi how are you
welcome
hello

→ The advantage of <jsp:include> is code reusability.

2. <jsp:param>:-

It is used to pass parameters to another file. it can be used with either <jsp:include> or <jsp:forward>

Syntax:- `<jsp:param name=" " value=" "/>`

eg:- <jsp:param name="n" value="5" />
<jsp:param name="branch" value="it" />

*Write

cal.jsp

```
<html>
<body>
<jsp:include page="fact.jsp">
<jsp:param name="n" value="5" />
</jsp:include>
</body>
</html>
```

fact.jsp

```
<html>
<body>
<%
int x = Integer.parseInt(request.getParameter("n"));
  int fact = 1;
  for(int i=1 ; i<=x ; i++)
    {
     fact = fact*i ;
    }
%>
</body>
</html>
```

3. <jsp:forward> :-

It is used to forward the control to the next webpage

Syntax:- <jsp:forward page="file_name" />
url

→ It is similar to sendRedirect which is available

1. <jsp:plugin> :- It is used to include some components like Bean (or) Applet into a Jsp page.

Syntax :- <jsp:plugin type="bean/applet" code="class-file"
align="left/middle/Right" width="inpxls" height="inp
/>

→ if the specified plugin is not available then to display some alternate text use <jsp:fallback>

Syntax:- <jsp:fallback>
msg
</jsp:fallback>

eg:- <jsp:plugin type="applet" code="P1.Sample" width=500
height=500>
<jsp:fallback>
Applet not loaded
</jsp:fallback>
</jsp:plugin>

5. <jsp:useBean> :-

It is used to access various properties (or) methods from a Bean file (or) java file.

Syntax:-

<jsp:useBean> id="object_instance" class="class_name"
Scope="page/request/session/application" />

In Jsp scopes are classified into 4 types

1. page :- It is default scope means it can accessible for particular webpage
2. request :- Scope is limited for a particular single request.
3. Session :- Scope is limited to entire Session.
4. application :- Scope is limited to particular object is visible to every webpage available in application.

...preter it will check whether the object-instance is created (or) not. If it is not created it will create a new one.

→ if it is created it will check the scope.

→ if scope is page then it will creates a new one otherwise it will access from old instance.

eg:- Write one java program to find cube of a number

sample.java

```
package p1;
public class Sample
{
   public cube (int n)
    {
      return nxnxn;
    }
}
```

WEB-INF → classes → p1 → Sample.class

bean.jsp

```
<html>
<body>
<jsp:useBean id="ob" class="p1.sample" />
<%
   int x = ob.cube(7);
    out.println("x is "+x);
%>
</body>
</html>
```

8. <jsp:get property> :-

Student.java

```
package p2;
public class Student
{  String name;
   public Student()
    { name = "abc";
    }
   public void setName(String n)
    { name=n;
    }
   public String getName()
    {
      return name;
    }
}
```

bean1.jsp

```
<html>
<body>
<jsp:useBean id="ob" class="p2.Student"/>
<%
  String n = ob.getName();
      out.print(n);
%>
<br>
<%
   ob.setName("xyz");
    n= ob.getName();
     out.print(n);
%>
</body>
</html>
```

of Bean properties and display that

Syntax:- <jsp:getProperty name="object_name"
property="property-name" />

4. <jsp:setProperty>:-

It is used to assign a new value to a Bean property.

Syntax:- <jsp:setProperty name="object_name"
property="property-name" value="value" />

eg:-

Employee.java

```
package p1;
public class Employee
{
  int salary;
  public Employee
  {
    salary=10000;
  }
  public void setSalary(int n)
  {
    salary=n;
  }
  public int getSalary()
  {
    return salary
  }
}
```

bean2.java.jsp:-

```
<html>
<body>
<jsp:useBean id="ob" class="p1.Employee" />
<jsp:getProperty name="ob" property="salary" />
<jsp:setProperty name="ob" property="salary" value=
"25000" />
<jsp:getProperty name="ob" property="salary" />
</jsp:useBean>
</body>
</html>
```

Directive Elements :-

→ directive elements are useful for providing some special instruction to a web container for processing Jsp page.

→ In Jsp, to declare a directive tag use the following form

<%@ and %>

Syntax :- <%@ directive-name attribute="value" %>

→ To specify multiple attributes use comma as separator.

In Jsp directives are classified into 3 types :-

1) include directive
2) page directive
3) Taglib directive

1) include directive :-

It is used to include the contents of another file in the current webpage.

It is similar to <jsp:include> action element.

Incase of include directive element it include the contents at the time of translation itself. But incase of include action element the content include at the time of processing the request.

Syntax:-

<%@ include file="url" %>

eg:- <%@ include file="hello.jsp" %>

2. page directive:-

It is used to specify some properties for the current jsp page. like importing packages, selecting a language etc.

Syntax:- <%@ page attribute="value" %>

attributes of page directive:-

1> import:-

It is used to import packages (or) classes into current jsp page

Syntax:-

<%@ page import="package_name" %>

eg:- <%@ page import="java.sql.*" %>

<%@ page import="java.util.Date" %>

<%@ page import="java.sql.*, java.util.*, java.io.*" %>

2> session:-

It is used to specify whether a particular page is participating in the session (or) not. The possible values are true or false

↓ default

Syntax:-

<%@ page session="true/false" %>

3. errorpage:-

It is used to specify the jsp file that is used for handling exceptions.

Syntax:- <%@ page errorPage="file_name" %>

eg:- <%@ page errorPage="handle.jsp" %>

4. isErrorPage:-

It is used to check whether a particular jsp page will act as exception handler (or) not. The possible values are true (or) false

↳ default

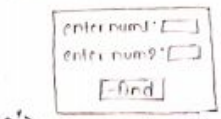
syntax:- <%@ page isErrorPage="yes/no" %>

true/false

eg:- <%@ page isErrorpage="true" %>

Write a jsp program to handle Arithmetic Exception:

f1.html




```
<html>
<body>
<form action="f2.jsp">
enter num1 : <input type="text" value="t1">
enter num2 : <input type="text" value="t2">
<input type="button" value="find">
</form>
</body>
</html>
```

f2.jsp

```
<html>
<body>
<%@ page errorPage="f3.jsp" %>
```

```
<%
int n1 = Integer.parseInt(request.getParameter("t1"));
int n2 = Integer.parseInt(request.getParameter("t2"));
int c = n1/n2;
out.print("c is:"+c);
%>
</body>
</html>
```

f3.jsp:-

```
<html>
<body>
<%@ page isErrorPage="true" %>
Exception Information:
<%= exception %>
</body>
</html>
```

5. extends:-

It is used for Inheritance by using extends attribute a current Jsp page is going to inherit the properties of other class.

Syntax:-

```
<%@ page extends="class_name" %>
```

6. language:-

It is used to specify the type of language are using. the default language is java.

Syntax:- `<%@ page language="java" %>`

7. buffer:- It is used to specify the output buffer size.

By default the output buffer size is 8kB.

Syntax:- `<%@ page buffer="16kb" %>`

8. contentType:-

It is used to specify which type of response has to generate by a web container. the default contentType is text/html

Some of the possible contentTypes are text/css, text/csv, image/gif, application/ms.word.

syntax:- `<%@ page contentType="text/html" %>`

3. taglib directive:-

It is a tag library it contains some pre-defined tags.

syntax:- `<%@ taglib uri="taglibraryname" prefix="name" %>`

Implicit Objects:-

During Jsp translation phase a web container creates some built-in objects those are called implicit objects

1. out
2. request
3. response
4. session
5. exception
6. config
7. application
8. page
9. pageContext

1. out:-

it is a output object created by jspWriter class. it is used to print some message

eg:-

```
<%
out.print("welcome");
%>
```

2. request :-
It is created by HttpServletRequest class. it contains only one method getParameter().

Syntax:-
```
<%
String x = request.getParameter("name");
out.print(x);
%>
```

3. response :-
It is used to maintain response information. It is created by HttpServletRequest class. it contains various methods like sendRedirect(), addCookie(), setContentType().

eg:-
```
<% out.print("welcome");
response.sendRedirect("hello.html");
%>
```

4. exception :-
It is used to display some information about exception. it is created by using Exception class.

eg:- `<%= exception %>`
`<%= exception.printStackTrace() %>`

5. page :-
It is used to refer the current jsp page. it is similar to this object.

6. session :-
It is created with the help of HttpSession class.

→ To place some information on the session use setAttribute()

Syntax:- setAttribute(name, value)

To retrieve the value from the session use getAttribute().

Syntax:- getAttribute("name");

Write a jsp program to manage session information.

Sess.html
```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br>
</form>
</body>
</html>
```

welcome.jsp
```
<html>
<body>
<%
String name = request.getParameter("uname");
out.print("welcome"+name);
session.setAttribute("user",name);
%>
<a href="second.jsp">second jsp page</a>
</body>
</html>
```

Second.jsp
```
<html>
<body>
<%
String name = (String)session.getAttribute("user");
out.print("hello"+name);
%>
```

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

7. **config:-** It is used to maintain config information about particular servlet.

It is created using servletconfig class

→ To pass any parameters we require <init-param> in web.xml.

→ It contains a method getInitParameter()

web.xml

```
<web-app>
<servlet>
<servlet-name> rushitha </servlet-name>
<jsp-file> /welcome.jsp </jsp-file>
<init-param>
<param-name> rushil </param-name>
<param-value> sun.jdbc.odbc.JdbcOdbcDriver </param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name> rushitha </servlet-name>
<url-pattern> /welcome </url-pattern>
</servlet-mapping>
</web-app>
```

welcome.jsp

```
<html>
<body>
<%
string name= request.getParameter("uname");
out.println("welcome" + name);
string v = (string) config.getInitParameter("rushil");
out.print(v);
%>
<br>
<%=v%>
</body>
</html>
```

8. **application:-**

It is created by using servletcontext class. It is used to pass some parameters to all jsp pages available in web application.

→ To pass information to all jsp pages use <context-param> tag in <web-app> tag.

→ To retrieve the values use getInitParameter()

eg:- web.xml

```
<context-param>
<param-name> rushil </param-name>
<param-value> sun.jdbc.odbc.JdbcOdbcDriver </param-value>
</context-param>
```

welcome.jsp

```
<html>
<body>
<%
string v = application.getInitParameter("rushil");
out.print(v);
%>
<br>
<%=v%>
</body>
</html>
```

9. **pageContext:-**

It is used to share data between various jsp pages based on its scope

→ To store the information use a function setAttribute()

syntax: pageContext.setAttribute("name","value","scope");

scope ⇒ pageContext.SESSION_SCOPE
pageContext.PAGE_SCOPE
pageContext.APPLICATION_SCOPE
pageContext.REQUEST_SCOPE

→ To retrieve information use getAttribute()

Syntax:

pageContext.getAttribute("name","scope")

→ To remove the page attribute from the page use following Syntax

Syntax:- pageContext.removeAttribute("name","scope");

* Write a jsp program to demonstrate pageContext

.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go">
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome"+name);
pageContext.setAttribute("user",name,pageContext.
                         SESSION_SCOPE);
%>
<a href="second.jsp">second jsp page</a>
</body>
</html>
```

Second.jsp

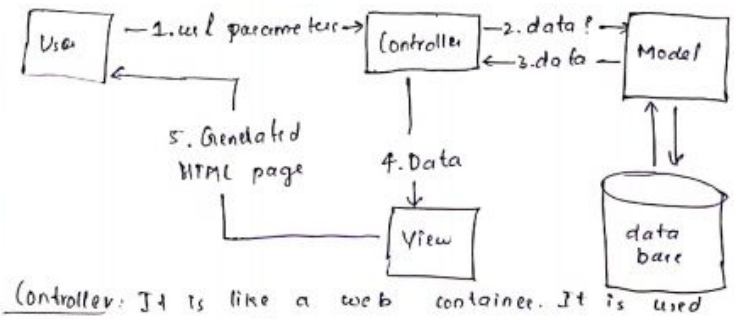
```
<html>
<body>
<%
String name=(String)pageContext.getAttribute("user",
                pageContext.SESSION_SCOPE);
out.print("hello"+name);
%>
</body>
</html>
```

MVC Architecture:

→ MVC architecture is useful for design applications using any server side scripting langs. like. JSP (or) ASP.

MVC - model view Controller

contain 3 components




Controller: It is like a web container. It is used for handling all requests.

Initially whenever client sends a request a web container recieves those parameters.

→ A web container will seperate both presentation logic and business logic before translations.

→ A controller will pass or send all required info. to model component.

Model: Model contains data and code required for business logic.

* All java bean components are also managed by model component.

It recieves the data from controller and process them sends the response to controller.

View: It contains the code required for presentation logic. It will recieves the responce from the controller and embed the responce in presentation logic. Then generate HTML page and send back to client.

UNIT-III Part-I

Bean Introspection: It is a mechanism used for introspect or analyse the functionalities of a java bean class.

A java bean contains 3 components

1) properties
2) methods
3) events

→ To retrieve all bean information we need to import "java.beans" package.

→ To get bean information use a built-in method getBeanInfo() available in the Introspector class.

Syntax: Introspector.getBeanInfo("class_name");

→ The resultant information will be stored in BeanInfo object BeanInfo b = Introspect.getBeanInfo("class_name");