S. No.	Date	Title	Page No.	Teacher's Sign/Remarks

— Web Technologies —

Unit-I :-

HTML common tags :-

List
Tables
images
forms
frames
Basics of css and types of css.

Client-Side programming :- (Java script) :-

Introduction to Java Script
Prompt dialog box
operators
Control structures
functions
Event handlers (onclick, onsubmit, etc.,)

Introduction to XML :-

XML basics
structuring data
Document type definition
XML name spaces
Document object model (DOM)
(TextBook-1)

Unit-II :-

1) Server-side Programming (PHP) :-

2) Declaring variables,
Data types

3) operators,
control structures,

4) Strings &
arrays

4) Functions

5) reading data from web form controls (like text buttons, radio buttons, list, etc).

6) Working with the ~~java.sql package~~ databases.

7) ~~poor~~ Handling sessions & cookies. (Text Book-2)

Unit-III :-

Introducing JDBC :-

JDBC drivers
features of JDBC
JDBC APIs
major classes and Interfaces
JDBC processes with the java.sql package
→ processes with the javax.sql package
working with transactions.

Web applications :-

exploring the HTTP protocol
web architecture models
the MVC architecture

Working with servlets:-

the features of servlets
exploring servlet API
the servlet life cycle
creating a servlet
the Http servlet Request and Http Servle
Response Interfaces
request delegation and request scope.

Unit IV:-

Handling Sessions:-

Introducing session tracking mechanism
the java servlet API for session tracking

Introducing JSP:-

advantages
the architecture
life cycle of JSP
JSP basic tags & implicit objects
action tags.

Implementing Filters:-

Working with filters
filter API configuring filters
Initiallzing parameter in filter

* HTML TAGS *

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	contains information for the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Define HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment
<label>	Defines a label for an input> element
<button>	Defines a clickable button
<textarea>	Defines a multiline input textarea
<form>	Defines an HTML form for user input
<input>	Defines an input control
	Defines important text
<b>	Defines bold text
	Defines emphasized text
<a>	Defines a hyperlink
<img>	Defines an image.

HTML:- Hyper Text Markup Language.

* Hyper ⇒ Link between one page to another page.

* Markup ⇒ It means tags.

< → opening tag (<html>)

> → closing tag (</html>)

⇒ Html is using to create a web page.

⇒ Webpage means document. It contains the information.

⇒ Website means collection of web pages.

Structure of HTML:-

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

(or)

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

⇒ HTML :-

It is a language, it developed by Tim Berners Lee in 1991. It is used to create a web page

⇒ Web page :-

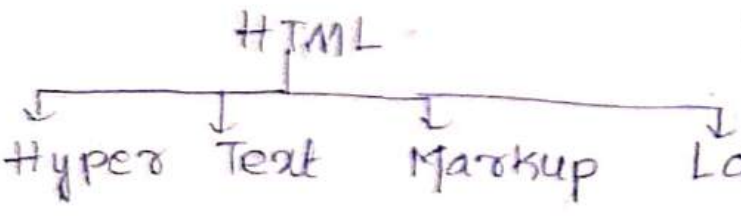
A web page is a page which contains information about a particular topic

⇒ Website :-

It is a collection of web pages, it containing information related to a particular topic.

→ Html document contains html tags and plaintext

HTML → It is not a programming language. It is a markup language.



⇒ Hyper :- Link between one page to another page.

⇒ Text :- collection of information.

⇒ Markup :- It is used to describe web pages that are tags.

⇒ HTML describe the structure of webpage,

By using Html, we can create our own website

Tag:-

⇒ Tag are derivated by angles brackets
<"(opening tag)" >"(closing tag)"

⇒ Tags are not case sensitive

⇒ HTML extension is
q. html
filename

Advantages:-

⇒ It is a very easy and simple language
⇒ It can be easily understand and modify
⇒ It is very easy to make a effective web page.
⇒ Using HTML we can add graphics, videos etc.

Basic tags:-

1) Html:- It defines a root element of a document
html tag contains starting tag and ending tag.
Ex:- <html> ---- </html>

2) head:- It is containing meta data. Data about data is placed between html tag & body tag.

Meta data :- Data about the html document

⇒ Meta data is not displayed.

⇒ Meta data is typically defined the document title, link, style, script and other meta information.

i) Title :- Title defines the title of the document. The title must be text only & it is shown in the browser title bar.

ii) Link :- It defines relation between a document & external source.

Ex :- <link ref "stylesheet" type = "text/css" href = "robo css">

iii) style :-

It defines style information of a html document.

Ex :- <style type "text/css">

iv) Script :-

It is used to define which type of scripting used in html document.

Ex :- <script language "Javascript">

v) Body :-

It define the body of the html document.

Ex :- <body> - - - - - - </body>

27/10/2022

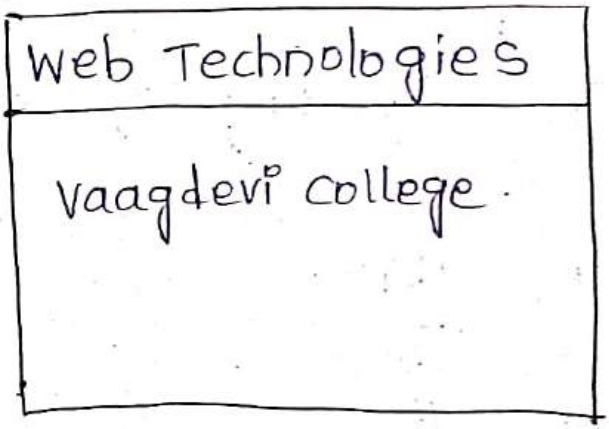
① Basic html program.

<!DOCTYPE HTML5>
document type version of Html.

```
<html>
<head><title>Web Technologies</title>
</head>

<body>
vaagdevi college
</body>
</html>.
```

output :-



① *Heading Tags :-

→ It is used for heading of a webpage

→ Heading are used to highlighting important topics.

There are 6 levels of headings.

<h1> ⇒ Largest heading

<h6> ⇒ smallest heading

Syntax:-

```
<h1> any content </h1>
<h2> any content </h2>
<h3> any content </h3>
<h4> any content </h4>
<h5> any content </h5>
<h6> any content </h6>
```

(2) Write a html program with all basic tags <h1>, <h2>, <h3>, <h4>, <h5>, <h6> and display it.

```
<html>
<head><title> Heading Tags </title></head>
<body>
<h1> Hello! </h1>
<h2> Hello! </h2>
<h3> Hello! </h3>
<h4> Hello! </h4>
<h5> Hello! </h5>
<h6> Hello! </h6>
</body>
</html>
```

Output :-

2) Formatting Tags:-

Formatting tags are used to present a text in different format.

i) <B> ⇒ Used to display bold type of text.

Syntax:- <B> Hello! </B>.

ii) <I> ⇒ Used to display italic type of text

Syntax:- <I> Hello! </I>.

iii) <U> ⇒ Used to underline the text.

Syntax:- <U> Hello! </U>.

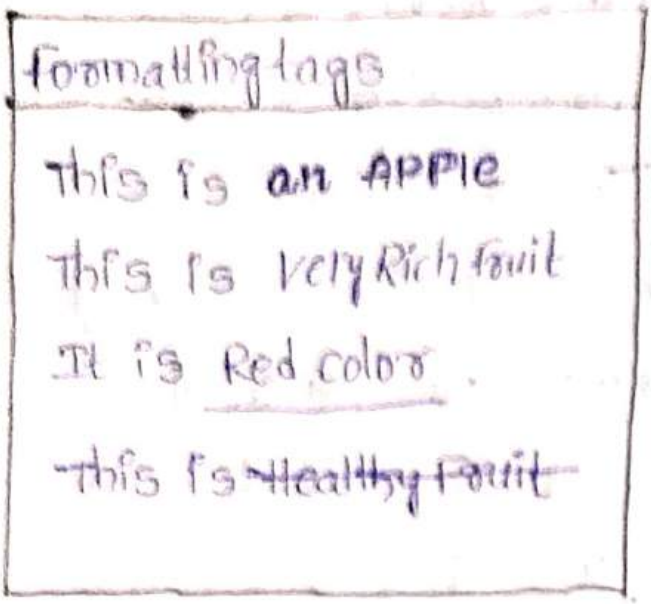
iv) <S> ⇒ Used to striking a text i.e it gives strike line from the middle of text.

Syntax:- <S> Hello! </S>.

(3) Write a html program using Bold text Italic text, underlined text, strike text

```
<html>
<head><title>formattingtags</title></head>
<body bgcolor="blue", text="white">
<p>This is <b> an Apple</b></p>
<p> this is <i> very Rich fruit</i></p>
<p> It is <u> Red color</u></p>
<p>This is <s>Healthy fruit</s></p>
</body>
</html>
```

Output :-



v) <sub> ⇒ Used for defining subscript text this tag is used to chemical formulas

Ex:- H_2O Syntax:- _{anycontent}

vi) <sup> ⇒ Used for defining superscript text. Defines mathematical expressions.

Syntax:- ^{any content}

Ex:- a^2+b^2

vii) Inserted text :-

<ins> ⇒ Used to define a text that has been inserted into a document.

Syntax:- <ins> any content </ins>.

④ Write a html program to format the text to superscript, subscript and inserted text.

```
<html>
<head> <title> formatting text </title> </head>
<body>
<p> the formula ps = (a+b)<sup> 2 </sup> .
</p> <br> line break
<p> the chemical ps = H<sub> 2 </sub> O<sub>
2 </sub> </p> <br>
<p> the formula ps = (a+b <ins> +c </ins>
<sup> 2 </sup> </p>
</body>
</html>
```

Output :-

Formatting text
The formula ps = $(a+b)^2$
The chemical is = H_2O_2
The formula is = $(a+b+c)^2$

28/10/20

* <BIG> ⇒ Used to display text in bigger font/text, they are current default size

Syntax :- <BIG> any content </BIG>

* <SMALL> ⇒ Used to display text in smaller font, they are in current default size.

Syntax :- <SMALL> content </SMALL>

* ⇒ It is same as bold tag.
Syntax:- content .
* ⇒ It is same as italic tag, used to display emphasized text.
Syntax:- content .
* <hr> ⇒ Used for creating a horizontal line.
Syntax:- <hr> content </hr>
*
 ⇒ Used to give a line break.
Syntax:-
 content </br>
* <blink> ⇒ Used to give a blinking effect to a particular text.
Syntax:- <blink> content </blink>
* ⇒ Used to display the text in middle of the page.
Syntax:- content
* ⇒ Defines a paragraph.
Syntax:- content
* <pre> ⇒ Used to display text, which is to be presented exactly as written in the pretag (<pre>)

Font tag :-

 :- Used to define font size, color and face of a text.

3 attributes :

1) size:- It specifies the size of the text

2) color:- used to give the color to a text.

3) face:- Used to display the face of the text.

(or)

Used to specify the font family of the text inside element.

Syntax:- <font color=name/value face="name" content.

(5) Write a html program on font tag?

```
<html>
<head><title>font tag</title></head>
<body>
<font size="6" face="arial black" color=
"blue">Hello!</font><br>.
<font size="8" face="calibri" color="Red
">Hello!</font>
</body>
</html>.
```

Output:-

* Comments :- Used to insert comments in the source code. these are not displayed in the browser.

Types :- 1) single line 2) Multi line.

(6) 1) Single line comments :-

Syntax :- <!-- content -->

Ex :-
```
<html>
<head>
<title> web Technologies </title></head>
<body>
<!-- single line comment --> (or ---)
Vaagdevi college </body>
</html>.
```

output :-

web Technologies
Vaagdevi college.

2) Multi line comments :-

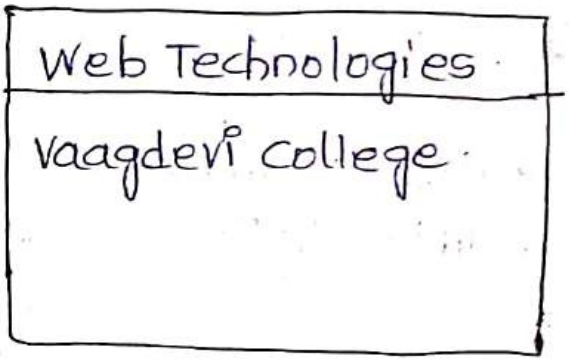
Syntax :- <!-- content ----

--->

Ex :-

```
<html>
<head>
<title> web Technologies </title>
</head>
<body>
  <!--comments are two types. Those are:
      single & multi line comments
      this example is multilined -->
  Vaagdevi college
</body>
</html>
```

Output:-



* List:-

→ List is used to display the data (or) information in a webpage.

Types:- 1) Ordered List
2) Unordered List
3) Description (or) Definition List

1. Ordered List:- It is used to create a list of related items in a specific order (or). which have an inherent order & each item is numbered.

⇒ Defined by <ol> tag.

⇒ Starts with <ol> tag.

⇒ List items starts with <li> tag.

⇒ the list items will be marked with nos by default.

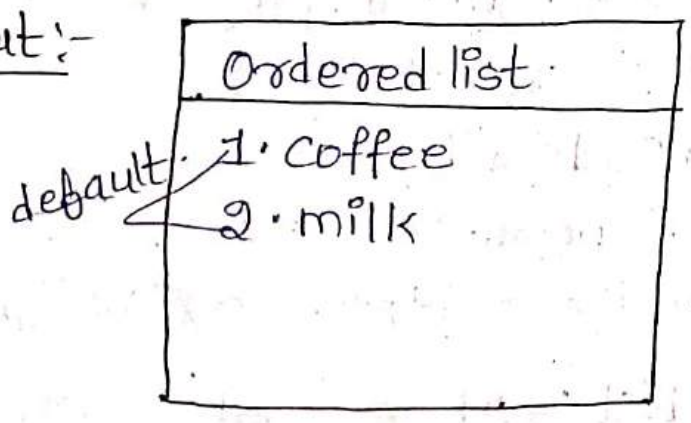
Syntax:-

```
<ol>
  <li> content </li>
</ol>
```

⑦ Eg:-

```
<html>
<head>
<body><title> Ordered list </title>
</head>
<body>
<ol>
<li> coffee </li>
<li> milk </li>
</ol>
</body>
</html>
```

Output:-



* Type attribute :-

The type attribute of the <ol> tag defines the type of the list marker.

1) Type "A" :-

The list items will be nor with

upper case letters.

2) type="a" :- the list items will be nors with lower case letters.

3) type="I" :- the list items will be nors with Bigger Roman nor's.

4) type="i" :- the list items will be nors with smaller roman nor's.

examples:-

```
<html>
<head>
<title>Order</title>
</head>
<body>
<ol type="A"> → type="a" or type="I" or type="i".
<li>coffee</li>
<li>milk</li>
</ol>
</body>
<html>
```

outputs:-

1)

Order

A. coffee
B. Milk

2)

order

a. coffee
b. milk

3)

order

I. coffee
II. milk

4)

order

i. coffee
ii. milk.

2. Unordered list :- which have no inherent order and each item is bulleted (default)

(or)

Used to create a list of related item in Particular unorder.

⇒ starts with <ul> tag.

⇒ List items starts with <li> tag.

⇒ The list items will be marked with bullets (Default or disk).

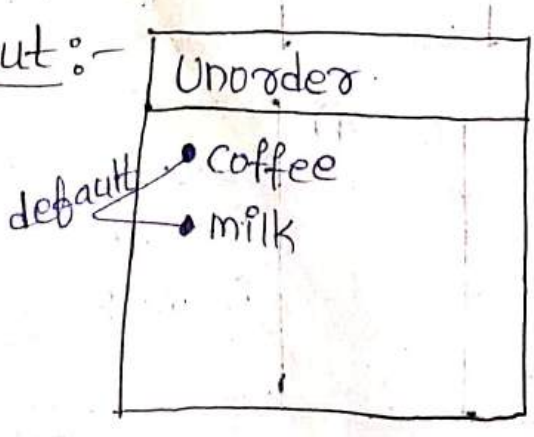
Syntax:-

```
<ul>
<li> content </li>
</ul>
```

⑧ Ex:-

```
<html>
<head> <title> Unorder </title> </head>
<body>
<ul>
<li> coffee </li>
<li> milk </li>
</ul>
</body>
</html>
```

Output :-



Type attributes :-

1. Type="circle" => sets the list item marker to a circle.
2. Type="square" => sets the list item marker to a square.
3. Type="disc" => this is the default style In this style, the list items are marked with bullets.
4. Type="none" => In this style, the list items are not marked.

Ex:-

```
<html>
<head><title> unorder </title></head>
<body>
<ul type="circle"> or none or square
 <li> coffee </li>
 <li> Tea </li>
</ul>
```

Output :- 1)

unorder
- coffee
- Tea

2)

unorder
- coffee
- Tea

3. Description (or) Definition List :-

A description list is a list of terms, with a description of each item.

⇒ <dl> tag defines the description list.

⇒ <dt> tag defines the term (name)

⇒ <dd> tag describes each term.

Syntax:-

```
<dl>
<dt> content </dt>
<dd> content description </dd>
</dl>
```

(a) Example:-

```
<!DOCTYPE html>
<html>
<head> <title> Description List </title>
</head>
<body>
<dl>
<dt> coffee </dt>
<dd> - black hot drink </dd>
<dt> Milk </dt>
<dd> - white cold drink </dd>
</dl>
</body>
</html>
```

Output:-

Description List

coffee
- black hot drink

Milk
- white cold drink

Tag	Description.
	Defines an unordered list.
	Defines an ordered list.
	Defines a list item.
<dl>	Defines a description list.
<dt>	Defines a term in a description list.
<dd>	Describes the term in a description list.

Marquee :-

Defn :- Used for scroll the text/images displayed either horizontally or vertically down your website.

Attributes :-

1) Behaviour :- Define the type of scrolling (scroll, slide & alternative).
2) Bgcolor :- specify the background color
3) Direction :- Define the direction scrolling the content (left, right, up, down).
4) loop :- Specify the how many times to loop.
5) Height :- Define the height of marquee
6) width :- Define the width of marquee

(10)

Ex:-

```
<html>
<head>
<title> marquee </title>
</head>
<body>
<marquee loop=6 direction="right"
    height = 500 width=800
    bgcolor="pink"> <font size=30
color="blue">welcome</font>
</marquee>
</body>
</html>
```

O/P:-

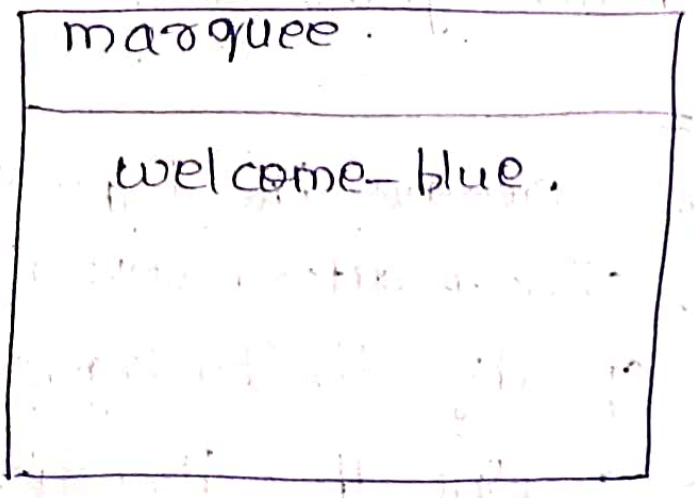


Image:- (<img>)

<img> used to put on image on html document

Attribute:-

1) src:- (source code)

It specifies the location of the image which is to be placed in a webpage.

2) Alt:- (Alternative text).

It specifies the alternative text for the image.

3) Border:- It specifies the border of image

4) Height:- Height of an image

5) width:- width of an image.

Syntax:-

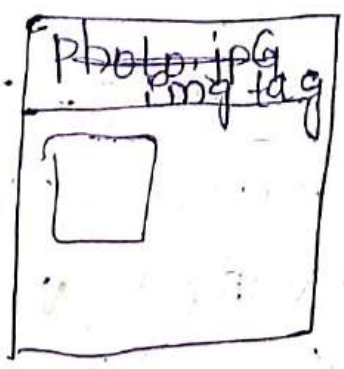
location of image.

```
<img src = "photo.jpg", width="100"
height="100" alt = "alternative text
</img>
```

Ex:-

```
<html>
<head><title> Img tag</title></head
<body>
<img src = "abc.jpg" alt="no
image" width= 200 height=200
</img>
</body>
</html>
```

o/p:-



* Frames:- (Dividing a website into rows/columns)

These are used to divide a webpage into different sections/frames.

⇒ Each frame is independent of others

Attributes:-

1) Frameset:-

It is used to divide browser window into multiple frames (or) It is the collection of frames in the browser window.

2) Rows:- It is used to divide a single webpage into multiple rows horizontally

3) ~~Columns~~ Cols:- It is used to divide a single webpage into multiple rows vertically.

4) Border:- used to specify a border.

5) Border color:- used to give a color to border.

6) Frame:- This tag is used to external webpages.

7) Name:- Name to a webpage.

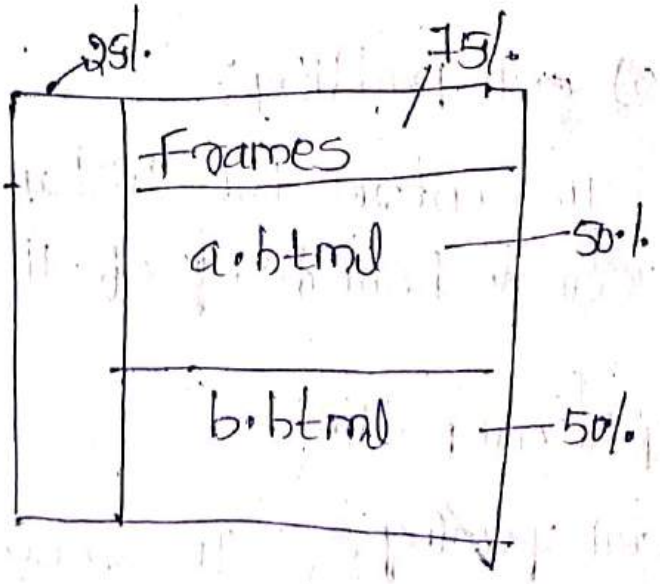
8) src:- location of webpage

ii)

Ex:-

```
<html>
<head><title> frames </title>
<frameset col = "25% 75% ">
<frameset rows = "50% 50%">
<frame src = "a.html"> (program name.)
<frame src = "b.html">
</frameset>
</head>
</html>
```

output :-



Tables:-

Html table allows you to arrange the data into rows & columns.

⇒ A table is divided into rows & columns and each row is divided into data cell.

→ A data cell contain text, images, list, paragraphs, forms, table etc.

Attributes:-

1) Table :-

It defines a table

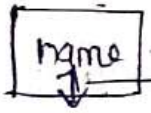
2) border :- It defines a border of the table

3) bgcolor :- To set the table background color.

4) align :- set the alignment (left, Right, centre).

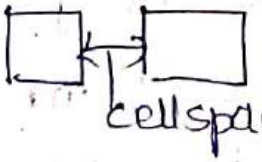
Imp: 5) cell padding :-

To control the distance b/w the datacell & boundaries of the cell.



cell padding.

6) cellspacing :- To control the distance blw the two cells.



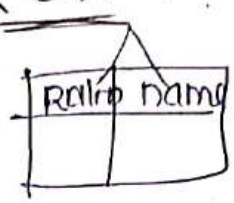
7) Height :- To set the height of the table

8) width :- To set the width of the table

9) Border: set the border of the table.

* <th> (table heading).
* <tr> (table row)
* <td> (table data).

→ <th> :- To define a header cell in a tabl



→ <tr> :- To define a row of the table
→ <td> :- To define a data of the table.
→ <caption> :- To define a caption of the table.

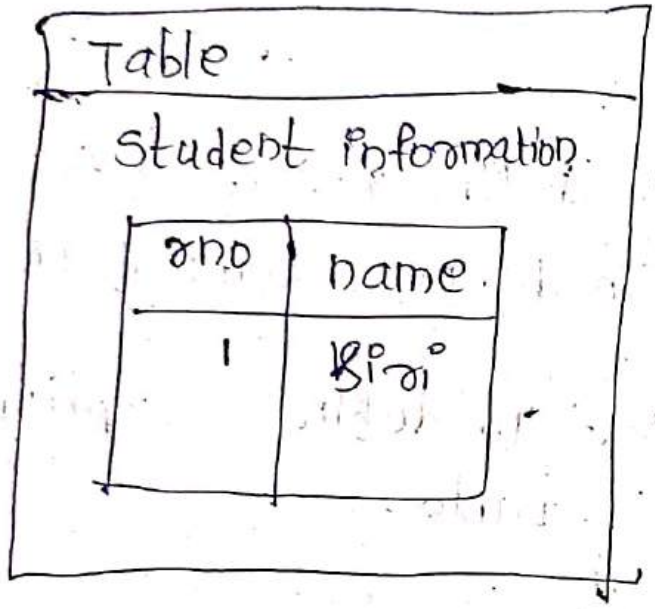
12
Ex:-

```
<html>
<head><title> table </title></head>
<body bgcolor="pink">
<Table border=5>
<caption> Student information </caption>
<tr>
<th> rno </th>
<th> name </th>
</tr>
<tr>
<td> 1 </td>
<td> vgg </td>
```

of siri

</tr>
</table>
</body>
</html>

o/p:-



D:- 5/11/22

* rowspan:- (merge)

It specifies the no. of rows / a cells should be span vertically.

Syntax:-

<td rowspan=2> (merge 2 cells)

* Colspan:-

It specifies the no. of columns / a cells should be span horizontally

Syntax:- <td colspan=2>

(or <th)

13/Ed:-

```
<html>
<head><title> rowspan & colspan</title>
                                </head>
<body>
<table border = 5>
<tr>
<th colspan= 2> Degree clg</th>
</tr>
<tr>
<th rowspan= 2> course</th>
</tr>
<tr>
<td> BSc</td>
</tr>
<tr>
<td> Bcom</td>
</tr>
</table>
</body>
</html>
```

o/p:-

rowspan & colspan

Degree clg	
course	BSC
	B'com

*CSS (Cascading Style Sheet):-

CSS:- CSS is a presentation language used to design a webpage attractively. CSS is a stylesheet language used to discribe the presentation semantics.

→ CSS is not a programming language. It is a styling language.

→ It is a creativity focused.

CSS Comments:-

1) single line comment :- /* ---- */
2) multi line comment :- /* ---

--- */

Advantages:-

→ CSS save time
→ pages download faster
→ Easy maintanence.
→ styles to html
→ multiple devices compatability.
→ Global web standards.
→ offline browsing.

Disadvantages:-

→ security ~~less~~ threat
→ Lack of variables
→ Inconsistent browser support.

CSS syntax:-

Selector - html tags h1/p/ br.
{ → Declaration. block.
property1 : value; → Declaration
property2 : value; → Declaration
}

Eg:- h1 {
color : blue;
font-size : 30px;
}.

CSS ~~&~~ consists of selector & declara-tion block.

→ Selector:- A selector is a html tags at which ~~the~~ style will be applied (h1, p)

→ Declaration:- Each declr consist of property name & value.

⇒ Declaration block:- It contains 1 or more declarations seperated by;

→ property seperated by ':'

* CSS Types:-

3 Types:

1) Inline style sheet - single text to webpage
2 Internal (or) embedded stylesheet
3. External style sheet - coll of webpage.

1. Inline style sheet:-

It is used to apply a unique style to single html document.

Syntax:-

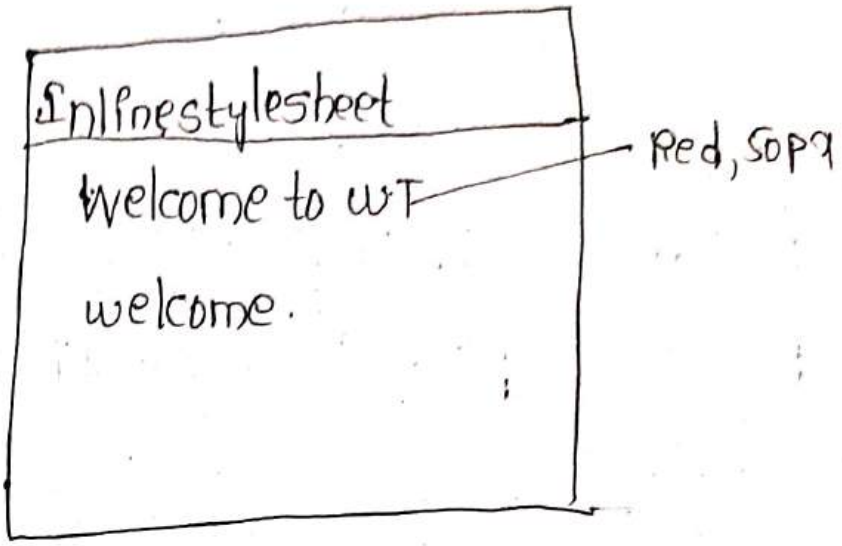
```
<tagname style="attribute1 : value;
        attribute2 : value ---->text
</tagname>.
```

Ex:- Inline.html (file save)

```
<html>
<head>
<title> Inline stylesheet </title>
</head>
<body>
<h1 style="color : red;
        font-size : 50px">
    welcome to WT </h1>
```

```
<h1> welcome </h1>
</body>
</html>
```

o/p:-



2) Internal (or) embedded stylesheet:-

It is used to define a styles for a single html page.

→ An internal css defined in the head section of html page within a style element.

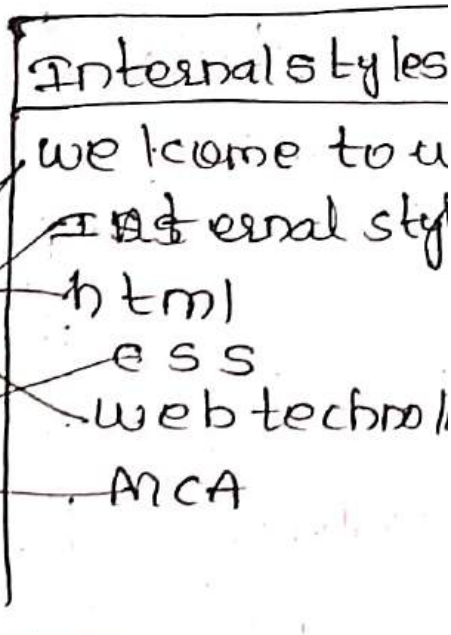
Syntax:-

```
<style type="text/css">
    selector1{
        attribute1: value;
        attribute2: value;
    }---
selector2{attribute$: value;
</style>}  attribute 2: value;
```

Ex:- internal.html

```
<html>
<head><title> Internal stylesh
                </title>
<style type = "text/css">
h1 { color: red; text-align: cente
b { color: blue; font-size = 50px;
</style>
</head>
<body>
<h1> welcome to WT </h1>
<b> Internal style sheet </b>
<h1> html </h1>
<b> CSS </b>
<h1> web Technology Lab </h1>
<b> MCA </b>
</body>
</html>
```

O/P:-



3) External style sheet :- (.html & .css)

It is used to define the styles for many html pages (more than 1)

→ To use an external style sheet add a link to within the head section of each html page.

;; Syntax for CSS :-

```
selector
    {
    attribute1 : value;
    attribute2 : value;
    }
```

;; syntax for html :-

```
<link ref = "stylesheet" type = "text
      ↓ reference          class" CSS"
href = "filename.CSS">
↓ hiperreference.
```

Eg:-

CSS

wt.css

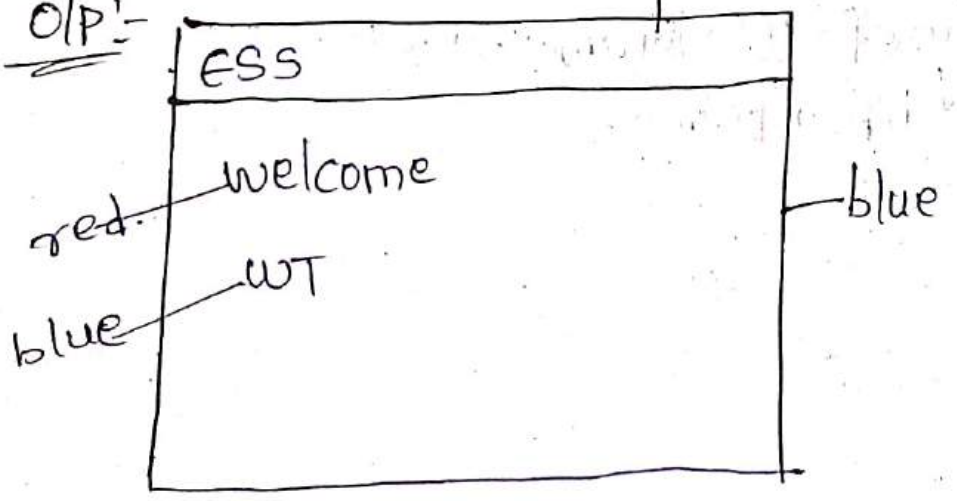
```
body {bg color : blue; }
h1 {color : red; }
b {color : blue; }
```

html

wt.html

```
<html>
<head><title>
 css</title>
<link ref="style
   sheet" type
     text/less"
 href = "wt.css
</head>
<body>
<h1>welcome</h
 <b>wt</b>
</body>
</html>
```

O/P:-



behaviour of webpage.

Java Script :- (pop-up msg)

OOPs (high-level tags) <a>, <h1>, <b><lb> ...

Java script is used mainly for enhansing the interaction of a user with the web page.

→ In other words, we can make your web page more lively & interactive with the help of Javascript.

→ Java script code is inserted blw <script> & </script>

HTML
1. It define the content of webpage.
CSS
1. To specify the layout (styles) of web page.
JavaScript
1. To specify the behaviour (pop-up) of webpage.

Applications (uses) :-

1. client side validations (only user can perform actions)
2. Dynamic drop-down menus
3. Displaying the date & time.

4. Displaying popup windows & dialog box. (like alert, prompt, confirm).

Features :-

1. It is open source scripting language. (edit, no password)
2. It is plat-form independent (run any where).
3. client side validation
4. functional styles (c - language structure if, ifelse, operators)
5. It is a case sensitive language.
6. Dynamic typing (editing)

Advantages :-

1. It is supported by most of the webbrowsers (firefox, Google)
2. It is simple to learn & implement.
3. It is executed client-side.
4. It performs an operations heavy fast.

Disadvantages :-

1. It can't be used for networking applications.
2. It doesnot have any multithreading & multiprocessing capabilities.

3. It has security issues. being a client side scripting Language.

- It divided into 3 types.

1. prompt() → dialog box
2. Alert() → warning msg
3. confirm() → ok & cancel

1. Prompt():- used to display a dialog box that prompts the user for input.

- This method returns the input value. if users ~~gets~~ clicks [OK] otherwise it returns null.

Syntax:-

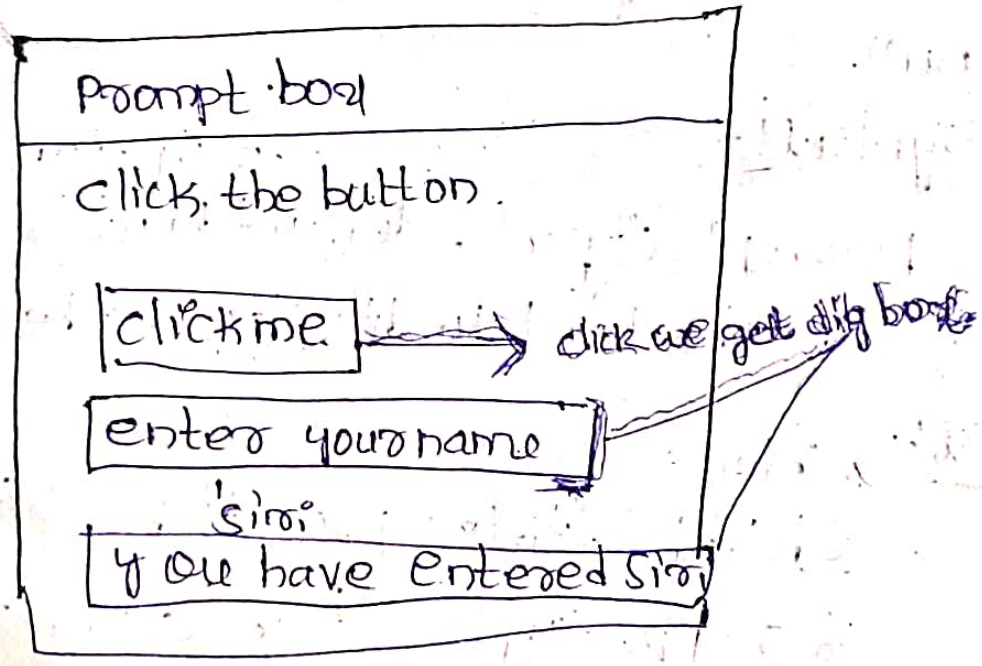
prompt(text, defaulttext)
↑ optional.
↓
the text to display in the dialog box

Ex:-

```
<html>
<head><title> prompt box </title>
<script type="text/Javascript">
function getvalue() {
var retval = prompt("enter your
name");
document.write("you have entered"
+retval);
}
```

```
</script>
</head>
 <body>
 <p> click the button </p>
  <form>
   <input type="button" value="clickme"
          onclick="getvalue()">
  </form>
  </body>
  </html>
```

o/p:-



2. Alert() :-

Alert bod is used to give a warning message to the user..

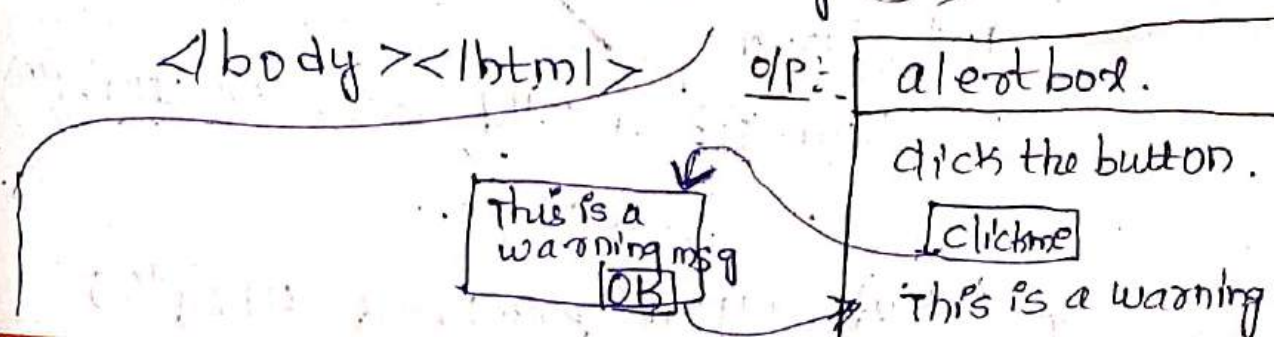
Syntax:-

```
alert("hello message")
```

Ex:-

```
<html>
<head><title> alert box </title>
<script type="text/javascript">
function warning(){
alert("this is a warning message");
document.write("this is a warning
                                ;
}
</script></head>
<body>
<p>click the button</p>
<form>
<input type="button" value=
                "clickme"
        onclick="warning()">
</body></html>
```

o/p:-



3. confirm() :- ok, cancel.

It is used to display a dialog box with a message on 'ok button & cancel button.

Syntax:-

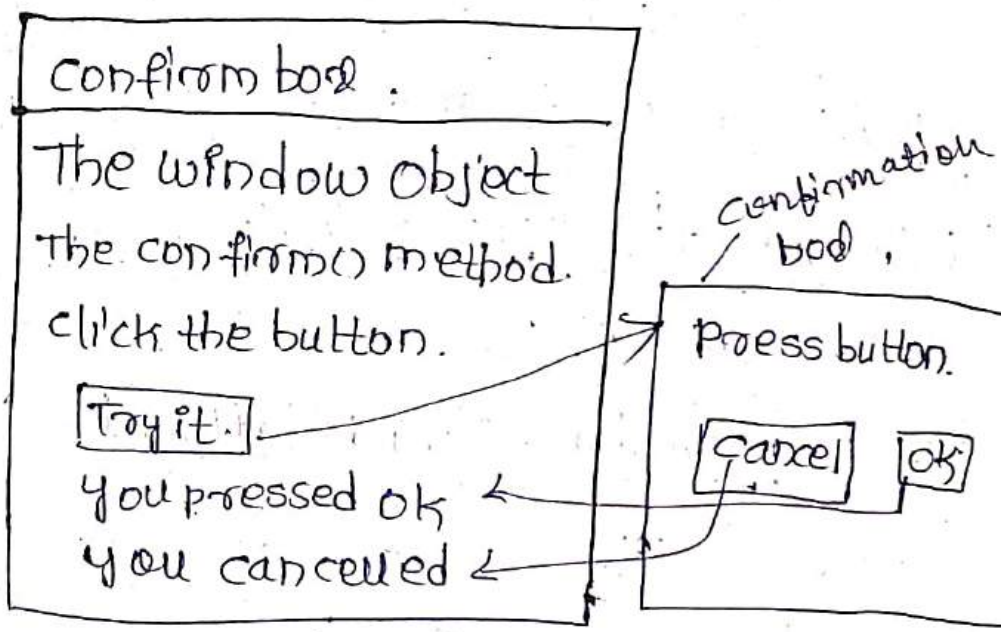
```
confirm("msg")
```

Ex:-

```
<html>
<body>
<head><title> confirm box </title>
<body>
<h1> The window object</h1>
<h2> the confirm() method </h2>
<p> click the button </p>
<button onclick = "myfunction()">
        Tryit</button>
<p id="demo"></p>
<script>
  function myfunction()
{
  let text;
  if (confirm("press button!")
      == true) {
  text = "you pressed OK!";
```

```
else {
  text = "you cancelled!";
  }
document.getElement By Id ("demo").inno
                                  HTML=text;
}
</script>
</html>.
```

o/p:-



Imp :-

7/11/2022

Operators :-

Operator is a special symbol used to perform operations on operands.

operations → +, −, *, /

operands → (variables, values)
↓ a, b or x, y — ↓ 1, 2, 3, 4, — —

Ex :- 1. x+4

2. x+5

3. 5+5

Types of operators :-

1. Arithmetic operators :-

+, −, *, /, %.

2. Comparision (or) Relational operators :-

<, >, <=, >=, !=

3. String operators :-

characters.

4. Logical operators :-

AND, OR, NOT.

5. Assignment operators

6. Bitwise operators

7. Special operators

1. Arithmetic operators:-

Used to perform arithmetic (x=10, y=20)

+	addition	x+y=30
-	subtraction	x-y=-10
*	multiplication	x*y=200
/	division	x/y = 0.5
%	modulus	x%y = 0
++	Inc	x++ or ++x = x+1 = 10+1=11
--	dec	x-- or --x = 9

2. Comparision operators / Relational operators:-

Used to compare 2 operands(value) & return a boolean value (TRUE, FALSE)

x=10, y=20.

<	lessthan	x<y → TRUE
>	greaterthan	x>y → FALSE
<=	lessthan or equalto	x<=y → TRUE
>=	Greaterthan or equal to	x>=y → FALSE
==	equal to	x==y → FALSE
!= or =!	notequal to	x!=y → TRUE

3) String operator:-

Used to concatenate (join/add) 2 or more strings.

Ex:- "web" + "Technology" → webTechnology

4. Logical operator:-

Used to compare two relational expressions and return a boolean values i.e TRUE or FALSE

x=10, y=20, z=30.

1. Logical AND (&&)

Ex:- (x>y) && (x>z)

↓

f && f → F.

Truth table.

x	y	&&
F	f	f
T	f	F
f	T	F
T	T	T

2. Logical OR :- (||)

Ex:- (x>y) || (x>z) → F || F → F

(or)

(x>y) || (x<z)

F || T → T.

Truth table

x	y	||
F	f	f
T	f	T
f	T	T
T	T	T

3. Logical NOT :(!)

Truth table

$(x > y) ! (x > z)$

F ! F → F → opposite

o/p :- True

x	!
T	F
F	T

4) Assignment operator :-

These operators are used to assign a values to a variable.

$x = 10$

=	Assignment operator	$x = 10$.
+=	addition assignment	$x += 10$ → $x = x + 10$ = 20.
-=	subtraction assignment	$x -= 10$ (or) $x = x - 10$ = 0.
*=	multiplication assignment	$x *= 10$ (or) $x = x * 10$.
/=	Division assignment	$x /= 10$ (or) $x = x / 10$.
%=	Remainder assignment	$x \%= 10$ (or)

6. Bitwise operators:-

These operators are used to perform operations on binary no.'s (0,1).

Eg:- x=10, y=11

1. Bitwise AND (&) → x & y → 10

Truth table:-

x	y	&
1	1	1
0	1	0
1	0	0
0	0	0

x = 10
y = 11
10

2. Bitwise OR (!):-

Truth table:-

x	y	!
0	0	0
1	0	1
0	1	1
1	1	1

x ! y → 11

x = 10
y = 11
11

3. Bitwise XOR (^) → x ^ y → 01 (or ⊕)

^ cap operator

Truth table:-

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

x = 10
11
01

1. Bitwise NOT (~):- unary negation operator

Truth table:-

x	~
1	0
0	1

~1 → 0

x = 1 0
~ 0 1

3. special operators:-

1. type of
2. delete
3. in
4. instance of

* Conditional (or) Control structures:-

Conditional statements are used to decide the flow of execution based on the different conditions.

→ If a condn is true then perform one action. & if the condn is false then perform another action.

Types:-

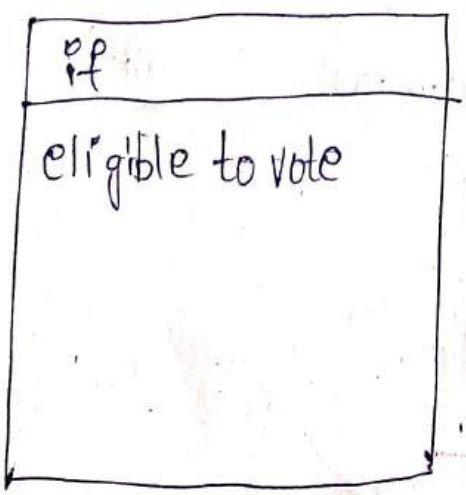
1. if - statement
2. if-else.
3. while
4. do-while.
5. for
6. switch

1) If- statement :- It is used to select a blocks of statements, based on certain condn's becomes true. Otherwise general execution flow continues.

Syntax:-
```
if(condn)
{
==
}
```

Eg:-
```
<html><head><title> if </title></head>
<body>
<script type="text/javascript">
var age = 20;
if(age>18){
document.write("eligible to vote");
}
</script>
</body>
</html>
```

O/P:-



2) If-else :-

If evaluate the content whether the condn is true or false.

Syntax:

```
if (condn)
{ ... // TRUE
}
else
{ // FALSE:
}
```

Eg:-

```
<html>
<head><title> If-else </title></head>
<body>
<script type="text/javascript">
var age=20;
if(age<18)
{
document.write("eligible");
}
else
{
document.write("not eligible");
}
</script></body></html>
```

o/p:

If-else
not eligible

3. while loop:-

It is used to execute a block of code repetly as long as specified condn is true. once the specified condn is false, the loop will be exited (stop).

Syntax:-

```
while(condn)
{
  code
  inc/dec
}
```

1st condn
2nd code

ex:-

```
<html>
<head><title> while </title></head>
<body>
<script type="text/javascript">
var x=1
while(x<10) {                    //code
document.write("<h1>"+x+"</h1>")
x=x+1 → inc
}
</script>
</body>
</html>
```

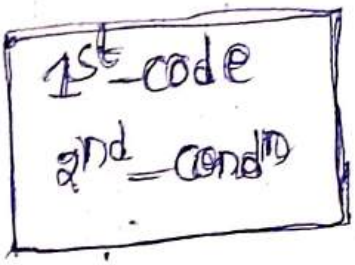
o/p:-

while
1 7
2 8
3 9
4
5
6

4. do-while loop :- The loop will be execute the block of code once before checking if the condn is true or false and then it will repeat the loop as long as the condn is true.

Syntax :-

```
do
{
  ==code
  ___.Inc/dec
}
while(condition)
```



Ex :-

```
<html>
<head><title>do-while</title></head>
<body>
<script type="text/javascript">
var x=1
do {
document.write("<h2>MCA</h2>")
x++;
}
while(x<5);
</script>
</body>
</html>
```

o/p :-

do-while
MCA
MCA
MCA
MCA

Q:- 9/11/2022

5. for loop:-

A set of instructions can be executed repeatly for a specific no. of times, by using for loop.

Syntax:-

```
for(initialization; condition; inc/dec)
{
   code.
}
```

Ex:-

```
<html>
<head><title> for loop</title></head>
<body>
<script type="text/javascript">
var n = prompt("enter n");
var i;
for(i=0; i<=n; i++)
{
document.write("hello welcome
        to Javascript "+"<br>");
</script></body></html>
```

O/P:-

Switch statement:-

It is used to perform different actions, based on different conditions.

→ This statement to select one of many code blocks to be executed.

Syntax:-

```
switch(expression)
{
 case x:
  //code block
  break;
 case y:
  //code block
  break;
 default:
  //code block
}
```

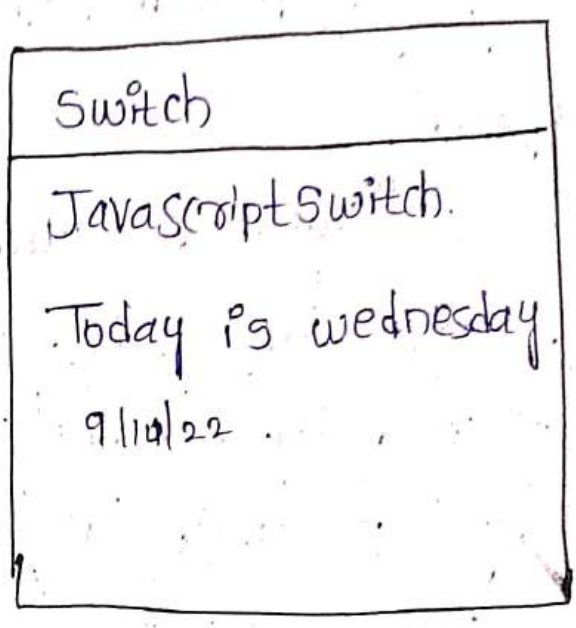
Ex:-

```
<html>
<head><title><switch</title></head>
<body>
<h2>JavaScript switch</h2>
<P id="demo"></P>
<script>
let day;
switch(new Date().getDay())
{
case 0:
    day="sunday";
    break;
case 1:
    day="monday";
    break;
case 2:
    day="Tuesday";
    break;
case 3: day="wednesday";
    break;
case 4: day="Thursday";
    break;
case 5: day="friday";
    break;
case 6; day="saturday";
    break;
}
```

```
document.getelements ("demo").innerHTML= "Today is" + day;

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

o/p:-



* Functions :-

A Javascript fun is a block of code. designed to perform a specific task.

Syntax:- function name → func name (parameter 1, paramter 2)
↓ keyword {

//code to be executed.

A Java script fun defined a 'function' keyword followed by a 'name' (funname) & followed by (parameter / arguments)

Funname → Letters, digits, _ & signs. (same rule as variables).

(Parameter) → parameter name seperated by comma's (,) → (par1, par2, ---)

the code to be executed by the func is placed inside { }.

→ Func parameters are listed inside the paranthesis in the fun def^n.

Function header:-

When JS reaches a header statement the func will stop executing

If the func cause invoke from a statem -ent javascript will return the execute the code after the invoking stmts

→ func often compute a return value the return value. the return value returned back to the colon.

Ex:- Func return:-

```
<html>
<head><title> func </title></head>
<body>
<h2> Javascript func </h2>
<P> This is example for func return</P>
<p id = "demo" </p>
var x = myFunction(4,3)
document.getElementId ("demo").inner
        HTML = x;
function myfunction(a,b).
{ return a*b;
}
</script>
</body>
</html>
```

o/p:-

Func
Javascript func.
This is example for
func return
12

⇒ JS fun is a block of JS code that can be executed when called for.

⇒ Script can be placed in the body or in the <head> section of a HTML page or in both.

i) <Head>section:- (JS placed in <head> section)

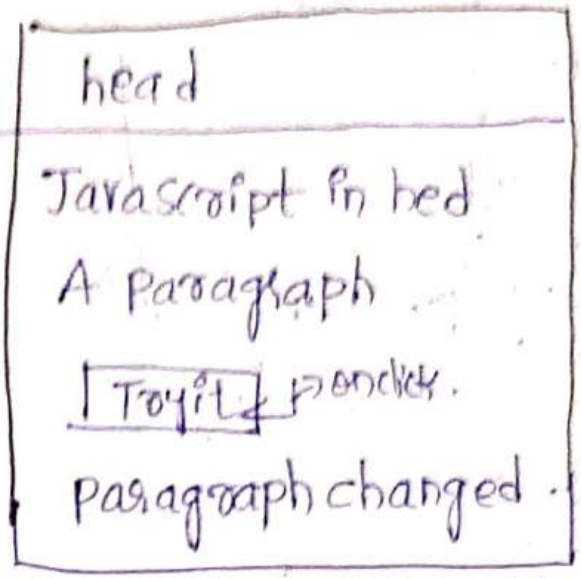
Ex:-

```
<html>
<body><head><title>head</title></head>
<script>
function myfunction()
{
document.getElementById("demo").
        innerHTML = "paragraph changed";
}
</script></head>
<body>
<h2>Javascript in head</h2>
<p id="demo">A paragraph</p>
<button type="button"
onclick="myFunction()">
Try it</button>
```

submit

</body>
</Html>

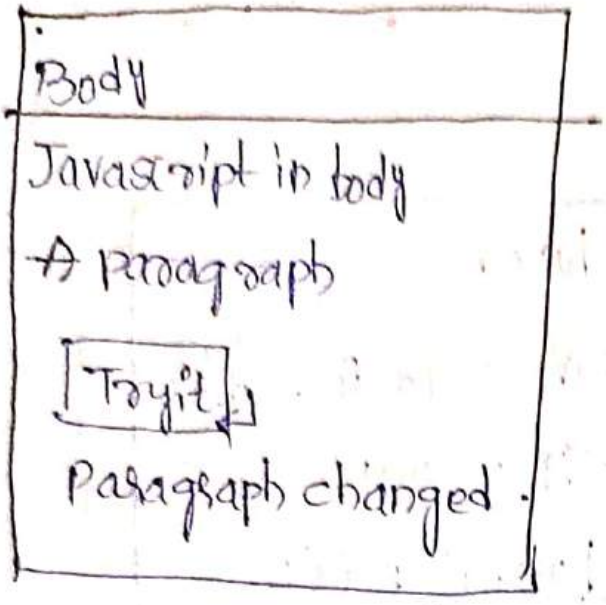
O/P :



ii) Javascript func in <body> section:-

```
<html>
<head><title>Body</title></head>
<body>
<h2>Javascript in body</h2>
<p id="demo">A paragraph</p>
<button type="button" onclick="myfunct
    -ion()">
Tryit</button>
<script>
function myFunction()
{
document.getElementById("demo").
innerHTML="paragraph changed";
}
</script></body></html>
```

o/p:-



Operators:-

1: //Arithemetic operators//

```
<html>
<body>
<Script type="text/javascript">
Var a=10;
var b=20;
Var c="text";
Var linebreak="<br>";
result=a+b;
document.write("addition is=", result);
document.write(linebreak);
result=a-b;
document.write("substraction is=", result);
document.write(linebreak);
result = a*b;
document.write("multipication is=", result)
document.write(linebreak);
result=a%b;
document.write("moduk is=", result);
document.write(linebreak);
result=a%b;
document.write("module is=", result);
```

```
document.write(linebreak);
result = a/b;
document.write("division is=", result);
document.write(linebreak);
result=++a;
document.write("Increament is=", result);
document.write(linebreak);
result=--b;
document.write("decreament is=", result);
</script>
</body>
</html>
```

output:-

addition=30
subtraction is=-10
multiplication is=200
module is=10
division is=0.5
Increment is=11
decrement is=19

2. Comparision operators:-

```
<html>
<body>
<script type="text/javascript">
var a=10;
var b=20;
var c="text";
var linebreak="<br>";
result=a>b;
```

```
document.write("a>b is=", result);
document.write(linebreak);
result1= a<b;
document.write("a<b is", result1);
document.write(linebreak);
result=a==b;
document.write("a==b is=", result);
document.write(linebreak);
result=a>=b;
document.write("a>=b is=", result);
document.write(linebreak);
result=a<=b;
document.write("a<=b is=", result);
document.write(linebreak);
result=a!=b;
document.write("a!=b is=", result);
document.write(linebreak);
</script>
<body>
</html>
```

Output:-

```
a>b is=false
a<b is=true
a==b is=false
a>=b is=false
a<=b is=true
a!=b is=true
```

3. Bitwise operators :-

```
<html>
<body>
<script type="text/javascript">
var a=10;
var b=11;
var linebreak="<br>";
result=a&b;
document.write("a&b is =", result);
document.write(linebreak);
result=a|b;
document.write("a|b is =", result);
document.write(linebreak);
result=a^b;
document.write("a^b is=", result);
document.write(linebreak);
result=~b;
document.write("~b is=", result);
document.write(linebreak);
</script>
</body>
</html>
```

O/P :- a&b is =10
a|b is=11
a^b is=11
~b is=00

4. Assignment operators:-

```
<html>
<body>
<script type="text/javascript">
Var a=3;
Var b=2;
Var linebreak ="<br>";
result = a+=6;
document.write("a+=b is=", result);
document.write(linebreak);
result= a-=6;
document.write("a-=b is =", result);
document.write(linebreak);
result= a*=b;
document.write("a*=b is=", result);
document.write(linebreak);
result= a/=b;
document.write("a/=b is =", result);
document.write(linebreak);
</script>
</body>
</html>
```

O/P:- a+=b is = 9
a-=b is =3
a*=b is=6
a /=b is=3

5. String Operators:-

```
<html>
<body>
<h1> JavaScript Operators </h1>
<p> Adding a number and a string, returns
    a string. </p>
<p id = "demo"></p>
<script>
let x = 5+5;
let y = "5"+5;
let z = "Hello"+5;
document.getElementById("demo").innerHTML=
x+"<br>"+y+ "<br>"+z;
</script>
</body>
</html>
```

O/P:

JavaScript Operators

Adding a number and a string, returns a string.

10

55

Hello5

Logical operators:-

1) Logical AND:-

```
<html>
<body>
<h1> JavaScript Comparison</h1>
<h2> The && Operator (Logical AND)</h2>
<p> The && operator returns true if
both expressions are true, otherwise
it returns false.</p>
<p id ="demo"></p>
<script>
let x=6;
let y=3;
document.getElementById("demo").inner
                    HTML = (x<10 && y>1)
                    + "<br>" + (x<10 && y<1)
</script>
</body>
</html>
```

Output:-

The && operator (Logical AND)
The && operator returns true if both expressions are true, otherwise it returns false.
true
false.

2) Logical OR :-

```
<html>
<body>
<h1> JavaScript comparison </h1>
<h2> the || operator (Logical OR) </h2>
<p> the || returns true if one or both
expressions are true, otherwise it returns
false. </p>

<p id="demo"></p>
<script>
let x=6;
let y=3;
document.getElementById("demo").innerHTML
= (x==5 || y==5)+"<br>"+
(x==6 || y==0)+"<br>"+
(x==0 || y==3)+"<br>"+
(x==6 || y==3);
</script>
</body>
</html>
```

Output :-

JavaScript comparison
The || operator (Logical OR)
the || returns true if one or both expressions are true, otherwise it returns false.

false
true
true

3) Logical NOT:-

```
<html>
<body>
<h2> Javascript comparison</h2>
<p>The NOT operator (!) returns true
for false statements and false for
true statements.</p>
<p id="demo"></p>
<script>
let x=6;
let y=3;
document.getElementById("demo").
                innerHTML=
!(x===y)+"<br>"+ !(x>y);
</script>
</body>
</html>
```

Output:-

JavaScript Comparison.
The NOT operator (!) returns true for false statements and false for true statements.

true
false.

Event Handling:- 10/11/2022

(onclick, on submit).

onclick:-

Html events/ things that happends to the html elements.
when Javascript is used in html pages Javascript can react on these events.
when Javascript code is included in html. Javascript react over these events & allow the execution. The process of reacting over the events is called Event Handling.
→ this Javascript handler the html events via event handlers
→ An event is something that happen -ds when user react with the web page such as when we clicked a link or button. pressed a key in a keyboard.

* html events:-

1. Mouse events
2. Keyboard events
3. form event
4. body event
5. window / document event

1. Mouse events:-

1) Onclick :- when mouse click an elem
-ent.

2) Ondbclick :- when mouse double click
an element.

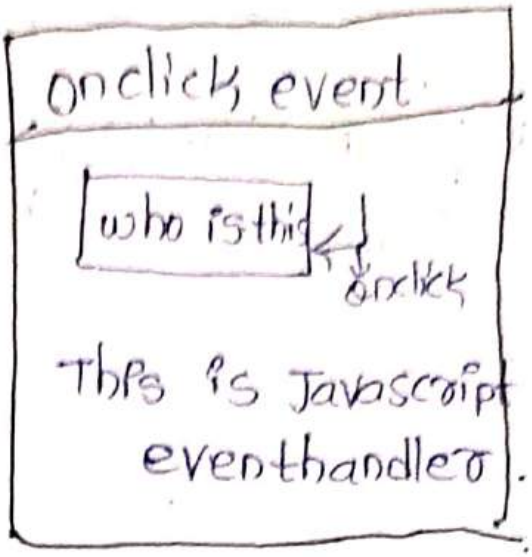
1.x Example for onclick :-

```
<html>
<head><title>onclick event</title>
                    </head>

<body>
<script type="text/javascript">
 function clickevent()
 {
 document.write("this is javascript
   eventhandler");
 }
 </script>
 <form>
 <input type="button onclick=
        "clickevent()" value="whois
                        this">

 </form>
 </html>
```

o/p:-



2. ~~Example for~~ onsubmit:-

This event occurs when a form is submitted.

* Example:-

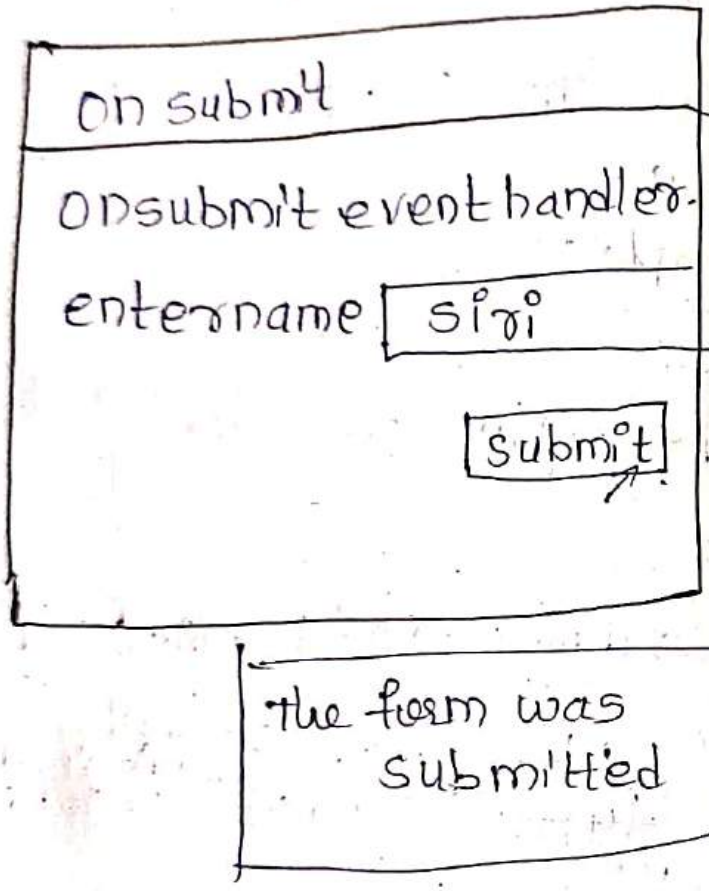
```
<html>
<head><title> on submit </title></head>

<body>
<p> on submit eventhandler </p>
<form> <formaction = "f1.php" onsubmit=
<input type = "submit" onsubmit =
"myfunction">
Entername : <input type = "text"
name = "fname">.
<input type = "submit" value =
"submit">

</form>.
```

```
<Script>
<function myfunction(){
alert("the form was submitted")
</script>
</body>
</html>.
```

O/P:-



Forms :-

Html forms are required to collect different kinds of user input -s. such as, contact details, name, emailaddress, phone no.r.

→ used to pass data to server.

→ Forms contains special elements called controlls like input box, radio box, check box., submit & recheck.

* <form> :-

It is used to create a form.

Attributes :-

1. Action :- specify the web address. or URL. where the data enter in the form is to be stored.

2. Method :- specifies which technical protocal the webserver will use to pass the form data to the program which process it.

1) get method() :- take the data entered in the form & send it to the URL specified in the action attribute for processing. It sends 1KB data. Large a.

2) post method() :- allows to transfer large amount of data.

HTML form elements:-

1) <Input> specify the input field where the user can input or enter data.

attributes:-

1) type:- specify the type of control (radio, password box, text & checkbox).
2) name:- assign a name to i/p control.
3) size:- size of i/p control.
4) maxlength:- max. no. of characters that can be entered.

form controls:-

1. Text field:- It allows to enter in a single line of the text into a form. [Hello]

Syntax:-

<Input type="text" name = "first" "f1" or "f2" size=10>

2. Textarea:- used to enter multiple lines of text within space provided.

Syntax:-

<textarea rows=10 cols=40> content
</textarea>

3. Radio buttons:-

Used to select single option from a group of options.

Syntax:- Ex:-

<input type="radio" name="Gender"
value="male">male

o/p:- omale.

4. Checkbox:-

Used to select multiple options form the list.

Ex:- <input type="checkbox" name=
"language" values="English">
English.

o/p:- ☑ English.

5. Submit:-

Used to send form data to a server

Ex:- <input type="submit" value="submit">

6. Reset:- D: 11/11/22

Used to reset from data

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

7. Password:-

Used to create a password field.

8. Select:-

Used to create dropdown list.

9. Option :- In the option element will only occur with select element.

Ex:- <option> India </option>.

10. Image button:-

It is like a submit button. Used to submit the data in the form to the server.

Ex:- <input type = "image" src = "abc.jpg" align = "center">.

11. Button:-

Used to create a button in a web page.

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

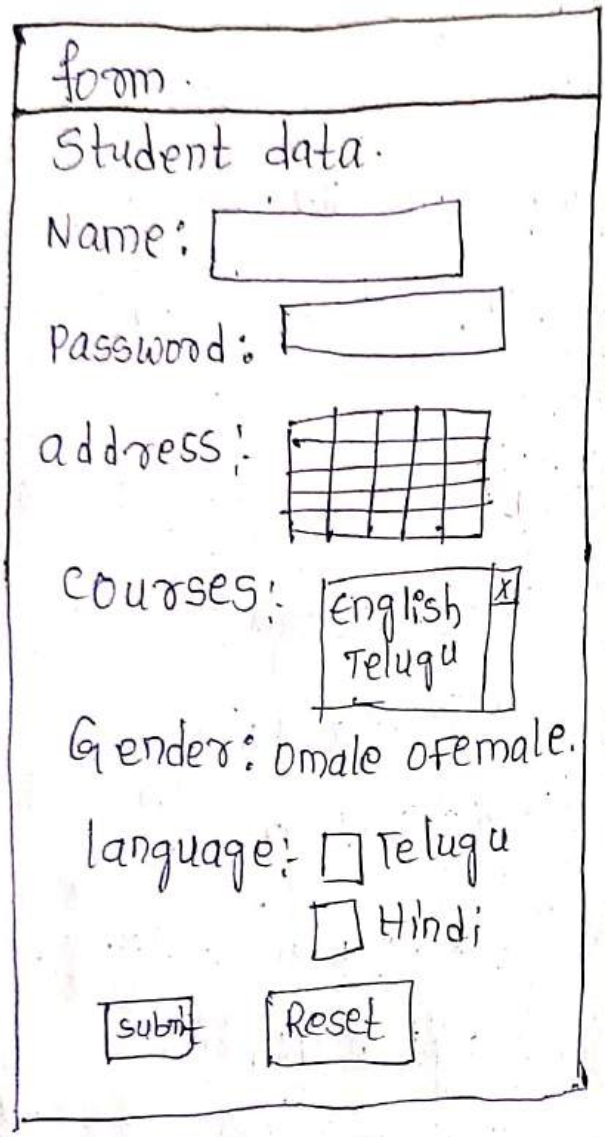
Ex:-

```
<html>
<head><title> Form </title>
</head
<body>
<form> <h1> student data</h1>
Name:<input type="text" name="a"
size = 20><br>
password : <input type="password"
name="pwd"><br>
Address:<textarea rows=5 cols=20>
</textarea><br>
courses:< select name="lang">
<option> English </option>
<option> Telugu </option></select>
Gender : <input type="Radio" name
="Gender"> male
<Input type="Radio" name="gender">
female<br>
Language known: <input type="checkbo
-x" name="a"> Telugu
<input type="checkbox" name="b"> Hindi
```

<Input type = "submit" value = "submit">

<Input type = "Reset" value = "Reset">

o/p:-



→ display the program with tags.

* Introduction to XML :- (storing & transfer of data)

(Extensible Markup Language)

→ XML stands for Extensible Marku
-p Language.

→ XML is a markup Language (like html)

→ XML is s/w & h/w indept.

→ XML is designed to store & transport data.

→ XML was designed to be self-descripti
-ve (sender infor & receiver infor, heading & msg body structure).

→ XML was designed to be human & machine readable. (Pc's, mobiles)

→ XML donot contain pre-defined tags (we can create our own tags (user-defined tags))

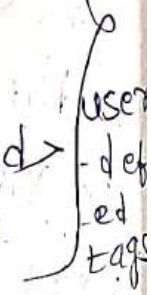
→ XML uses document type def or XML Schema to describe the data.

→ XML is designed to carry data not to display the data.

Syntax:-

```
<?xml version="1.0" encoding=
        "UTF-8"?>

<root>
  <child1>
    <subchild> --- </subchild>
  </child1>
  <child2>
    <subchild> --- </subchild>
  </child2>
</root>
```



→ encoding :- specify the character encoding used in document.

* XML rules :-

→ XML tags casesensitive.

→ XML must start with a letter or _ (underscore)

→ XML can't contain start with XML

→ It can contains letters, no.'s, -, underscore etc.

→ Opening & closing tags must be return the same letter (<name> ... </name>)

→ In XML all elements must be properly nested within each other.

→ XML must have a starting & closing tags. (It shows only tree-structure)

→ XML document must contain one root element i.e the parent of all other elements.

⇒ XML comments:-

1) single line comments:-

<!------>

2) Multi-line comments:-

<!----

--->

⇒ XML Elements:-

An XML element is every-thing from (including the elements <start> to the including the elements <end>).

An element contains

1. Text 2. Attributes & other elements.

* XML Attribute:

→ XML elements can have attributes just like html.

→ Attributes are designed to contain data related to a specific elements

Ex:-

```
<person gender = "female">
```

tag. attribute value.

Imp Difference b/w XML & HTML:-

HTML (.html)	XML (.xml)
1. It stands for Hyper Text Markup Language.	1. XML stands for Extensible Markup Language.
2. It is static in nature (used to display the data).	2. It is dynamic in nature (used to store/transport the data).
3. It is markup language (tags)	3. It provides frame work to define mark-up language.
4. It can ignore small errors.	4. It does not allow errors.

5. It is not case sensitive.	5. It is a case sensitive.
6. Tags are pre-defined tags.	6. Tags are user-defined tags.
7. Tags are used for displaying the data.	7. Tags are used to discribe the data not for displaying.
8. closing tags are not necessary.	8. closing tags are necessary.
9. It doesn't carry data. It just display it.	9. It carries the data from to & from in the database.
10. Additional applications is not required for passing of JS code into the html document.	10. Document object model (DOM) is required for passing JScode & mapping of text.

Example:-

```
<?xml version="1.0"?>
<bookstore>
    <book>
    <author> abc </author>
    <bname> xyz </bname>
```

<price> 950 </price>
<pages> 100 </pages>

</book>

</bookstore>.

↓

O/P:- Same program.

To display o/p we use DTD

D:- 15/11/24

* Structuring the data:-

XML Tree-structure:-

→ XML documents are formed as a elements trees.

→ An XML tree starts with root/ parent elements & branches (child) from the root to child elements.

→ An element can have sub element (child elements)

Syntax:-

```
<root>
  <child>
    <subchild> ---- </subchild>
  </child>
</root>
```

* the terms parent, child & sibling (sub child) are used to describe the relationship blw elements.

* DTD (Document Type Definition) → used for structure of xml document; also Declaration.

→ A 'DTD' defines the structure and the legal elements and attributes of a XML element.

→ An XML document with current syntax is called well-formed.

→ An XML document. validate against a DTD is both well formed & valid.

Syntax for DTD :-

```
<!DOCTYPE element DTD identifier
  [
    Declaration 1
    Declaration 2
    .
    .
    .
  ]>
```

* the purpose of DTD used to define the structure, legal elements & XML elements. attributes.

Ex:-

```
<!DOCTYPE Note                    (Document type → root element)
[
<!ELEMENT Note (to, from, heading, body)>     (child elements)
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

(#PCDATA) → Parseable character data

1. DOCTYPE Note :- Define that the root element of the document is 'Note'.
2. Element Note :- Define that the Note element must contain elements. i.e to, from, heading, body..
3. Element to (#PCDATA) :- Define the "to" element to be of type of "PCDATA"
4. Element from :- Define the "from" element to be of type of "PCDATA"

5. ELEMENT heading :- Define the "heading" element to be of type of "PCDATA".

6. ELEMENT body :- Define the "body" element to be of type of "PCDATA".

* Types of DTD :- .XML

1. Internal (embedded) DTD
2. External DTD

1. Internal DTD :- elements are declared with in the XML files (DTD file + XML files)

Syntax:

```
<! DOCTYPE  root-element
   [element_Declaration]>
                ↓sub elements
```

Ex:- filename.XML :-

```
<?xml version="1.0">
<! DOCTYPE Address [   → root element
                  same
   <! ELEMENT Address (name,   → dtd file
      company, phoneno)>

   <! ELEMENT name (#PCDATA)>
   <! ELEMENT company (#PCDATA)>
   <! ELEMENT phoneno (#PCDATA)
   >
```

<Address>

<name> sfoi </name>

<company> Deloittee </company>

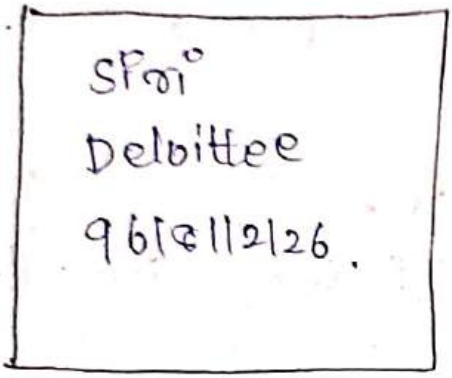
<phone> 9618112126 </phone>

</Address>

]>

xml file.

o/p:-



2. External DTD:- (.xml)

Elements are declared outside XML file, XML is one file & DTD is another file.

Syntax:

<!DOCTYPE root-element SYSTEM "filename">

Ex:- add.xml

<?xml version="1.0">

<!DOCTYPE SYSTEM "add.dtd"

<Address>

<name> sfoi </name>

<company> Deloittee </company>

<phone> 7---- </phone>

<| Address>

add.dtd :-

<!ELEMENT Address(name, company, phoneno)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)>

<!ELEMENT phoneno (#PCDATA)>

]>

o/p :-

*XML namespace :-

prefix.

→ A namespace for the prefix must be defined.

→ the namespace can be defined by an XML ns (name space) attribute in the <start> of an element.

Syntax :- <m:apple xmlns:m="fruit"> (same)

m → prefix

"fruit" → mobile

→ XMLns Ps a coll^n of names that can be used as elements/attributes names in an XML document.

→ An XML used for providing uniquely named elements & attributes in an XML document.

→ XMLNS:prefix:"URI" → uniform Resource locater which identifies the internet domain address.

Eg:- <?xml version="1.0">

<apple> (or) <m:apple>
prefix

<modelname> iphone13 </modelname>

<price> ₹70,000 </price>

</apple> (or) </m:apple>

not same prefixes. <apple> (or) <f:apple>

<imported> simla </imported>

<price>200</price>

</apple> (or) </f:apple>
prefix

* XMLNS advantages :-

→ It avoides duplicate element names & attributes defined in an XML document. Where the XML element defined in one organization product may conflict with the others.

→ With 'NS' elements & attributes can be reused in other documents or schema without any error.

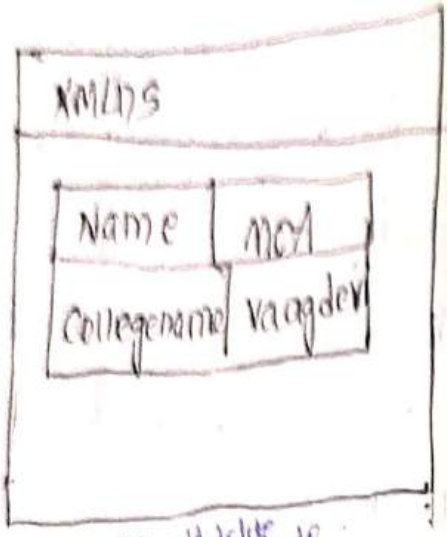
* XMLNS example program :-

a1.html

```
<html>
<head><title> XMLNS </title></head>
<body>
<h:table>
<h:tr>
<h:td> Name</h:td>
<h:td> MCA </h:td>
</h:table>
<f:table>
<f:tr>
<f:td> collegename</f:td>
<f:td> vaagdevi</f:td>
</f:tr>
```

```
</f:table>
</body>
</html>.
```

o/p:-



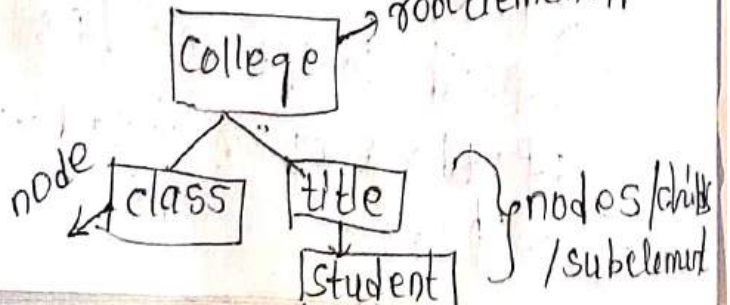
modify, add, delete html code

XML DOM (Document Object Model):

The HTML DOM defines a standard way for accessing & manipulating Html documents. The DOM is a platform & language nutral interface that allows programs & scripts to dynamically access & update the content & structure & style of a document.

→ The HTML DOM is a standard for how to get, change, add, delete HTML elements. It is a Tree-structure.

Ex:-



→ DOM is a collection of nodes/sub elements/childelements or places of info organised in a hierarchy (tree structure)

eg:
```
<college>
  <class> ... </class>
  <title> student content </title>
</college>
```

*2) HTML DOM :-

ex:-
```
<html>
  <body>
  <h id="demo"> this is heading</h>
  <p> this is paragraph</p>
  <script>
  document.getElementById("demo").
          innerHTML = "Hellowoorld!";
  </script>
  </body></html>
```

O/P :-

this is heading
this is paragraph.
Hellowoorld!

The programming interface to the DOM is defined by a standard Properties & methods.

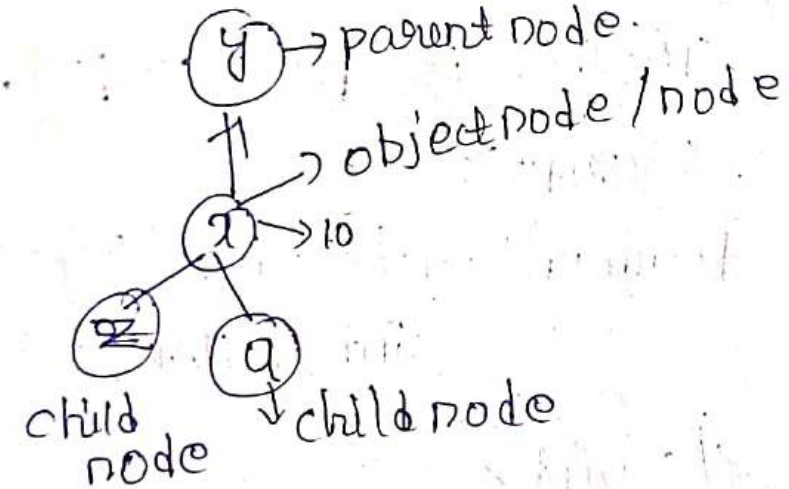
* Properties :

Property is a value that you can get/set (student/node/child element)

* 1) x.nodeName → the name of (x)

2) x.nodevalue → x → value of (x)
(x=10)

3) x.parentNode → the parent node of (x)
→ x → y



4) x.childNodes → z, a

5) x.attributes → x

* Methods:

It is an action. (you can do add, delete, update, modify)

1. document.write → To print the o/p
2. document.getElementById → find an element by element Id.
3. document.getElementByTagName (name) → get odd elements with a specified tag name.
4. document.getElementByClassName (name) → find elements by class name.
5. X.appendchild (node) → Insert a childnode to X.
6. X.removechild (node) → remove a childnode from X.

* changing html elements:-

1. element.innerHTML = new html content :-
Change the inner Html of an element
2. element.attribute = newvalue :-
Change the attribute value of an html element.

3. element.style.property = new style
:-
change the style of an html element;

* Adding & Deleting elements :-

~~I. Adding~~ :-

1) document.createelement(element)
→ create an html element.
2) document.removechild(element)
→ remove an html element.
3) document.appendchild(element)
→ add an html element.
4) document.replacechilld(new, old)
→ replace an html element.
5) document.~~bolght~~ write(text)
→ Write into the html elements.

* XML schema :- (XSD) structure of XML document.
→ It is commonly known as XML schema Definition (XSD)
→ It is used to describe variable & content of XML data.
→ An XML schema describes the structure of an XML document.
→ It is a method of expressing constraint about XML documents.

Purpose of XML Schema :-

→ Used to define the legal building blocks of an XML document.
→ the elements & attributes that can appear in a document.
→ the no. of child elements.
→ Data types for elements & attributes
→ Default & fixed values for elements & attributes.

U-II

D:- 17/11/22

1) Server-Side programming (PHP)

→ PHP stands for Pre-processor Hypertext (or) Hypertext pre-process

-or

→ PHP is a open source., general purpose interpretted, Object oriented, server side scripting language developed by "Rasmus Lerdorf" in 1994.

→ free to download & use (open source)

→ Any type of applications to run (general purpose)

→ It without compilation (Interpretted)

→ It supports class & object (object oriented)

→ To run PHP program to serveside (serverside scripting language)

→ PHP scripts are executed on server side. PHP can be embedded into HTML

↓
PHP code is included in HTML code.

→ PHP file can contain text, html, CSS, JS & PHPcode.
→ PHP code is executed on the server & the result is returned to the browser as a plaintext.
→ PHP file have extension .PHP.
→ PHP can generate dynamic page content (server-side).
→ PHP can add, delete, modify data in your database.
→ PHP can create, open, read, write, delete & close files on the server.
→ PHP can collect form data.
Ex:- online application form.
→ PHP can used to control user access
Ex:- Online Banking, online payment.

*Syntax for PHP code:-

<? php → open tag
//code
?> → end tag.

→ The php scripts starts with <?php & end with ?>

→ A php file normally contains html tags & some php scripting code.

→ php statement end with a semi-colon (;)

* php Comments :-

1) Single line comment //-----
2) Multi-line comment /* ----- */

Example :-

```
<html>
<head> <title> php </title> </head>
<body>
<h1> my first php </h1>
<?php → start tag
echo " Hello world! "; → code.
?> → end tag
</body>
</html>
```

same as print statement for print o/p.

o/p :-

php
My First php
Hello world!

PHP

* Features of php:-

1) Simple:- It is very simple & easy to use. compare to other scripting languages.

2) Interpretted:- There is no need for compilation. It is an interpretted language.

3) Faster (performance):- php script is executed much faster than other scripting languages.

4) Open source:- It is free of download & use.

5) Platform indept:-
php code will be run any platform (Linux, unix, max, windows)

6) Case sensitive:- php is case sensitive language at the time of variable declaration.

7) Embedded:- php code can be easily embedded within html tags & scripts.

8) Database support:-
php support all the leading data -bases (mysql, ODBC, oracle, sql etc)
9) security:- php is a secure language to develop the webpage.
10) Error reporting:- "php" has some predefined error reporting constant -s to generate a warning / error notice..
11) Real time access monitoring:-
php provides access logging by creating the summary of accessing for the user.

2)* Declaring variables:-
Variables are containers for storing information.
→ In a php a variable starts with the "$" followed by the name of the variable ($x=10)
→ variable is a name of storage location or variables are used to store data (string / Text / no. etc)

→ php is a loosely typed language so, we do not need to declare the datatypes of the variables. It automatically analysis the values and makes conversions to its correct datatype.

→ Assignment operator (=) is used to assign a values to a variable.

Ex:- 1. $x = 10.
2. $name = "abc".

→ If the value is not assign to variables then by default the value is null.

Q:- 18/11/22

* Rules:-

1. php variable start with ($) dollar sign followed by the variable name.

Ex:- $x = 10.

2. variable name must start with letter or underscore (_).

Ex:- 1. $x = 10;
2. $_x = 10;

3. variable name must not start with number. ($-123x=10) ✗

4. variable name can't contain spaces.

Ex:- $ x=10 (✗)

5. php variable name can contain alpha numeric value & underscore.

Ex:- 0-9, A-Z, _

6. php variables are case sensitive.

Ex:- $name & $NAME.

Ex:-
```
<?php
$txt="Hello world!";
$x=5; $y=10.5;
echo $text; echo "<br>";
echo $x; echo "<br>";
echo $y;
?>
```

O/P:-

Hello world!
5
10.5

Php Datatypes :-

Variable can store data of diff. -rent types & diff datatypes can do different things.

php supports the following datatypes

→ String :-

A string is a seq of characters

Ex :- $txt = "Hello world!";

A string can be any text inside quotes. you can use single / double quotes.

→ Integer :-

An Integer is a datatype. is a non-decimal number.

Ex :- $x = 5;
var_dump($x);
↓ fun

o/p :- int~~eger~~(5)

→ float :-

Is a no. with decimal point.

Ex :- $x = 10.5;
var_dump($x);

o/p :- float(10.5)

php Boolean:-

A boolean represents two possible states (TRUE/FALSE).

eg:- $x = TRUE;
$y = FALSE;

* php Array:-

An array stores multiple values in one single variable.

eg:- <?php
$cars = array("volvo", "BMW", "Toyota");
var_dump($cars);
?>

(0, 1, 2)

o/p:- array(3) { [0] => string (5) "volvo"
[1] => string (3) "BMW"
[2] => string (6) "Toyato"

* php object:-

An object is a instance of class.

eg:- <?php
class student {
function marks_cal() {
echo "Display marks";
} }

new object.
$ student1 = new student;
$student1 => marks_cal();
?>

o/p:- Display marks.

* php nullvalue :-

Is a special datatype, which can have only one value "null".

→ A variable of datatype null is a variable that has no value assign to it.

Ex:- $x=10; if y o/p:- x=10 int
$x=NULL
var_dump($x);

o/p:- null.

* Resource :-

It is the storing of a reference to funcs & resources external to php.

Operators in php :-

operators are used to perform opers on variables & values.

ex: a + b (operands (variables, values))
↓ operator or a + 5

8 Types :

1) Arithmetic operator
2) Assignment operator
3) Comparision operator
4) Increment / decrement operator.
5) Logical
6) string
7) Array
8) Conditional Assignment.

1) Arithmetic operator :-

i) addition (+) → $x + $y.

ex :-
```
<? php
$x=10; $y=6;
echo $x+$y;
?>
```

o/p :- 16.

ii) Substraction(-) → $x-$y

Eg:- <?php
$x=10, $y=6;
echo $x-$y;
?>

o/p:- 4

iii) Multiplication(*) → $x*$y

Eg:- <?php
$x=10; $y=6;
echo $x*$y;
?>

o/p:- 60.

iv) Division(/) → $x/$y

Eg:- <?php
$x=10; $y=6;
echo $x/$y;
?>

o/p:- ~~6~~ 4

7/4 1 5/6
6)10(1.6
6
40
36
4

v) Modulus (%) → $x%$y

Eg:- <?php
$x=10; $y=6;
echo $x%$y;
?>

o/p:- 4.8

vi) exponentiation → $x ** $y;

Ex:- <?php
$x=10; $y=6;
echo $x**$y;
?>

o/p:-

2) Assignment operators:-
Used to assign a values to a variables

i) = → $x=$y o/p:-10

Ex:- <?php
$x=10;
echo $x;
?>

ii) x+=y → x=x+y →
<?php
$x=20; $x+=100;
echo $x;
?>

o/p:-120

iii) x-=y → x=x-y →
<?php
$x=50; $x=30;
echo $x;
?>

o/p:-20

iv) x*=y → x = x*y

Ex:- <?php
$x=5; $x*=6;
echo $x;
?>

o/p:- 30.

v) x/=y → x = x/y →
<?php
$x=10; $x/=5;
echo $x;
?>

o/p:- 2.

vi) x%=y → x = x%y →
<?php
$x=15; $x%=4;
echo $x;
?>

4) 15 (3)
12
(3)

o/p:- 3

3) Comparision operators:- / Relational
used to compare two operands & returns a boolean value.

1) == → equal → $x==$y
<?php
$x=100; $y=100;
var_dump($x==$y);
?>

o/p:- TRUE

2) < → lessthan → $x < $y

Ex:- <?php
$x=10; $y=50;
var_dump($x<$y);
?>

o/p:- TRUE.

3) > → greaterthan → $x > $y

Ex:- <?php
$x=100; $y=50;
var_dump($x>$y);
?>

o/p:- TRUE

4) <= → less than or equal to → $x <= $y

Ex:- <?php
$x=50; $y=50;
var_dump($x<=$y);
?>

o/p:- TRUE.

5) >= → greaterthan or equal to.

Ex:- $x >= $y → <?php
$x=50; $y=40;
var_dump($x>=$y);
?>

o/p:- false. (50>=40)

6) != → $x != $y

ex:-
```
<?php
$x=100; $y="100";
var_dump($x!=$y);
?>
```
o/p:- true (100 not equal to "100" - string so not equal to)

4) increment/decrement operators :-

i) pre increment :-

(++$x) - 1st inc then add.

```
<?php
$x=10;
echo ++$x
?>
```

o/p=11

ii) post increment :-

($x++)

```
<?php
$x=10;
echo $x++;
```

o/p:- 10.

iii) pre decrement :-

(--$x)

```
<?php
$x=10;
echo --$x;
?>
```
o/p:- 9

iv) post decrement :-

$x--

```
<?php
$x=10;
echo $x--;
?>
```
o/p:-10

String Operators :- concatenation (adding strings)

1) concatination .

$txt1 . $txt2

Ex :- <?php
$txt1 = "Hello";
$txt2 = "world!";
echo $txt1 . $txt2;
?>

O/P :- Helloworld!

2) concatenation assignment :- .=

$txt1 .= $txt2

Ex :- <?php
$txt1 = "Hello";
$txt2 = "world";
~~echo~~ $txt1 = $txt2;
echo $txt1;
?>

O/P :- Hello world! ($txt1 $txt2)

6) Logical operators :-

1) Logical AND :- (&&)

$x && $y

Ex :- ?php
$x = 50; $y = 30;
if ($x == 50 && $y == 30)
echo "&& success";
?>

Truth Table :-

x	y	&&
T	T	T
T	f	f
F	T	f
f	T	f

(∵ x = 50 ✓ x == 50 TRUE)

↓ or success.

O/P :- && success

ii) Logical OR (||) → $x || $y.

Ex:- ?php
$x=50; $y=30;
if($x==50 || $y==10)
echo "|| Success";
?>

x	y	||
T	T	T
T	F	T
F	T	T
F	F	F

o/p:- || success

iii) Logical NOT (!$x)

$x !$y → ?php
$x=50; $y=30;
if(!$z) false → o/p:- opposite i.e true
echo "! success";

o/p:- success.

7) Array operators:-

Used to compare two arrays.

i) Union (+):- $x+$y.

?php
$x = array("a" => "red", "b" => "green");
$y = array("c" => "blue", "d" => "yellow");
echo($x+$y);
?>

o/p:- array("a" => red, "b" => green,
"c" = blue, "d" = yellow)

ii) equality (==) → $x == $y.

→ $x == $y

```
?php
$x = array ("a" => "red", "b" => "green");
$y = array ("c" => "blue", "d" => "yellow");
var_dump($x == $y);
?>
```

o/p:- bool(false).

iii) Inequality (!=) → $x != $y.

ex:- ?php

```
$x = array ("a" => "red", "b" => "green");
$y = array ("c" => "blue", "d" => "yellow");
var_dump($x != $y);
?>
```

o/p:- bool(true)

8) Conditional Assignment operator:-

The conditional Assignment operator are used to compare set a value depending on conditions.

Syntax:- $var = (condn) ? value1 : value2;

ex:- $ = -12

echo($x > 0) ?

o/p:- the nos is negative.

Php operators examples:-

1) Arithmetic operators:-

```
<?php
$x=29;
$y=4;
echo ($x+$y), "\n"; (or) "<br>";
echo ($x-$y), "\n";
echo ($x*$y), "\n";
echo ($x/$y), "\n";
echo ($x%$y), "\n";
?>
```

o/p:-
33
25
116
7.25
1

2) Assignment operators:-

```
<?php
$y=75;          → Simple assign
echo $y, "<br>";
$y=100.
$y+=200         → Addition.
echo $y, "<br>";
$y=70;          → substraction
$y-=10;
echo $y, "<br>";
```

```
$y=30;
$y*=20;          → Multiplication
echo $y,"<br>";
$y=100;
$y/=5;           → Division
echo $y,"<br>";
```

o/p: 75
3000
60
600
20
0

```
$y=50
$y%=5; → remainder
echo $y;
?>
```

3) Comparision operators:-

```
<?php
$a=80; $b=50;
var_dump($a<$b); echo "<br>";
var_dump($a>$b); echo "<br>";
var_dump($a<=$b); echo "<br>";
var_dump($a>=$b); echo,"<br>";
var_dump($a!=$b);
?>
```

o/p: bool(false)
bool(true)
bool(false)
bool(true)
bool(true)

4) Increament/ Decreament operators:-

```
<?php
$x=2;
echo ++$x; "first increments then prin
        \n", "<br>";
echo $x, "<br>";
$x=2;
echo $x++, "first prints then increm
   -ts\n", "<br>";
echo $x, "<br>";
$x=2;
echo --$x, "first decrement then print
        \n", "<br>";
echo $x, "<br>";
$x=2;
echo $x--; "first prints then decre
        -ments\n", "<br>";
echo $x;
?>
```

o/p:- 3 first increments then prints
3
2 first prints then increments.
3
1 first decrements then prints.
1
2 first prints then decrements.
1

5) Logical (or) Relational operators:-

```
<?php
$x=50;
$y=30;
if($x==50 && $y==30)
echo "&& success","<br>";
if($x==50 || $y==20)
echo "|| success","<br>";
if(!$z)
echo "! success";
?>
```

o/p:- && success
|| success
! success

6) String operators:-

```
<?php
$x="web"; $y="Technology"; $z="Lab";
echo $x.$y;
$x.=$y.$z;
echo $x;
?>
```

o/p:- webTechnologyLab
webTechnologyLab

7) Array operators:-

```
<?php
$x=array("k"=>"car", "l"=>"Bike");
$y=array("a"=>"train", "b"=>"plane");
var_dump($x+$y); echo "<br>";
var_dump($x==$y); echo "<br>";
var_dump($x!=$y);
```

?>

o/p:- array(9) { ["k"] => string(3) "my"
["d"] => string(9) "Bike"
["a"] => string(5) "Boats"
["b"] => string(5) "plane" }
bool(false)
bool(true)

8) Conditional (or) Ternary operators:-

```
<?php
$x=-12;
echo ($x>0)? "the no.r is positive":
     "the no.r is -ve";
?>
```

o/p:- the no.r is negative.

10:21|11|22

*Control structure:-

1) If:-

Syntax:-

```
if(condition)
{ //code
}
```

Ex:-

```
<?php
$age=50;
if($age>30){
echo "your age is greater
than 30";
}
?>
```

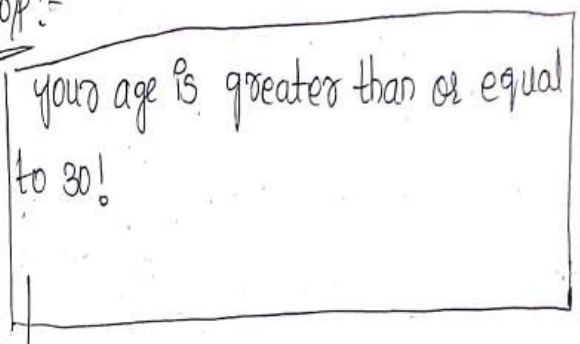
o/p:- your age is greater than 30!

2) else

Syntax:-

```
if(condn)
{
//code is executed if the expression
 evaluates for true
}
else{
//code is executed if the expression
 evaluates to false
}
```

op:-



Ex:-

```
<?php
$age=50;
if($age<30){
echo "your age is less than 30!";
}
else{
echo "your age is greater than (or)
 equal to 30!";
}
?>
```

3) else-if:-

Syntax:-

```
if(expression1){
//code is executed if the exp1 evalu
-ates to true
}
elseif(expression2){
//code is edecuted if the exp2
evaluates to true.
}
elseif(expression 3){
//code is executed if the exp3
evaluates to true
}else{
//code is executed if the expression1
exp2, exp3 evaluates to false;
a default choice.
}
```

Eg:-

```
<?php
$age=50;
if($age<30){
echo "yourage is lesstban 30!%";
}
else if($age>=50 && $age<=40){
} echo "your age b/w 30 & 40!!";
```

```
elseif($age>40 && $age<50)
{
echo "your age is b/w 40&50!";
}
else{
echo "your age is greater than 30!";
}?>.
```

(checkout And Truth table)

o/p:- your age is greater than 30!

4) switch statement:-

```
Syntax:- switch(condn){
        case1: // block of code to be
                executed
               break;
        case2: //block of code to be
                executed
               break;
        .
        .
        .
        default: //default block code
        }
        ?>
```

Eg:- ?php
$today="monday";
switch($today){
case1: echo "saturday";
break;
case2; echo "sunday";
break;
case3: echo "Monday";
break;
default: echo "have a nice day";
}
?>

O/P:- Monday

5) whileloop:- repeating

while(condition){
// statement
}

Eg:- <?php
$i=0;
while($i<=5){
echo $i." ";
$i++;
}

O/P:- 0 1 2 3 4 5

6) DO-while loop:-

do{
// statement
} while(condn)

Eg:- <?php
$i=0;
do{
echo $i." ";
$i++;
}
while($i<=5)
?>

O/P:- 0 1 2 3 4 5

7) For loop:-

```
for(init; condn; updation){
//stmts
}
```

Ex:-
```
<?php
for($i=0; $i<=5; i++){
echo $i." ";
}
?>
```

O/P:- 0 1 2 3 4 5

8) Foreach loop:-

Syntax:-
```
foreach($myarray as $element){
//stmt
}
```

Ex:-
```
<?php
$myarray = array("Hello", "world");
foreach($myarray as $element)
{
echo $element;
}?>
```

O/P:- Helloworld

Strings:- collection of characters

1. strlen():- this fun returns the length of a string.

Ex:- <?php
echo strlen("Hello world!");
?>

O/P:- 10

2) str_word_count:- the fun counts the no. of words in a string

Ex:- <?php
echo str_word_count("web Technology Lab");
?>

O/P:- 3

3) strrev():- Hi → iH

fun reverse a string

Ex:- <?php
echo strrev("Technology");
?>

O/P:- ygolonhceT.

1) strpos() :- Position

Fun searches for specific tex within a string.

Ex:- <?php
echo strpos("Helloworld", "world");
?>
O/P:- 5

search

2) str_replace :- This fun replaces some characters with some other characters in a string

Ex:- <?php
echo str_replace("world", "Dolly", "Helloworld");
?>

replace

O/P:- Hello Dolly!

Q:- 23/11/22

Arrays in php:-

Are used to store multiple values in one single variable.

→ It is a special variable that can hold more than one value at a time.

Ex:-

i) without an array:-

```
$car1 = "volvo";
$car2 = "BMW";
$car3 = "Toyata";
```

ii) using an Array [0] [1] [2]

```
$cars = array("volvo", "BMW", "Toyata");
```

(array → keyword; values)

→ An array can hold many values, under one single name.

→ values can be accessed by index no. or name.

Creating an Array in php:-

syntax:-

```
$cars = array("volvo", "BMW", "Toyata");
```

length of array:-

count($cars) → 3

Types :-

1) Indexed Array
2) Associative Arrays
3) Multidimentional Arrays.

1) Indexed Arrays :- (Arrays with numeric index)

→ the index can be assign automatically
→ Index always start with 'zero' ⇒ [0]

Ex :-

```
<html><body>
<?Php
$cars = array ("volvo", "BMW", "Toyato");
echo "I like". $car[0]. ", ", $cars[1].
'and" $cars[2]. ";" ;?>
?>
```

o/p. I like volvo ", ", volvo BMW and Toyato"."

2) Associative Arrays :- (key+value → pairs / named keys)

used to named keys that you assign to them (array with named keys).

ex:-

```
<html><body>
<?php
$age=array("a"=>"35","b"=>"28");
echo "a is ".$age['a']." years old";
?>
```

(=> keyword. "a" key, "35" value.)

o/p:- a is 35 years old.

3) Multidimensional Arrays :-

Array containing one or more arrays.

ex:-
```
$cars=array("Volvo","22","8");
      array("BMW","25","13");
      array("Toyato","5","12"));
```

o/p:-

Volvo	22	8
BMW	25	13
Toyato	5	12

Sorting arrays:-

The elements in an array can be sorted in alphabetical / numerical order (ascending order / descending order)
small → Big

sort():-

sort arrays in ascending order.

Ex:- <?php
$cars = array("volvo", "BMW", "Toyato");
sort($cars)
?>

o/p:- BMW
Toyato
volvo.

rsort():- sort arrays in descending order.

Ex:- <?php
$cars = array("volvo", "BMW",
"Toyato");
rsort($cars)
?>

o/p:- volvo
Toyato
BMW.

PHP functions :-

A fun is a block of stmts that can be repeatedly in a program, A fun will be executed by a call to the fun.

Syntax :- keyword.

```
function funname() {
//code
};  }
```

Ex :- A funname must start with a letter or underscore.

→ Fun names are not case sensitive.

→ without parameters

Ex :- <?php keyword funname.

```
function writemsg() {
  echo "Hello world!"; — code.
}
writeMsg();.1 → call to the fun.
?>
```

o/p :- Hello world!

→ Func with parameters:-

```
<?php
function add($n1, $n2){
  $num = $n1 + $n2;
  echo "sum of two no.r's is" $num;
}
add(10;20);
?>
```

o/p:- sum of two no.r's is 30.

→ Fun returning values:-

```
<?php
function sum(int $x, int $y){
  $z = $x+$y;
  return $z;
}
echo "5+10=". sum(5,10). "<br>";
echo "7+13=". sum(7,13). "<br>";
echo "2+4=". sum(2,4);
?>
```

o/p:- ~~15 sum(5,10)~~ 5+10=15

~~20 sum(7,13)~~ 7+13=20

~~6 sum(6)~~ 2+4=6.

turn 4 pgs D:- 24/11/22

Reading data from webform

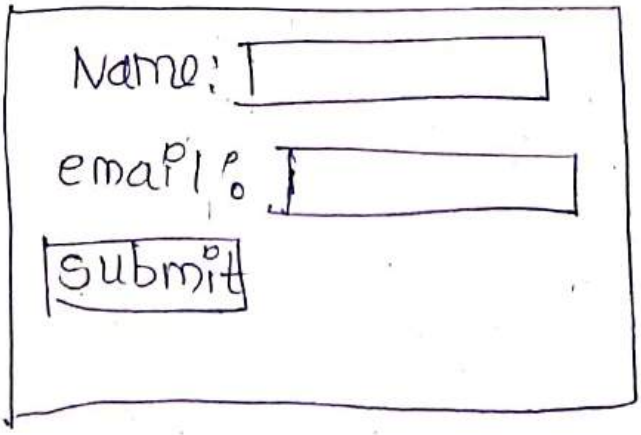
Controlls- (form, attributes, methods, form control)

Ex:- form.html

```
<html><head><title> php form</title>
</head>
<body>
<form action="welcome.php" method="post">
Name:<input type="text" name="name">
<br>
email:<input type="email" name="email"><br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

(another program passing; webaddress; post / get)

o/p:-

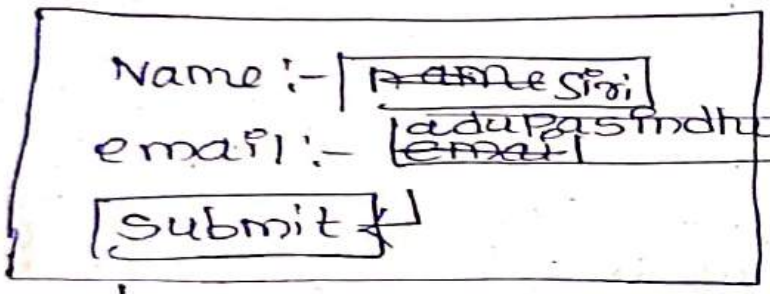


welcome.php

same as before my underline.

```
<?php
$name = $_POST["name"]
$email = $_POST["email"]
 echo "welcome". $name. "your
      email id is". $email;
 ?>
```

o/p:-



↓

"welcome" siri "your email id" is adupasrndhu.oj@gmail.com

PHP sessions and cookies:-

→ PHP sessions :(store & pass info)

A session is way to store information (in variables) to be used across multiple pages. A session is used to store & passes info from one page to another page temporarily (untill user close the website:- gmail, FB, whatsapp).

→ Info stored in server side.

→ Why use session in PHP:-

when you work with any application you open it do some changes & close it. this is like a session.

→ session variable solve this problem by storing user info to be used across multiple pages.

Start a PHP session:-

A session is start with "session_start()" or "bool session_start()".

→ Session variables are set with the PHP "global variables".

PHP variable declaration :-

php session "$_SESSION" variable.

Eg :- demo1.php

```
<?php
session_start();
?>
<!DOCTYPE html>
<html><body><?php
$_SESSION["favcolor"]="green";
$_SESSION["favanimal"]="cat";
echo "Session variables are";
?></body></html>
```

session start

variable declaration.

variables.

To print variables :-

demo2.php

```
<?php
session_start();
?>
```

```
<!DOCTYPE html>
<html><body>
<?php
 echo "favorite color is" . $_SESSION[
    "favcolor"] . "<br>";
echo "favorite animal is" . $_SESSION
   ["favanimal"];
 ?></body></html>
```

O/P:-

~~Favorite c~~
session variables are:
"favorite color is" green
"favorite animal is" cat.

* Destroy a php session: D: 25/11/2022

To remove all global session variables and destroy a session use session_unset() & session_destroy().

eg:-

```
<?php
session_start()
?>
<!DOCTYPE html>
<html><body>
<?php
session_unset(); -> remove all session variable
session_destroy(); -> destroy the session.
?>
</body>
</html>
```

PHP filters :-

PHP filters are used to validate & filter data coming from insecure sources like user i/p.

→ PHP filters are used to validate & sanitize external i/p.

* **Validating Data:-**
  If the data is in proper form.

* **Sanitizing Data:-**
  remove any illegal character from the data.

* **PHP Filter Functions:-**
  Needed for checking user i/p (clt side) & is designed to make data validation easier & quicker

1) **(PHP_var():) & (filter_var()) :-**
   filter a single variable with a specified user.

2) **filter_var_array() :-**
   filter a multiple/several variables with same/different filters.
   the specified user

3) filter_input():-

Get one i/p variables & filter it.

4) filter_input_array :-

Get several i/p variables and filter them with the same or different filters.

Eg:- → Sanitizing a string → $str
→ validate an integer → $int
→ validate an ip address → $ip
→ validate email → $email.

Eg:-

```
<!DOCTYPE html>
<html><body>
<?php
$int=100;
if(!filter_var($int, FILTER_
        VALIDATE_INT)
{
 echo("Integer is valid"); else{
echo ("Integer is not valid); }
?>
</body></html>
```

o/p:- Integer is valid

Working with Database:-

Mysql is the most popular database system used with PHP.
→ Mysql is a database system used on the web that runs on a server.
& It is ideal for both small & large applications.
→ It is free to download & use.
→ mysql & mysql uses standard sql.
→ It is very fast, relable & easy to use.
→ It is develop, distributed & support by "Oracle corporation".

*Database query:-

A query is a question/request.
Ex:- select lastname from employee;
→ PHP create a mysql database.
Ex:- syntax:- Create database databasename;
→ Create table:-
create table tablename(attribute1 datatype(size), attribute2 data type(size));

Insert values into tablename:-

Insert into tablename values
(att1, att2, - - -);

Reading data from webform controls (text, radio, list etc).

Form controls - Textbox, Textarea, Radio, checkbox, List. Buttons.

→ HTML forms are used to pass data to a server

→ Forms contains special elements called controls like, Input box, Radio button, checkbox, submit, reset, password - - etc.

<form> tag used to create form.

Attributes:-

1) Action:- specifies the web address (or) URL where the data entered in the form is to be stored.

2) Method:- specifies which technical protocal the webserver will use to pass the form data to the program which process it.

Two methods are available:-

1) GET method:-

The GET method simply takes the data entered in the form & send it to the URL specified in the action attribute for processing.

→ It sends 1 kb (100 character) data only

2) POST method:-

The POST method allows to transfer large amount of data.

Form elements:-

<Input tag>:- specifies an input field. where the user can input or enter data.

Attributes:-

1) Type:- Specifies type of control.

Ex:- Radio, checkbox, text. etc.

2) Name:- Assign a name to the i/p control.

3) size:-

sets the horizontal width of the field.

4) maxlength:- the max no. of character that can be entered.

Form controls:-

1) Text field:-

The textfield allows to entering a single line of text into the form

Ex:- <input type="text" name="first" size=10>.

2) <textarea>:-

Used to enter multiple line of text within space provided.

→ Used to create textarea.

<text area rows=10 col=407----

</textarea>.

3) Radio button:- used to select a single option from a group of options

Ex:- <input type = "radio" name="Gender" value = "male" > male.

4) checkbox:- used to select multiple options from the list.

<Input type = "checkbox" name = "Lang" value="English"> English

5) Button:- Used to create a button in a webpage, not perform any action user has to give the value to perform action.

<Input type = "button" value = " " >.

3) List :- (or) Select :-

Used to create drop-down list.

<option> the option element can only occur with a select element. It represent one choice.

<select = lang>

<option> ---- </option>.

Form controls:-

1) Textfield:-
The textfield allows to entering a single line of text into the form
Ex:- <input type="text" name="first" size=10>.

2) <textarea>:-
Used to enter multiple line of text within space provided.
→ Used to create textarea.
<text area rows=10 col=40>----
</textarea>.

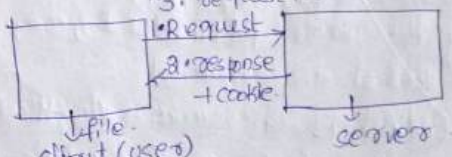
3) Radio button:- Used to select a single option from a group of options
Ex:- <input type="radio" name="gender" value="male">male.

4) checkbox:- Used to select multiple options from the list.
<input type="checkbox" name="Lang" value="English">English

5) Button:- Used to create a button in a webpage, not perform any action user has to give the value to perform action.
<input type="button" value=" ">

6) List:-(or) select:-
Used to create drop-down list.
<option> the option element can only occur with a select element. It represent one choice
<select - lang>
<option>----</option>.

Php Cookies: (file - coin of data).
Cookie is a generally used to identify a user(client). Cookies are text files stored on the client computer.



→ Php transparently supports HTTP cookies you can both create & retrieve cookies values.
→ User request to server for a site.
→ Server script sends a set of cookies to the browsers.

Ex: name, rollno, contact no, age.
Browsers stores this information on local machine for future use. When next time browser sends any request to browser then it sends those cookies information to the server & server uses that information to identify the user.

Create cookies in php:- (.php)

1) setcookie():- (In

Syntax:-

setcookie(name) value, expire, path, domain, server, httponly)

Only the name, pasameter is required. all other are optional.

A cookie is a small file that the server embeds on the user's computer

Ex:- (To create & modify the cookie)

```
<?php
$cookie_name = "user";
$cookie_value = "vaag";
setcookie($cookie_name, $cookie_value, time() + (86400*30), "/");
?>
```

modify cookie vaag - you can give anything

1day → 60x60x24

M S h

```
<!DOCTYPE HTML>
<html><body>
<?php
If (!isset($_COOKIE[$cookie_name]))
{ to find out if the cookie is set
echo "cookiename". $cookie_name. "is not set"; }
else{
echo "cookie". $cookie_name. "is set!
"<br>";
echo "value". $_COOKIE($cookie_name);
}?>
</body></html>
```

*) Delete a cookie in php:-

```
<?php
Setcookie("user","",time() 3600);
?>
<!DOCTYPE html>
<html><body>
<?php
echo "cookie user is deleted";
?>
</body></html>.
```

To delete a cookie use the setcookie() with an expiration date in the past.

→ Example program on working with Database:- (1) mysqli object oriented improved.

→ open a connection to mysql:- (mysqli object oriented)

1) Ex:- <?php
$servername = "localhost";
$username = "username";
$password = "password";
// create connection
$conn = new mysqli($servername, $username, $password);
// check connection
if ($conn->connect_error) {
die("connection failed:" . $conn->connect_error);
}
echo "connected successfully";
?>

2) ex:- (mysqli procedural):-
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// create connection
$conn = mysqli_connect($servername, $username, $password);
// check connection
if (!$conn) {
die("connection failed" . mysqli_connect_error());
}
echo "connected successfully";
?>

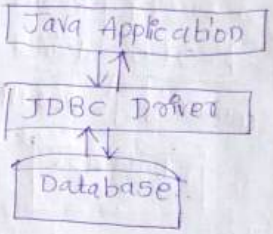
3) php data objects (PDO):-
<?php
$servername = "localhost";
$username = "username";
$password = "password";
try {
$conn = new PDO("mysql:host=$servername; dbname=myDB", $username, $password);

```
//set the PDO error mode to exception
$conn->setAttribute(PDO::ATTR_
ERRMODE, PDO::ERRMODE_EXCEPTION);
echo "connected successfully";
}
catch(PDOException $e){
 echo "Connection failed :". $e->getMes
                          age();
}
?>
```

→ myDB is a specified database.
→ PDO require a valid database to connect to if no database is specified an exception is thrown.

Unit-3 D: 6/11/2022

JDBC → Java Data Base Connectivity



JDBC is a JavaAPI to connect & execute the query with the database. It is a part of JavaSE (Java standard edition).
→ JDBCAPI uses JDBC drivers (s/w) to connect with the database.
→ It is a standard java-API for data-base independent connectivity b/w the java programming language
→ the JDBC driver includes :-
1) Making a conn to a database
2) Creating sql & mysql statements
3) Executing sql/mysql quarries in the database
4) Viewing & modifying the resulting records

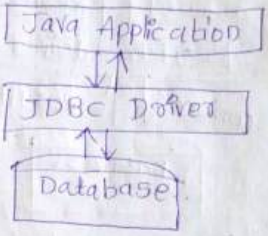
```
// set the PDO error mode to exception
$conn -> setAttribute (PDO:: ATTR_
ERRMODE, PDO :: ERRMODE_EXCEPTION);
echo "connected successfully";
}
catch(PDOException $e){
  echo "Connection failed :". $e -> getMes
                              age();
}
?>
```

→ myDB is a specified database.
→ PDO require a valid database to connect to if no database is specified an exception is thrown.

Unit-3

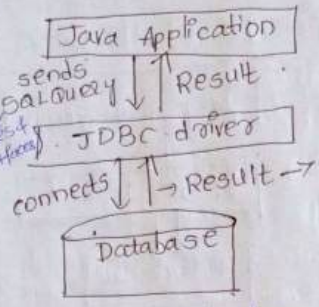
D: 6/11/2022

JDBC → Java Data Base Connectivity



JDBC is a JavaAPI to connect & execute the quesay with the database. It is a part of JavaSE (Java standard edition).
→ JDBCAPI uses JDBC drivers (s/w) to connect with the database.
→ It is a standard java-API for data-base independent connectivity b/w the java programming language
→ the JDBC driver includes :-
1) Making a conn to a database
2) Creating sql & mysql statements
3) Executing sql/mysql quarries in the database
4) viewing & modifying the resulting

* Working of JDBC :-



JDBC act as a interface b/w java application & database.

→ Java application invokes some set of classes & interfaces (SQL query) from JDBC driver inorder to send SQL querries to the database.

→ JDBC driver receive the SQL results from the database. So, finally the JDBC driver send the result to the java application.

* Architecture of JDBC :-

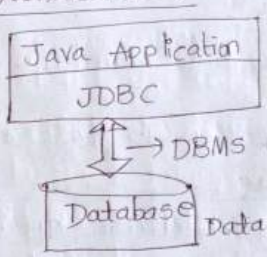
The JDBC supports both 2-tier architecture & 3-tier architecture.

* Java Application :- It is a java applet or a servlets that comm with a database

→ the JDBC API allows java progra-ms to execute sql statements & retrieve results

→ Driver manager (JDBC driver) :- plays an important role in the JDBC architecture. It uses some database specific drivers to effectively connect enterprise appli-cation to database.

2-tier Architecture :-



In the 2-tier model a java application directly comm with database.

for this purpose it requires a JDBC driver to comm with partic database being accessed

After that the user sends a query to the database.

the database process the query & then sends the response back to the user

In this model the database may not always present in a single machine. It can be located on a different machine on a network which a user is connected. This is known as client-server configuration. where user act as a client & the machine having the database.

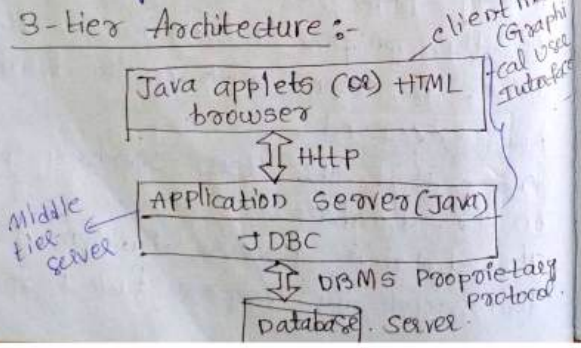
Advantages:-

1) It is ~~used~~ easy to maintain & modification also.
2) communication is faster.

Disadvantages:-

1) In this architecture application performance will be degrade based on increasing the user.

* 3-tier Architecture:-



In 3-tier model the user queries are send to the middle tier server.
→ from middle tier server the queries are again send to the database.
→ the database takes the queries from middle tier server & process these queries and then sends the response back to the middle tier server.
→ After that middle tier server sends the results back to the user.

Advantages:-

1) It provides high performance
2) It gives improves security

Dis:-

1) It is more complex to maintain.

Imp

* JDBC Drivers:-

The s/w ie created according to the JDBCAPI is called "JDBC Driver"
→ JDBC drivers act as an interface b/w java application & database.
→ It converts Java calls into database specific calls.

→ and database specific calls into java calls.

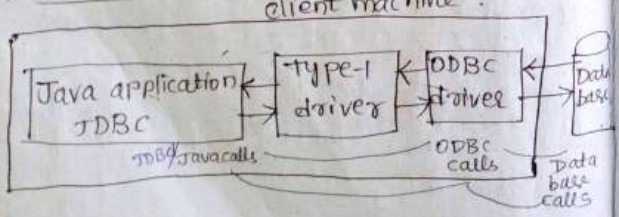
→ JDBC driver is a s/w component that enables java application to interact with the database.

Types of JDBC Drivers :-

There are 4 types of JDBC Drivers.

1) Type-1 (or) JDBC-ODBC bridge driver
2) Type-2 (or) Native-API driver (Partially Java driver)
3) Type-3 (or) Net protocol driver (or) middle ware driver (or) fully Java driver.
4) Type-4 (or) Native protocol (or) thin driver (or) pure Java driver.

1) Type-1 driver :-



Type-1 driver is also known JDBC-ODBC bridge driver.

→ It is developed by "sun micro systems" & supplied as a part of JDK (Java Development kit).

→ Internally, this driver takes the help of ODBC driver to communicate with the database.

→ It converts all JDBC calls into ODBC calls & send them to ODBC drivers

→ the ODBC driver converts all ODBC calls into database specific calls.

→ the type-1 driver act as bridge b/w JDBC & ODBC. Hence, the name come into picture.

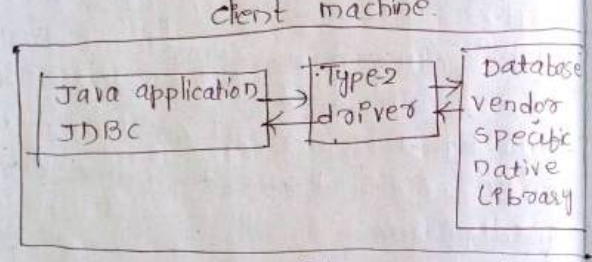
Advantages :-

1) Type-1 driver is very easy to use & maintain.
2) Using type-1 driver we can access any database.
3) It is available as part of JDK & Hence we are not require to install it seperately.

Dis :-

1) The performance is very less. Because 1st it converts JDBC calls into ODBC calls & ODBC driver converts ODBC calls into database specific calls.
2) It may not be suitable for large scale applications.

2) Type-2 drivers :-



It is known as Native API drivers.
→ Type-2 driver is similar to type-1 driver, except that ODBC driver is replaced with database vendor specific native library.
→ Native libraries are set of fns written in non-java

→ we have to install vendor provided native libraries on the client machine
→ this driver converts JDBC calls into vendor specific native library calls.
→ the native library calls can be understandable directly by the database.

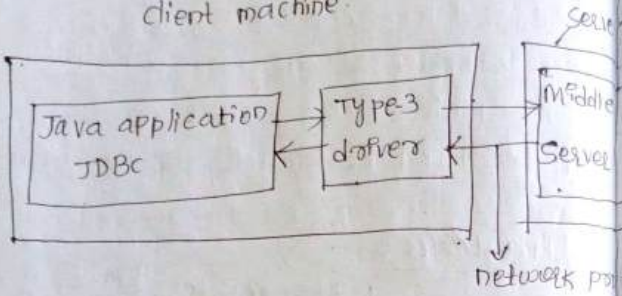
Advantages :-

1. It provides better performance than type-1 driver because it requires only 1 level conversion i.e JDBC to native library calls.
2. No need of ODBC calls

Dis :-

1) It is database dependent driver. Because it internally uses database native libraries
2) It is platform dependent driver.
3) We have to install native libraries on client machine.

3) Type-3 driver :-



→ It follows 3-tier architecture, where JDBC request are passed through the network to the middle tier server.

→ The middle tier server translates the request to the data ~~dat~~ base specific library & then sends it to the database.

→ the database server executes the request & gives back the result.

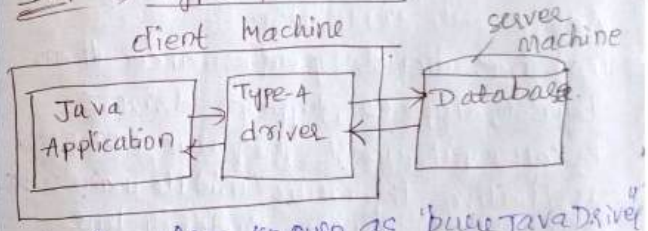
Adv :- 1) This driver server based, So need for vendor database library to present on the client machine.

Dis :-

1) Type-3 driver requires database specific coding to be done in the middle tier.

2) Maintainence of the middle tier becomes costly

Imp 4) Type-4 driver :-



It is also known as "pure Java Driver" or "thin driver".

→ It uses database specific native protocols to comm with database. (DB calls)

→ It converts JDBC calls into data - base specific calls directly. So, that client applications comm directly with the database server.

→ It is developed only in Java hence it also known as "pure Java driver".
→ It is a platform independent driver.
→ This driver won't require any ODBC driver / native libraries / middle ware server at client side & hence it is also called as thin driver.

Adv:-
1) It has better performance than type-1, type-2, type-3 drivers. Because it won't require ODBC driver/native libraries/middle ware server.
2) It is a platform independent driver.

Dis:-
1) It is database dept driver. Because it comm, directly with the database.
2) The user needs a different drivers for each database.

Eg:- thin driver for Oracle, Connector-J for mysql.

Imp
*features of JDBC :-

→ JDBC is an API using which we can communicate with any database without rewriting our java application.
→ Most of the JDBC drivers are implemented in java. Hence, JDBC is known as platform independent technology.
→ Using JDBC API, we can able to perform all basic database operations easily.
→ Using JDBC, you can request any type of queries from the database.
→ JDBC provides support for advanced datatypes, such as, BLOB & CLOB etc.
→ Using JDBC you can set save points for database & later you can rollback to desired savepoint.
→ You can use JDBC with different java applications. such as java applet

Java servlets, Java Server pages (JSP)

→ Using JDBC, you can send multiple updates to database & this is known as "Batch updating".

* JDBC steps:-

There are 5 steps to connect any java applications with the database using JDBC.

Step1:- Register the driver class

S-2:- Create connection

S-3:- Create statement

S-4:- Execute query

S-5:- close connection.

S1:- Register driver class:-

The "forName()" of class is used to register the driver class.

→ This method is used to dynamically load the driver class.

Syntax:-

Public static void forName(String className) throws ClassNotFoundException (or)

ex:- Class.forName("oracle.jdbc.driver.OracleDrivers")

S2:- Create Connection (getConnection())

→ the getconnection() of driver manager class is used to establish connection with the database.

syntax:-

public static Connection getConnection (String url) throws SQLException

(or)

public static connection getconnection(String url, String name, String password) throws SQLException.

S-3:- Create statement:- (createStatement()).

→ This method of connection interface is used to create statement.

⇒ The object of statement is responsible to execute queries with the database.

Syntax :- ~~create Sta~~

public Statement createStatement() throws SQLException.

Eg:- Statement stmt = con.createStatement().

S-4 :- Execute the query (executeQuery())

This method of statement is used to execute queries to the database.

→ This method returns the object of result set, that can be used to get all the records of a table.

Syntax :-

public ResultSet executeQuery(String sql) throws SQLException.

Eg:-

ResultSet rs = stmt.executeQuery("select * from emp"); → object.

S-5 :- Close connection :- (close())

⇒ The close() of connection interface is used to close the connection.

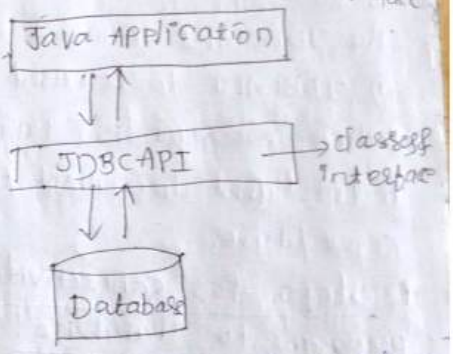
Syntax :-

public void close() throws SQLException.

Eg:- con.close()

19/12/2022

JDBC API :- Application programming interface



⇒ JDBC API stands for Java Database Connectivity standard Application programming Interface that allows Java programs to access DBMS.

→ The JDBC API consist of a set of classes & interfaces written in the Java programming language.

→ It provides Java API that allows Java programs to access DBMS.

→ The JDBC-API consist of two packages.

1) Java.sql 2) javax.sql.

The JDBC-API consist of two parts.

1) JDBC-API to be used by the application programmers (Java Application).

2) Low level to connect to a database server/driver.

* Steps for connectivity b/w java program to database.

1) Import the packages.
2) Load/register the driver, using forName().
3) Register the driver using driver manager.
4) Establish a connection using the conn class method
5) create a stmt (createStmt())
6) Execute a query
7) close the connection. (close())

⇒ JDBC have classes & Interfaces.

* Interfaces:-

6 Types.

1) connection Interface.
2) Statement Interface.
3) prepared Interface
4) callable statement Interface.
5) ResultSet Interface.
6) ResultSetMetaData Interface.

1) connection Interface:- getConnection()

Used to connect to the database

Syntax:-

Connection con = DriverManager.getConnection(url-uname)
(object) ↓ arguments.

2) Statement Interface:-

Used for create an statement(sql statement)

Syntax:-

Statement st = con.createStatement( ); (object) update/delete/insert

Statement contains two methods

1) Executeupdate() (Insert, delete & update)

2) Executequery() (select)

3) Prepared Interface :- This element is a child to Statement Interface.

Syntax :-

PreparedStatement PS = con.prepareStatement("I, into student values(" ", " ", " ");

↓ name, id, phone no etc

4) callable statement Interface :-

It contain stored procedure. (used to inorderd to call stored procedure i.e coll of multiple queries).

Syntax :

CallableStatement cs = con.prepare call ("{ call sp(" ", " ", " ")}")

5) Result Set Interface :-

It contains corresponding records.

→ All the records are stored into the result set.

Syntax :-

ResultSet rs = st.executeQuery("select * from ~~the~~ student");

6) ResultSetMetaData Interface :-

To know the infos about the table record.

Syntax :-

ResultSetMetaData rsmd = rs.getMetaData(" ");

classes :-

DriverManager :-

the drivermanger provides a basic service for managing JDBC driver.

→ The drivermanager class act as a an interface blw users & drivers.

→ the DriverManager class will attempt to load the drivers classes into the JDBC driver system.

* Packages :-

A java package is a directories that contains a group of similar types of classes, interfaces & sub-packages.

Uses :-

1. Reusability
→ Placing common code in common folder.
2. Maintanence
It provides a convinient way to organize your works.
3. Name Conflict :-
Resolves the conflict b/w two classes with same names.
4. Access protection :-
It can be used to provide visibility control.

* Types of packages :-

1) Built in / pre-defined package.
2) User defined package.

1) Java.sql :-

It contains core JDBC interfaces, that provide the basics for connecting to the database & interacting with data stored in the database.
→ Java.sql is the part of the J2SE (Java standard edition) is used to create applications for desktop environment.
→ J2SE it consist of all the basics of 'Java' language i.e vars, primitive datatypes, streams & strings.

→ The java.sql package contains the entire JDBC API that sends sql statements to relational data-bases & retrieve the results of executing those sql statements.

The conf mgmt category contains the classes & interfaces used to establish a conn with a database.

1) java.sql.connection :-

Create a conn with specific database you can use sql strmt to retrieve the desired results the content of a connection.

2) Java.sql.driver :- class

Creates & registering an instance of a driver with the Drivermanager interface.

3) Java.sql.Drivermanager :- class

Provides the fusility to manage database drivers.

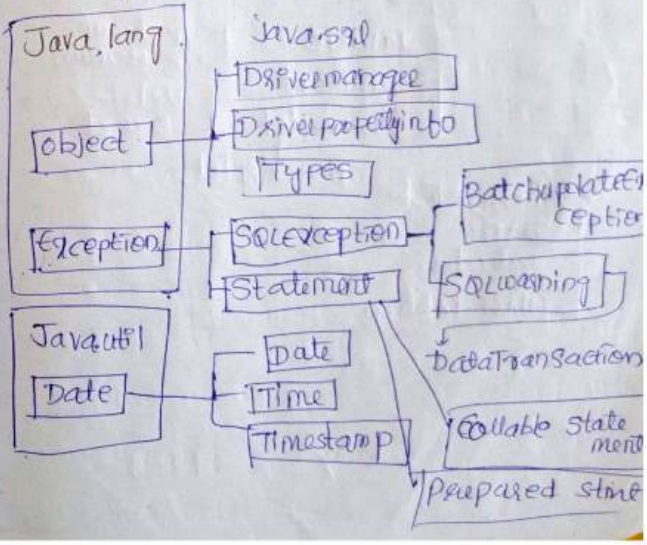
4) Java.sql.Driver property info :-

Retrieve the property required to obtain the connection.

5) Java.sql.SQL.Permission :-

SEtup logging stream with the Drivermanager.

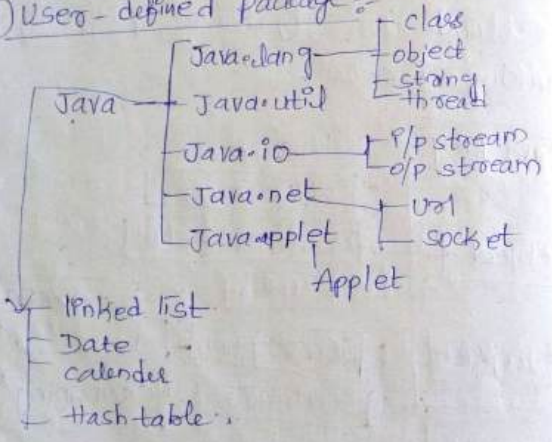
Public class Driver property info extend to obj driver property for making a connc.

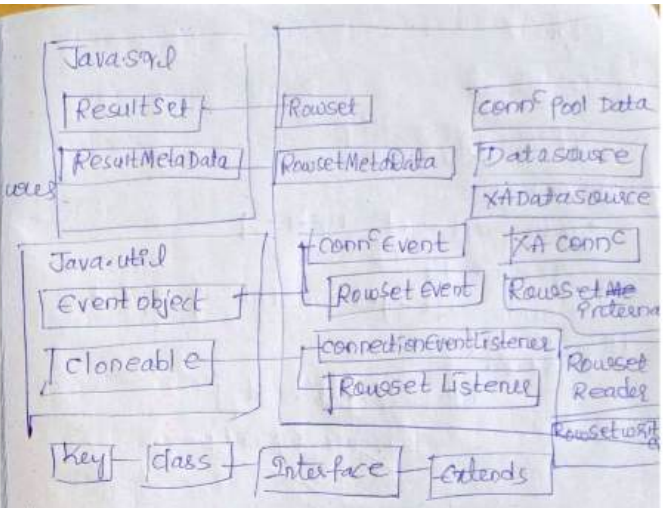


* javax.sql : It is the JDBC API that interface that interacts with Java naming & directory interface & manages connection pooling, among other advanced JDBC features. The javax.sql package contains the JDBC 3rd standard extension API.

→ The classes & interfaces in this package provide new functionality.

1) User-defined package :



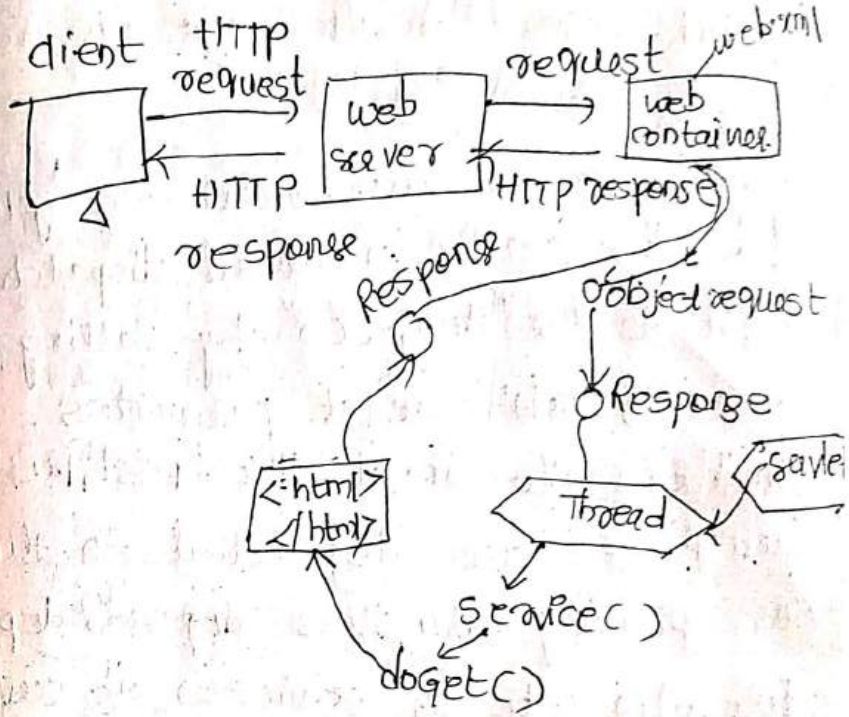


→ DriverPropertyInfo → public class DriverProperty-tyInfo extends object drives properties for making a connection.

WT-3

WT

Working of servlet:-



→ **webserver:-**

It can handle HTTP request send by clients & respond the request with an HTTP response.

→ **webcontainer / servlet container / servlet engine:-**

It is a part of webserver that interact with servlet.

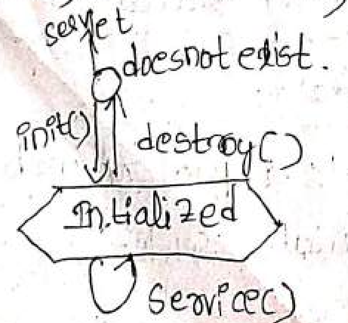
→ Service method: the service() dispat-ches the reqst to the correct handler method based on the type of requ-est.

for example, if server receive a get re-qst the service() would dispatch the reqst to the "doGet()" by calling the doGet() with reqst parameters, similarly the reqst like post(), put(), head() etc are dispatched to the corresponding handlers dopost() doput() dohead() etc by service() of servlet.

*Life Cycle of a servlet:-

there are three important methods in servlet life cycle.

1) init() 2) service() 3) destroy()



1) init():

This method is designed to be called only once in the life cycle of servlet. It is called when the servlet is first created.

→ It is used for one time initializations

→ the servlet is normal created when a user first invokes a 'URL' correspon-ding to the servlet.

Syntax:-

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

public → access modifier; void → return type; ServletException → Exception name.

2) Service():-

This method is the main method to perform the actual task.

→ the servlet container calls the service() to handle reqst coming from the client.

And to write the formatted response back to the client.

→ Each time the server recieve a reqst for a servlet. the server creates a new thread & calls the service().

→ the service() checks the HTTP reqst type (Get(), post()) & call doGet() or dopost().

Syntax:

```
public void service(servletRequest req,
                    servletResponse res)
{
  Actual business Logic for providing
          service to client
}
```

(Interface → req object)

3) destroy(): this method is called only once at the ~~at~~ end of the life cycle of servlet.

→ It gives our servlet a chance to close database conn & perform other clean up activities.

Syntax:

```
public void destroy(){
{
  Closing databaseconnection
}
```

Exploring Servlet API :-

The servlet are the java programs that run on the java enable web server/ Appl server.

→ they are used to handle the reqst obtained from the webserver, process the reqst, produce the response then send a response back to the web server.

→ API which contains all the neces -ary interfaces & classes.

→ servlet API has 2 packages.

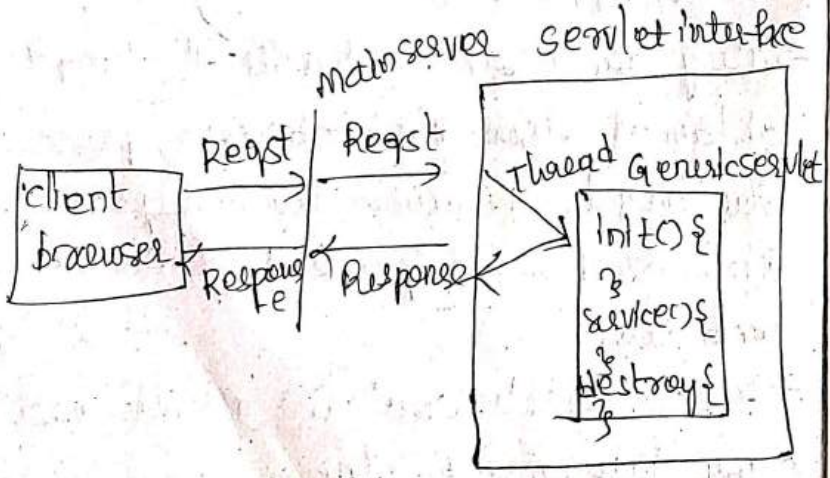
① Javax.servlet → Generic servlet.

② Javax.servlet.http → httpservlet.

① Javax.servlet :- this package provi-des the no.of interfaces & classes to support 'generic servlet' which is protocal Independent.

→ these interfaces & classes discribe & define the contract blw a servlet class & run time environment provi-ded by a servlet container.

*Generate Servlet class :-



→ servlet API provide generic servlet class in "javax.servlet" package.

→ An abstract class that implemen-ts most of the servlet basic methods.

→ Implements the servlet, servletConfig & serializable interfaces.

→ Protocol Independent servlet.

→ providing simple version of the 'life' cycle method.

Syntax :-

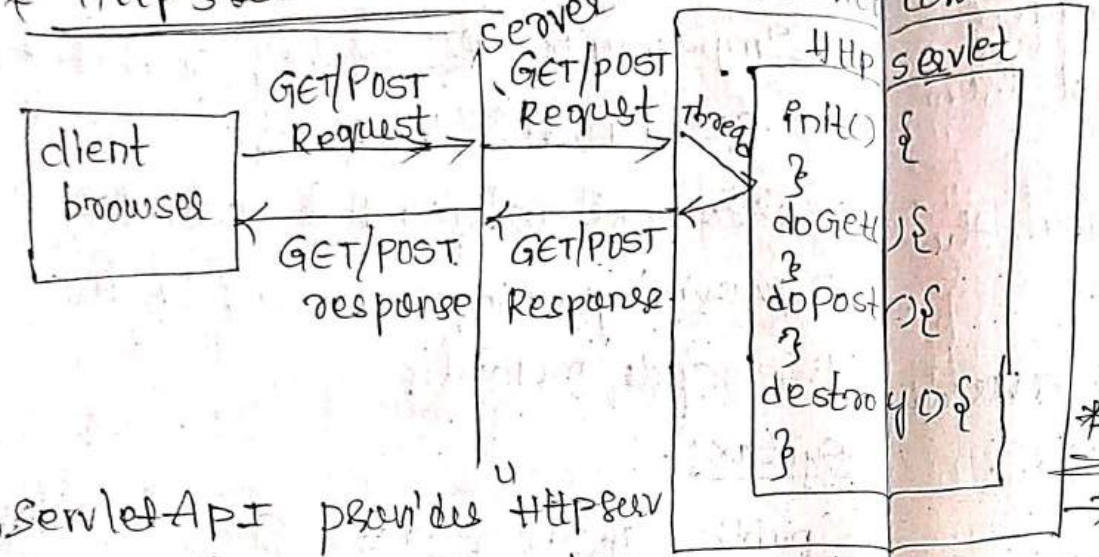
public abstract class GenericServlet
extends Java.lang.object
implements servlet, servletConfig, Java.io.Serializable.

② Javax.servlet.http :- this package provides the no.of interfaces & class-es to support "httpservlet" which is http protocal dependent.

→ these interfaces & classes describe & define the contract blw a servlet

class running under http protocol & the run time environment provided by a servlet container.

* Http servlet class :-



→ ServletAPI provides HttpServlet class" in javax.servlet.http package

→ An abstract class to be sub-class to create an http specific servlet. i.e suitable for web site/web application

→ extends generic servlet class & implements Serializable interface.

→ Http protocol dependent servlet.

→ To extend javax.servlet.http.servlet class and override atleast one of the below methods.

1) doGet() :- To support http get reqst by the servlet.

2) doPost() :- To support http post reqst by the servlet.

3) doPut() :- To support http put reqst by the servlet.

4) doDelete() :- To support http dilete reqst by the servlet.

*features of servlet :-

→ servlet is a technology that is used to create web apps.

→ servlet is an API that provides many interfaces & classes including documentation.

→ servlet is an interface that must be implemented for creating any servle

→ servlet is a class that extends the capabilities of the servers and resp -onse to the incoming reqst.

→ Servlet is a web component i.e

deployed on the server. to create dynamic web pages.

→ Servlets are secure, portable & easy to use replacement of 'CGI'

→ servlet is a dynamic loaded module that services reqst from a webserver.

→ servlets are executed within the JVM (Java virtual machine)

→ servlet is running on the server side it doesn't depend on browser.

* Creating a servlet :-

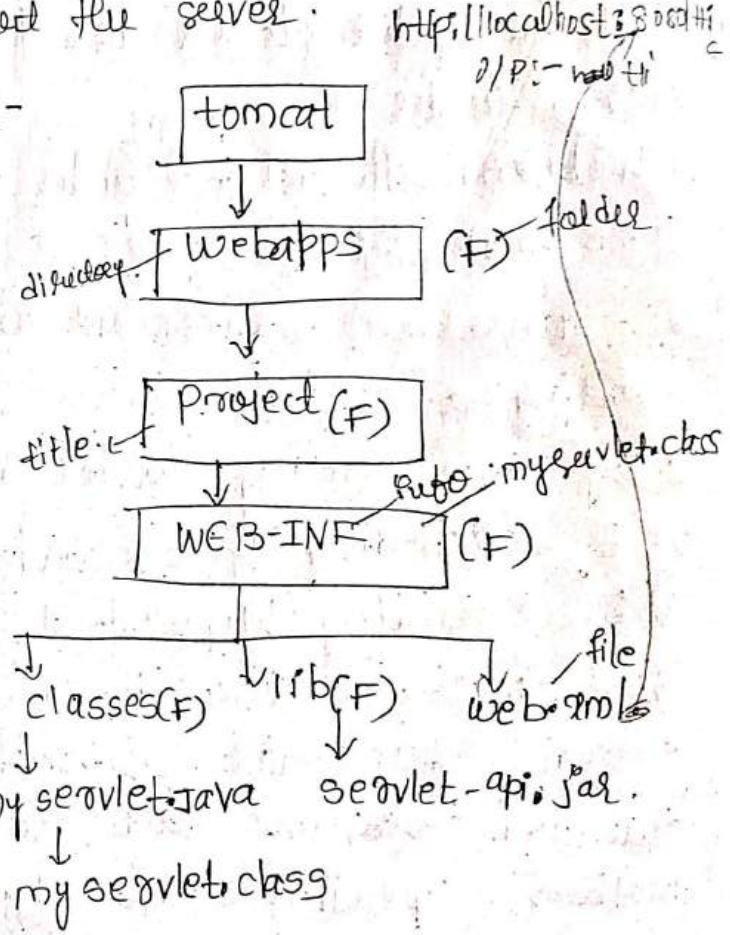
→ first download latest version of Tomcat server & install it on your machine

→ ApacheTomcat is an opensource webserver for testing servlets & JSP technology.

→ After including Tomcat server follow below steps.

1) create directory structure of your appln.
2) create a servlet.
3) compile the servlet.
4) create deployement discriptor (DD) for your appln.
5) Start the server.

1):-



"Sunmicro systems" defines a unique directory structure that must be follow to create a servlet appln.

→ Create the above directory structure inside apache-tomcat/webapps directory.

→ All the servlet classes are kept in classes folder.

→ web.xml file is kept under WEB-INF folder.

2) :- there are 3 different ways to creating a servlet

1) By implementing servlet interface
2) By extending GenericServlet class.
3) By extending HttpServlet class.

But mostly a servlet is created by extending http servlet abstract class. the web container calls the service method depending on the type of request the service() calls either the doGet() / doPost().

write code in a notepad file & save it as a myservlet.java & paste the class file in the WEB-INF/classes/ directory.

3:- To compile a servlet a java file is required. Different servlet requires diff javafiles.

→ In apache Tomcat server servlet_api.jar file is require to compile a servlet class.

steps to compile a servlet :-

1) C:\vaaq\tomcat\webapps\project\WEB-INF\classes\javac myservlet.java.

↓ errors

2) download servlet_api.jar file.

To open:

C:\vaaq\tomcat\webapps\project\WEB-INF\lib\servlet-api.jar.

no errors

To set path:-

~~Classe~~

→ c:\vaag\tomcat\webapps\project\
WEB_INF\classes\javac myservlet.
Java -classpath "c:\tomcat11\lib\
servlet-api.jar"

→ c:\vaag\webapps\project\WEB_INF\
classes\myservlet.java

↓
myservlet.class

4):- is an xml document i.e used by webcontainer to run servlets & jsp pages.

⇒DD is used for several imp purposes
1) mapping URL to servlet class
2) Initializing parameters
3) Defining error page.
4) security goals.
5) Declaring tag libraries.

ex:- web.xml :-

```
<web_app version="3.0">
<servlet>
  <servlet_name>hello</servlet_name>   ← Give a internal name to your servlet
  <servlet_class>myservlet</servlet_class>   ← Servlet class that you have created
</servlet>
<servlet_mapping>   ← this tag maps internal name to public url name
<servlet_name>hello</servlet_name>
<url_pattern>/hello</url_pattern>
</servlet_mapping>   ← url name this is what the user will see to go to the servlet.
</web_app>
```

5):- http://localhost:8080.
* Run the servlet
http://localhost:8080/hello.

U-4

JSP [Java Server Pages]

* Jsp stands for "Java Server pages"
→ It is a "server side technology".
→ 'Jsp' is the comb" of template text & JSP elements.
→ the template text can be scripting code such as html, WML (Web Mark up Language) XML, or a simple plain text.
→ Jsp elements can be action tags, custom tags (user-defined tags) or JSPTL elements.
→ Jsp elements are responsible for generating dynamic contents.

Ex:-
```
<html>
<body>
<%  ⇒ scriptlet tag
out.println("Jsp is better than servlet");
%>
</body>
</html>
```

* scriptlet Tag:- It is used to write valid java code in our Jsp page.
* out:- Pre-defined object/ Implicity object.
→ It is used for creating "webapp" (web app).
→ It is used to create dynamic web content.
→ In this Jsp tags are used to insert java code into html pages.
→ It is an advanced version of servlet technology.

→ It is a web based technology help us to create dynamic & platform independent web pages.
→ In this java code can be inserted in html/xhtml pages or both.
→ Jsp is converted into servlet by jsp container before processing the client request.

Jsp syntax:-

Syntax available in JSP are following:-

1) Declaration Tag:- It is used to declare variables.

Syntax:-

<%! Dec var %>

Ex:- <%! int var=10; %>

2) Java script tag:- It allows to add any no. of java code, variables & expressions.

Syntax:- <% java code %>

Ex:- JSP

3) Expression tag:- It evaluates & converts the expression to a string.

Syntax:- <%= expression %>

Ex:- <% num1 = num1+num2 %>

4) Java Comments:-

It contains the text that is added for info which has to be ignored.

Syntax:- <%// java comments %>

* Jsp process of Execution:-

Steps for execution of JSP are following:-

1) Create html page from where request will be send to server.

Ex:- try.html

2) To handle to reqst of user next ps to create .jsp file (new Jsp

3) create project floder structure

4) create html file (my.html)

5) create .war file

6) start Tomcat

7) Run applications.

* Advantages of JSP :-

→ It does not require advance knowledge of java

→ It is capable of handling excepti-ons

→ Easy to use & learn.

→ It can contain tags, which are easy to use & understand.

→ Implicit out,println objects are there which reduces the length of code.

→ It is suitable for both java & non-java programs.

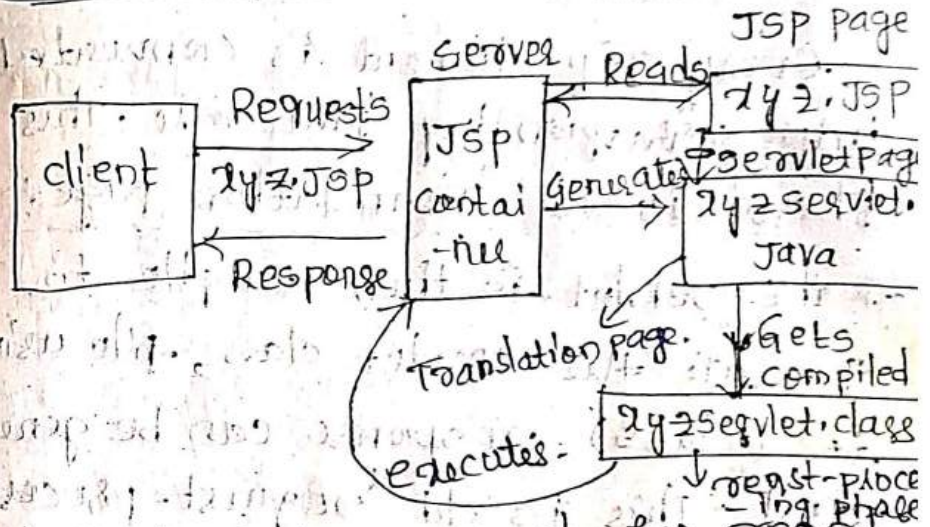
* Disadvantages :-

→ Difficult to debug for errors.

→ First time access leads to wait-age of time.

→ Its o/p is html which lacks feature

* Working of JSP :-



→ Clients makes a reqst for reqrd JSP page to the server.

→ On receiving the reqst the JSP container searches & then reads the desired JSP page.

→ Then the JSP page is converted to corresponding servlet.

→ while converting JSP page into servelet page every template text is converted into corresponding println page

Ex:- <html>
<% %>
</html>

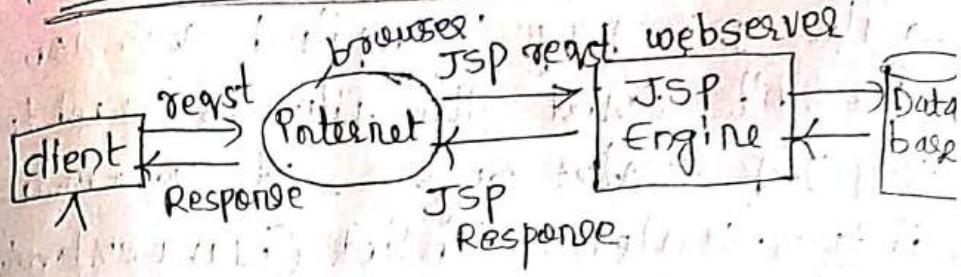
→ Every JSP element is converted into corresponding java code. this phase is cld "Translation page."

→ the servlet is then compile to generate the servlet class, file using the (.class) response can be generated. this is cld "request-processing phase".

→ the JSP container executes the servlet class file.

→ the requested page is then returned to the client as a response.

* Architecture of JSP :-



→ the JSP arch is a 3 tier-arch where each part has its own role & functities.

→ the JSP based on the web appln needs a JSP engine, which is a web container. → web containes can be defined as a webserver component.

→ the JSP container intercepts the JSP page reqst.

→ when a user navigates to a page ending with .JSP extention in their web server.

→ the web browser sends an http reqst to the web server.

→ the web server checks if a compiled version of the JSP page.

already exist.

→ the JSP page compiled version doesn't exist the file is send to the JSP servlet engine. It converts into servlet content (.java extension) this process known as "Translation phase".

→ then after translated .java file of the servlet is compiled into the .java servlet .class file. this process is known as "compilation phase".

→ the compiled .class file of the servlet is executed & the result is send back to the client machine.

* Life Cycle of the JSP :-

There are 5 phases in JSP lifecycle

1) Phase- Page Translation.

In this phase the JSP container in the web server translate the JSP page into servlet page.

2) Compilation :-

After translating the jsp page into servlet, the jsp container compiles the servlet page to .class file.

3) Loading & Initialization :-

After getting the .class file the container loads the .class file & creates a servlet object & initialize it by using init() method.

4) Request Handling :-

the servlet object handles the reqst by using service() method.

5) Destroy :-

After providing the service are response to the client then servlet obj is to be destroyed by using destroy() method.

* JSP Tags :-

There are 3 types of tags in JSP.

1) scriptlet tag :- It allows to write the valid java code in the JSP page.

Syntax :-

```
<%
    any valid java code
%>
```

Eg :-

```
<%
    out.println("valid java code");
%>
```

2) Declaring variables :-

Used to declare variables or methods.

Syntax :-

```
<%
    variable declaration
    method declaration
%>
```

Eg :-

```
<%
    String msg = "vaag";
%>
```

3) Expression tag :- Used to insert java values directly into the o/p.

Syntax :-

```
<%=
    expression code
%>
```

Eg :-

```
<%
    String msg = "vaag";
%>
<%= msg %>
```

* Directive elements in JSP :-

The directive elements are used to specify the attributes of a page.

→ there are 3 basic directives available in JSP.

→ there are

① page directive ② include directive

③ taglib directive

Syntax :-

<%@ directive attribute name = "value" %>.

① page directive :-

The page directive is used to provide information about the page

→ Using page directive we can import packages to our JSP page.

Attributes :-

1) Language :-

It specifies the language used by the JSP page.

2) Content Type :- This attribute specifies the MIME type.

3) Import :- It is used to import packages.

4) Errorpage :- It is used to report the unhandled excepti-

5) session :- It specifies the session data.

Ex :-

```
<%@ page language = "Java"
Content Type = "text/html" %>
<%@ page import = "java.util.*"
%>
<html>
<body>
<%
out.println("Todays Date is: "+new
        Date().toString());
%>
</body></html>.
```

② Include directive :- The include directive is used to provide the name of the file that can be included in the JSP page.

Ex :- <%@ include file = "one.html" %>

③ Taglib directive :- the taglib dire-ctive is used to specify the Custom tags.

explicit → user defined obj 18/1/23

* Implicit objects in JSP:-

These are pre-defined objects (created by compiler) (created by JSP compiler). That are accessable to all JSP pages.

→ JSP implicit objs are used to make the page dynamic.

→ the JSP container automatically initiate these objects while writing the script contain in the scriptlet expression tags.

Types of Implicit object:-

1) Request
2) Response
3) Application
4) session
5) config
6) exception
7) out

(1) Request:- Reqst objects are passed as parameters to the JSP service when a client reqst is made.

→ the JSP reqst is an implicit object of HttpServletReqst. i.e created for each JSP reqst by the web browser.

Syntax:-

```
request.getParameter("IP control name");
```

→ It can be used to get reqst values such as, parameter, Header info, remote address, server name, server port, content type, character encoding etc.

2) Response:- Is an implicit obj of 'HttpServletResponse'. the instance of HttpServletResponse is created by the web container for each JSP reqst.

→ the Response obj is used to carry the response of a client reqst of the service() is executed.

Syntax:- response.setContentType("text/html");
↓ method.

3) Application :- this obj provides ObjectTypeof Servlet Context.

→ this obj provides the resources shared within a web appln.

Syntax :-

application.getServletInfo();

This instance of servlet context is created only once by the web container when appln/project is deployed on the servlet.

→ This obj can be used to getInitialize parameter from configure file (web.xml).

4) session :- This varb is used to access the current client session.

Eg :-

session.setAttribute();
session.getAttribute();

→ session is an implicit obj of httpsession.

5) config :- It helps in passing the Info to servlet/JSP page during Initialization.

Syntax :-

config.getServletName();

→ config is an implicit obj of Servlet.config. This obj is used to getInitialize parameter for a particular JSP page.

→ the config obj is created by the web container for each JSP page.

6) Exception :- this obj is for handling error pages & contains Info about runtime errors.

→ Exception is an implicit obj of java.lang.Throwable class.

7) Out :- It is used to call the methods related to input output functions.

Eg :- out.println ("welcome to JSP");

checkout
Eqls :- Turn 3 pgs →

* Jsp action tags (elements) :-

There are many Jsp action tags elements. Each Jsp action tags is used to perform some specific tags → the action tags are used to control the flow b/w pages & to use Java bin. They are :-

1) jsp : forward :-

forwards the reqst & response to another resource. It is used to forward the reqst to another resource like (It may be) Jsp, html or another resource

syntax :-

without parameter :-

```
<jsp:forward page = "relative URL | <%= expression %>" />
```

(or)

with parameter :-

```
<jsp:forward page = "relative URL
<%= expression %>" />
```

```
<jsp:param name = "parametername"
value : parametervalue | <%= expression %>
(any sentence)
/>
</jsp:forward>
```

2) jsp: include :-

The jsp: include action tag is used to include the content of another resource. It may be Jsp, html or servlet.
→ It can be used to include, static as well as dynamic web pages.

syntax :-

with parameter :-

```
<jsp:include page = "relative URL | <%=
expression %>" />
<jsp:param name = "parametername"
value. parametervalue | <%= expression
%> />
</jsp:include>
```

without parameter :-

```
<jsp:include page = "relative URL | <%=
expression %>" />
```

3) Use Jsp:useBean action tag :-

It is used to locate / initiate a 'Bean' class. If bin obj of the bean class is already created it doesn't create the bean depending on the scope.

Syntax :-

Attributes:

<Jsp:useBean id = "InstanceName"
Scope = "page/request/session/application"
class = "packageName.className" type =
"packageName.className"
beanName = "packageName.className|<%=
expression %>" >
</Jsp:useBean>.

* Attributes :-

1) id :- It is used to identify the 'bean' in the specified 'scope'

2) scope :- represents the scope of the bean.

3) class :- Initiates the specified bean class.

4) type :- provides the bean a datatype, if the bean already exist in the scope

5) beanName :- Initiate the bean using java.beans.beans.instantiate() .

Eg :-

beanName = "packageName.className
|<%= expression %>" >.

4) Jsp:setProperty & getProperty Action tags :-

These are used for developing web appl^n with javabean.

→ the Jsp:setProperty Action tag sets a property value/values in a bean using the setter()

Syntax :-

<Jsp:setProperty name = "instanceof Bean" (or "bean") property = "*" | property = "propertyName" (or "username") />

→ the Jsp:getProperty Action tag returns the value of the property.

Syntax:-

<jsp:getPropertyname="object"
property="name" />
↓ username.

5) jsp:param (parameter) Action tag :- used to sets the par, value. It is used in forward & include tags.

6) jsp:plugin Action tag :- (Displaying applet in Jsp):-

This action tag is used to embedded applet in the jsp file.

→ This tag downloads plugin at client side to execute and an applet or bean.

* Applet :- It is a small dynamic prog" that can be transferd via the internet & run by a web browser.

Syntax:-

<jsp:plugin type="applet/bean" code = "name of class file"
code base = "directory Name of class File"
</jsp:plugin>.

Examples for implicit objects in Jsp:-

1) Request:- (index.html)

<form action="welcome">
<input type="text" name="uname">
<input type="submit" value> "go">
</form>

web.xml:-

<web_app>
<servlet>
<servlet_name> raag </servletname>
<jsp_file> welcome.JSP </jsp_file>
</web_app>

welcome.JSP:-

<%
Out.println("welcome" + request.getparameter
("uname"));
String driver = Config.getInitParameter("
dname");
Out.print("driver name is =" + driver);
%>.

2) Response:-

index.html

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br>
```

welcome.jsp:-

```
<%
response.sendRedirect("http://www.
                       google.com");
%>
```

4) session:-

index.html:-

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br>
</form>
```

welcome.jsp:-

```
<%
String name=request.getParameter("uname");
out.print("welcome"+name);
session.setAttribute("user",name);
<a href="second.jsp"> second JSP page
</a>
%>
```

second.JSP:-

```
<%
String name=(String)session.getAttribute
                       ("user");
out.print("Hello"+name);
%>
```

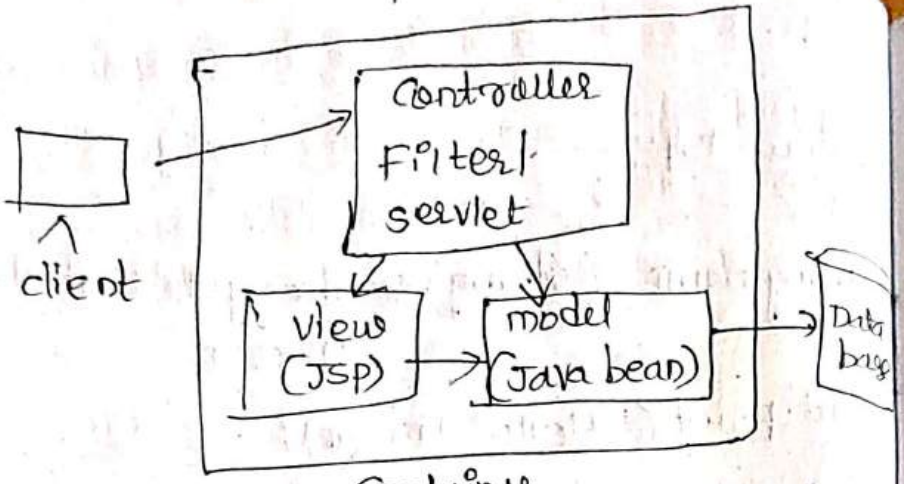
6) Exception:-

error.JSP

```
<%@ page is errorpage="true"%>
<html><body>
sorry following exception occured:<%=
exception:%>
</body></html>
```

* MVC in JSP:-

MVC stands for model view & controller
It is a design pattern that separates the bussiness logic, presentation logic & data.



Container.

Controller:- Acts as an interface blw view & model. Controller intercepts all the incoming requests.

model:- Represents the state of the application. i.e data. It can also have bussiness logic.

View:- Represents the presentation i.e UI (user Interface).

Advantages:-

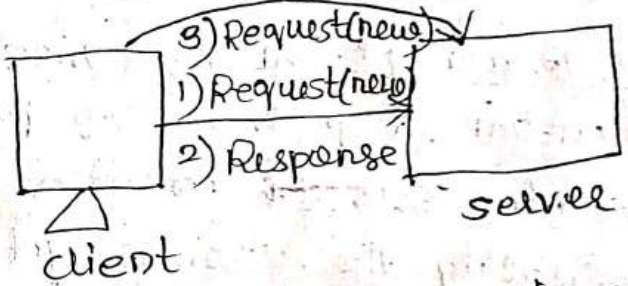
1) Navigation control is centralized.
2) Easy to maintain the large application.

Eg:- we are using servlet as a controller. Jsp as a view component, Java bean class as a model.

we have created 5 pages:-

1) index.jsp → a page that gets input from the user.
2) ControllerServlet.java → a servlet that acts as a controller.
3) login_success.jsp & login_error.jsp → files acts as view components.
4) web.xml – file for mapping the servlet.

* Session Tracking Machanism:-



Basically there are 2 types of protocols

1) statefull protocol 2) stateless protocol.

1):- In this protocol part of data is exchanged blw client & server. and these protocols always keep track of coming sessions.

Eg:- TCP (Transport control protocol)

2):- In this protocol part of data is exchanged blw client & server. And the protocol can't remember previously held comms.

Eg:- HTTP.

→ But sometimes, there is a need to keep track of previous comm session. In such situations we go for "session tracking".

*Session Tracking: It is a mechanism by which can keep track of previous sessions blw server & browser.

→ For creating the sessions, getsession() can be used.

→ Session tracking mechanism is a way to maintain state (data) of an user. It is known as "session mgmt in Servlet/JSP".

→ this method returns the object which stores the bindings with the names that use this object (getsession

→ these bindings can be managed using getAttribute() & setAttribute().

→ In session tracking two things are playing an imp role

1) HttpServletRequest Interface which support getsession()

2) HttpSession class which supports the binning, managing methods. such as, getAttribute() & setAttribute().

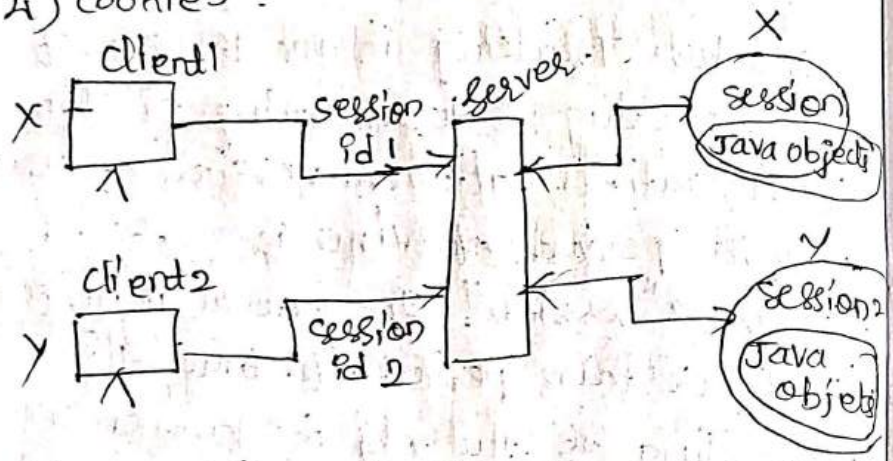
→ Session Tracking is the process of remembering & documenting customer conversation: the conversion of user over a period of time is referred to as a "session". In general it refers to a certain period of time. the recording of the obj is known as "tracking".

* Java servlet API for Session Tracking :- 4 types

1) User Authentication :-

~~the user will~~

1) User Authentication
2) Hidden fields
3) URL writing
4) cookies



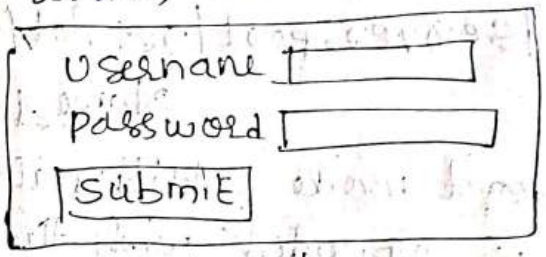
The servlet containes manages the session tracking task & user need not do it explicitly using java servlets.

→ Every client of the server will be mapped with a Javax.servlet.http.HttpSession object. Java servlets can be used the session obj to store & retrieve java objects across the session.

1) User Authentication :-

User will provide username & password to login to the application based on the user can be identified & the session can be maintained.

Ex:-



2) Hidden Fields | Hidden Form Fields :-

In case of hidden form field a hidden (invisible) text field is used for maintaing the state of an user.

→ We store the info in a hidden field & get it from another servlet.

Ex:- input type = "Hidden" name = "uname" value = "100";

Adv :- It will always work whether cookie is disable or not.

Dis :- It is maintained at server side.

→ Only textual info can be used.

3) URL writing :- In this we append a token/identifier to the URL of the next servlet/next resource.

Ex :- Http://server:port/servlet/ServletName

→ When a reqst made additional parameter is appended with the URL. In general added additional para-meter will be sessionid or sometimes userid.

Adv :- It will always work whether cookies is disirable or not (browser independent)

Dis :- It will work only with Links.

4) Cookies :-

Cookies are small programs which can make use of info submitted on currently accessed webpages.

→ A cookie is a key value paired created by the server & installed on the client's browser. When client makes a request for the first time.

→ Browsers also maintain a list of cookies installed in them & send them to the server as a part of sub-sequent http request.

→ The server can easily identify that these request is a part of a sequence of related request.

→ That is by using cookies provide an elegent solution to session tracking

Syntax :- cookie(String key, String value)

↓ Pairs values.

Adv :- the cookie is added by the addCookie() of the HttpServletResponse interfaces (add to browser side)

Adv :- 1) simple technique of maintaining the state

2) Cookies are maintained at client side.

Div :- 1) It will not work if cookie is desirable from the browser

2) only text value can be set in cookie object.

Online – elearning / distance learning

COVID 19: Long distance.

Tradition. class

Take notes.

WT-2

Working with Transactions:-

A transaction is a set of operation which works has an atomic unit.

(or)

It can be defined as a group of task.

A transaction only completes if all the operations completed successfully.

Method of Transactions:

1) SetAutoCommit(boolean status)
2) commit
3) roll back.

1):- It is used to set the autocommit mode to true/false.

Syntax:-

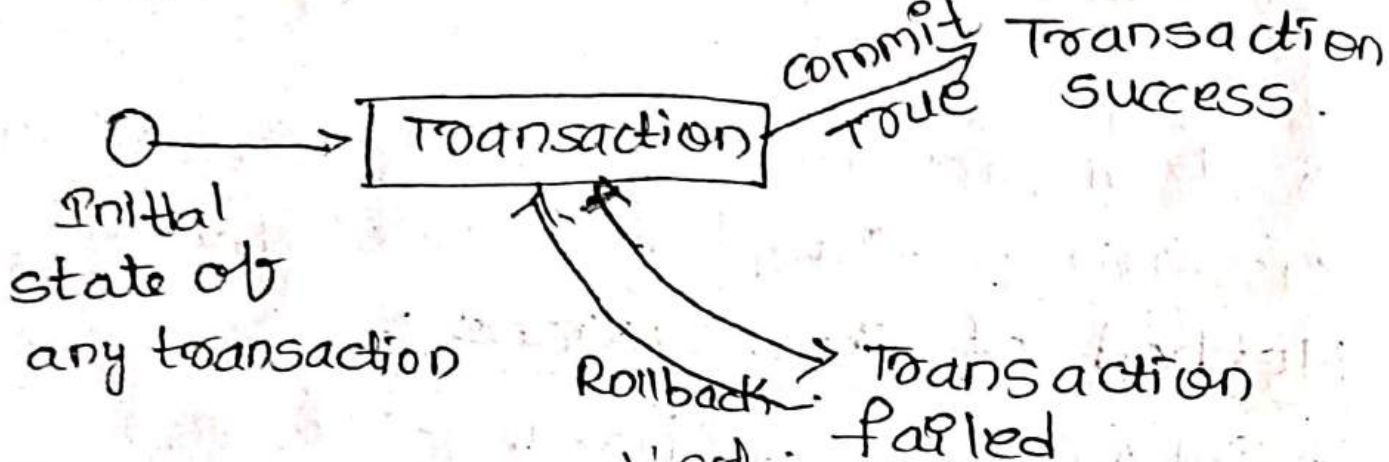
void setAutoCommit(boolean autocommit) throws SQLException

2):- It is used to commit all the changes made in the current transaction.

Syntax:- public void commit() throws SQLException.

3:- It is used to undo all the changes made in the current transactions.

Syntax:- public void rollback() throws SQLException.



Ex:-

```
Connection conn; -> object.
conn.setAutocommit(true/false)
insert {
-----  Error/Exception.
-----
}
conn commit;
conn.rollback;
```

Web Application :- / web App

It is a s/w, which can be access from the browser (applications)

→ In web Appln (web app) is a application program. i.e stored on a remote server & delievered over the internet through a browser interface.

→ A web Appln can be designed for a wide variety of uses & can be used by anyone from an orgn to an individual for reasons.

→ Commonly, used web appl's can include, webmail, online calculators google chrome, mozilla firefox, safari

→ the term web appln refers to a s/w system that provides a user interface through a web server.

→ web appl" can be simple, consisting of only ~~starting~~ static web pages & dynamic webpages.

google search

1) Static web pages :- (same content for all users)

these are stored in the filesystem of web browser usually display the same info to all visitors.

emails, HTML

2) Dynamic web pages :- (diff content for all users)

these are constructed by a program that produces the HTML.

this type of web Appl's provide individual info to the user.

* Web Application work :-

· web Appl do not need to be downloaded. Since, they are accessed through a network, users can access a web appl through a web server.

· such as :- Google chrome, Firefox.

→ Most web apps / web appl's are return in Javascript, html file, CSS ; client side programming, server side programming.

web app
I.
to

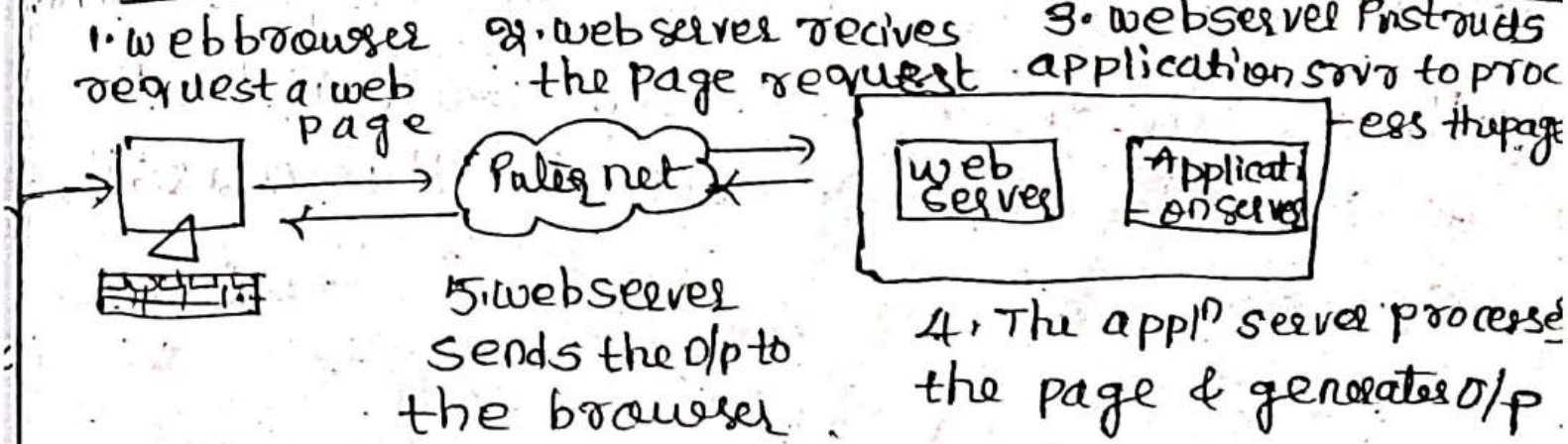
* Benifits of web applications :-

1) Allowing multiple users access to the same version of an applications.
2) web apps do not need to be install-ed.
3) web apps can be accessed through various platforms such as :- Desktop, laptops, mobiles.
4) It can be accessed through multiple browsers.

* Examples of web Applications :-

It include online forms, shopping cards, word processors, spread sheets, video & photo editing, file conversion, file scanning & email programs (email, yahoo!, gmail).

* The flow of the web Applications :-



1) In general, a user sends a request to the web server using web browsers such as, microsoft edge, Google chrome, fire fox. etc. over the internet.

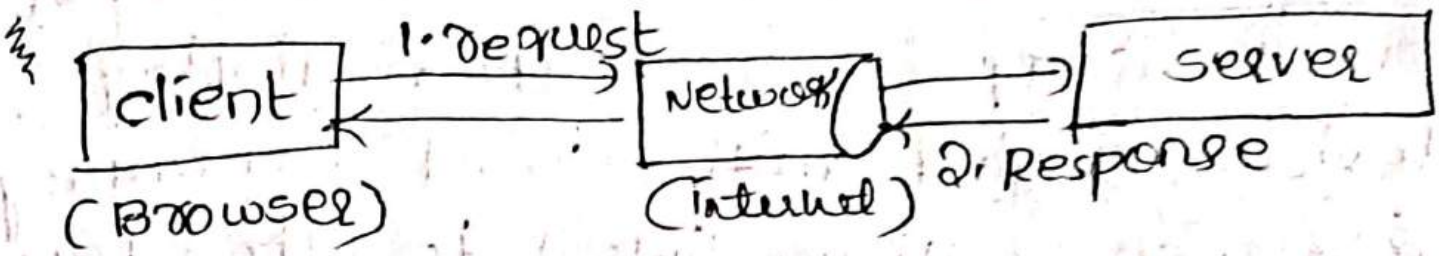
2) Then the request is forwarded to the appropriate web appln server by the web server.

3) web appln server performs the requested operations/ tasks like processing the database, querying the database, produces the results of the requested data.

4) the obtained result is send to the web server by the web appln server along with requested data / info / processed data.

5) the webserver respond to the user with the requested / processed data / info & provides the result to the users screen

Web Application Exploring the HTTP protocol:-



HTTP stands for Hyper text Transfer protocol.

the Hypertext Transfer protocol is the foundation for data comms on the web & It involves request/response interactions.

HTTP is an Application layer protocol used to deliver resources in distributed hypermedia info system.

In a web appl the request initiates activites that are implemented over the middle ware (network) & the response typically involves returning resources to the browser.

HTTP Resources :-

the resources delivered as part of this protocal. Typically include,

Hypertext, markup using the hypertext markup language.

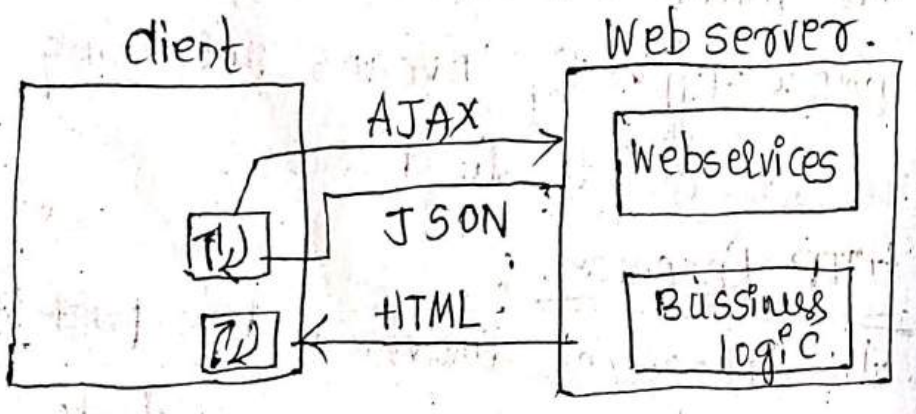
1) Hypertext :- Text can be displayed on a computer or other display device, styled with CSS & containing references (Hyper links) to other hypertext i.e reader is able to immediately access, usually via double click (mouse click).

2) Hyper Media :- the logical extension of hypertext to graphics, audio & video.

3) Hyper Links :- Define a structure over the web.

4) Scripts :- code that can be executed on the client side

*Web Architecture Models :-



the web appln architecture discribes the interaction between appln's, databases & middleware systems on the web. It ensures that multiple applns work A web appln architecture has to not only deals with efficiency but also with ~~relatively~~ reliability, scalability, security. Has soon as the user hits the go button after typing URL in the address bar of the web browser. It requests for that particular web address. the server sents to the brow -ser has a response to the request mode. the browser then executes these files to show the requested page. Finally the user is able to interact with the website.
the most important thing to note here is the code passed by the web browser. A web app works in a similar way.
Hence a web appln artr has to include

all the sub components as well as the external appl^n's interchanges for the entire s/w appl^n.

Working :-

There are 2 different codes (sub programs) running side by side.

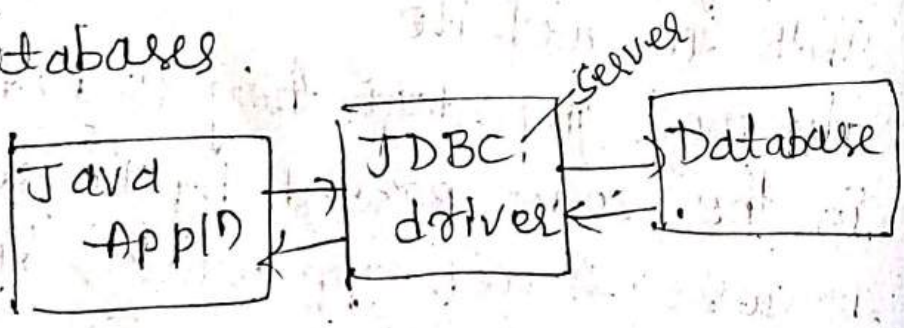
1) client-side code
2) server-side code.

1) :- the code that is the browser & response to that some user i/p.

2) :- the code that is the on the server & response to HTTP request

~~The MVC Architecture~~ :- Modles

1) One web server - one Database.
2) Multiple webservers - one Database
3) Multiple webservers, multiple Databases.



1):- It is the most simple as well as the least redable web app component model, such as a model uses a single server as well as a single database.

one web server, one database web appln component model is not typically used for real web appl^n's.

It is mostly used for running test projects as well as with the intend of learning.

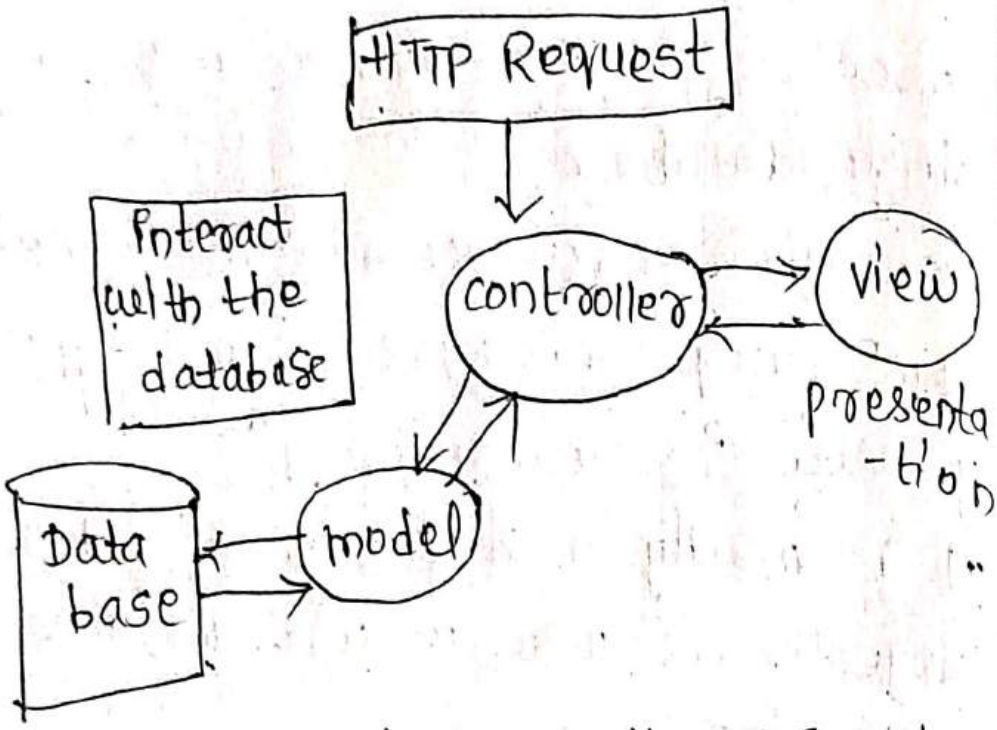
2):- Atleast two web servers are required for this web appln component model.

Even when one of the web server goes down the other one will take charge.

3):- It is the most efficient web appl^n. Componed model. Because neither the web server nor the ~~web~~ databases have a single point of failure. there are two options for this type of model. either to store identical data in all the employee databases or distribute evenly among them.

* The MVC Architecture :-

(MVC → model View Controller)



→ MVC is a design pattern created for developing appl^ns. specifically web appl^ns.

→ MVC works as controller model view approach.

→ MVC consist of three appl^ns. they are :-

1). UL Logic 2) Input Logic 3) Bussines Logic.

1) :- It deals with the View or front end of the appl^n.

2) :- It deals with the controller.

3) :- It deals with the model of an appl^n.

1) Model :- The model encloses the clean appln related data. But, the model does not deal with the any logic about how to present the data

2) View :- The 'view' element is used for presenting the data of the model to the user.

3) Controller :- The controller is inbet-ween the model & the view element.

* Benifits of MVC Architecture :-

1) Logical clustering of related acts on any controller can be achieved through MVC.

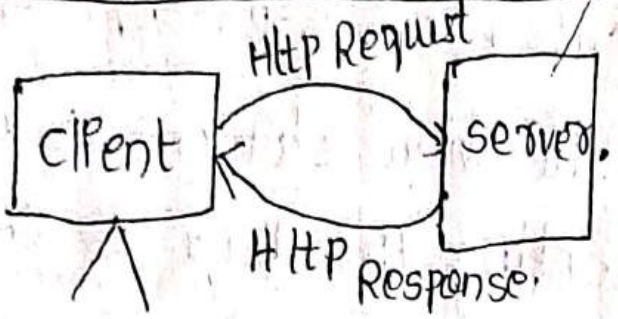
2) Various developers can work at the same time on different parts of the same appln.

* Introduction to servlet

Java program

doesnot accept all reqst so to accept the reqst we use servlet.

Http Request - Response :-



* HTTP is a state less request - response based comm protocol.
→ It is used to send & receive data on the webpage.
→ It uses a ~~ref~~ reliable TCP ~~Protoe~~ conn either for the transfer data to & from clients which was web browsers.

HTTP Request	HTTP Response
1) key element of reqst stream	1) key element of response stream.
2) HTTP method get/POST (action to be perform)	2) A status code (for whether the reqst was success - ful / not)

3) the page to access (URL)	3) content type (text, picture, HTML)
4) Form parameters	4) the content

* Need of servlets

1) suppose a client needs a dynamic web page. But the server does not handle the dynamic pages. It can handle the static web pages
2) So, inorder to satisfy the clt reqst for dynamic webpages we use servlets

* servlets:

servlets are simple java programs that run on the server. Hence the name servlets come into the picture.
→ servlets are most commonly used with HTTP. Hence sometimes, servelts called as "HTTP servlets."
→ servlets make use of the java standard extension classes in the packages javax.servlet & javax.servlet.HTTP.

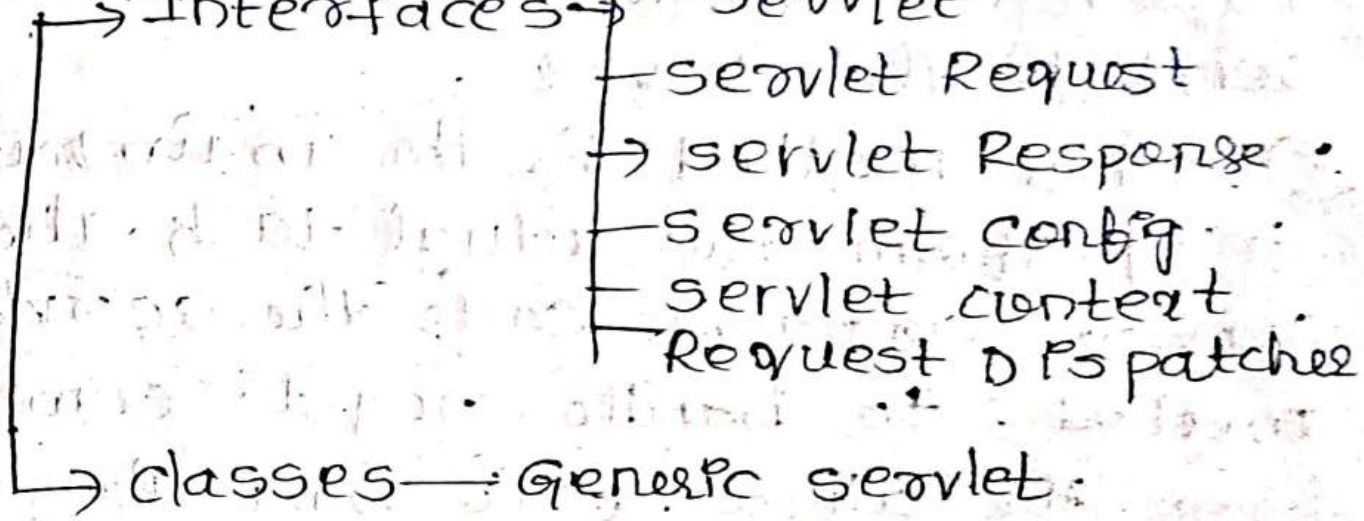
Uses of servlets:-

→ servlets can process & store the data submitted by an Html-form
→ servlets are useful for provi the dynamic contents.
Eg:- Retreiving & updating the data -ses.
→ servlets can be used in the coo & session tracking.
→ servlet API is a part of JEEAP servlet is an API for develop web base appl^ns.
→ the servlet is a server side technology. using which can deve -p appl^ns which run on a serve
→ Developing a servlet appl^n mean implementing the servlet interface either directly / indirectly.
→ A servlet progm can be develop in 3 ways.
1) A servlet progm can be devel -ed by implementing servelet inter

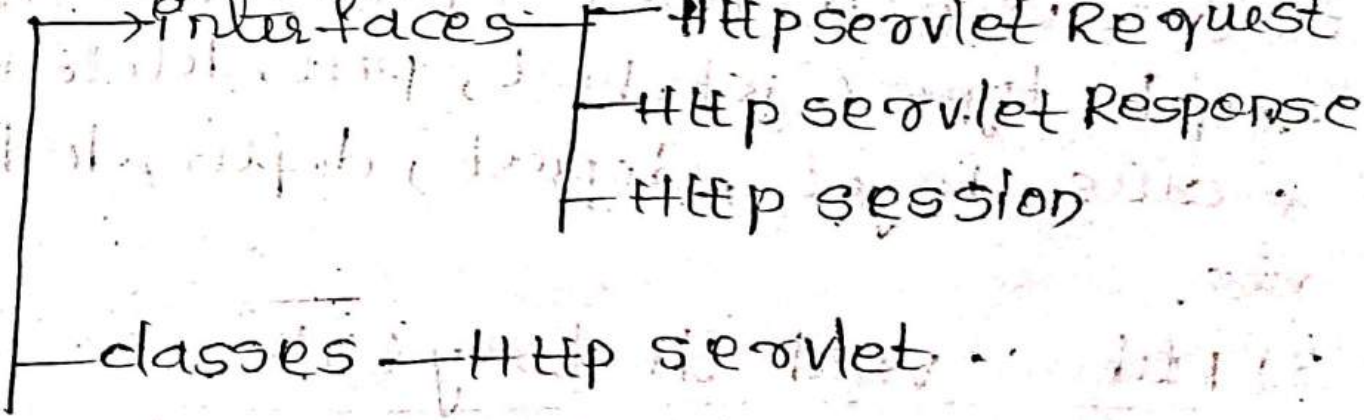
2) by extending generic servlet class
3) by extending Http servlet class.

→ The servlet API is provided to the programmer in the form of 2 packages.

1) Javax.servlet package :-



2) Javax.servlet.http package :-



*Methods of servlets :-

1) public void init(servletConfig).

2) public void service(servletRequest, servletResponse).

1) :- This method will be executed only 1 time. when the servlet object is created / when the 1st reqst is send to the servlet.

2) :- This method is the main method to perform the actual task. the servlet container calls the service method. to handle reqst coming from the client & to send the response back to the client. the service method checks the HTTP reqst type (get, post, put, delete etc) & calls doget, dopost, doput, dodelete etc.

3) Public void destroy():-

this method is executed only 1 time when the servlet is destroy-ed.

4) public servletConfig getServletConfig() :-

-This method returns an object of servletconfig.

5) public String getServletInfo() :-

this method returns the discription about the object.