UNIT-I

Introduction to PHP

PHP:- (Hypertext Preprocessor) is an open-source HTML-embedded server-side scripting language which is used to develop dynamic & interactive web applications and also used as a general purpose programming language.

PHP was developed by "Rasmus Lerdorf" in 1995, & later it has been developed as open source.

→ PHP page is a file with a .php extension that contains the combination of "HTML Tags & PHP scripts.

→ PHP is Server-side scripting language.

* Server side scripting means that the PHP code is processed on the web server rather than the client machine.
* PHP supports many databases (MySQL and PHP combination is widely used)
* PHP is an open source scripting language.

PHP Syntax:-

A PHP script starts with <?php and ends with ?>:

```
<?php
echo "Hello world!";
?>
```

Variable declaration in PHP:- Local, global, static

php automatically convert the variable to the correct data type depends on its value

eg:-

```
<html>
<body>
<?php
$x = 5;
$y = 10;
$z = $x+$y;
echo "Result is" $z;
?>
</body>
</html>
```

Rules for PHP Variables:

→ A Variable starts with the $ sign, followed by the name of the Variable.

→ A Variable name must start with a letter or the underscore character

→ A Variable name cannot start with a number.

→ A Variable name can only contain alphanumeric characters and underscores (A-Z, 0-9, and _)

→ Variable names are case-sensitive ($age & $AGE are 2 diff variables)

ex:-

```
<html>
<body>
<?php
$txt="www.com";
echo "I Like $txt";
?>
</body>
</html>
```

ex:

```
<html>
<body>
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
</body>
</html>
```

O/P:- 9

PHP Data Types:-

Variables can store data of different types,

1. String
2. Integer
3. Float - $x = 10.365; var_dump($x);
4. Boolean → $x = True, $y = false
5. Array
6. object → first we must declare a class of object.
7. NULL → $x = "Hello world"; $x = null; var_dump($x); O/P: NULL
8. Resource

?>

String functions:-

A string is a sequence of characters like "Hello world". @ CSE

```
<!DocType html>
<html>
<body>
<?php
$x = "Hello world!";
$y = "Hello world!";
echo $x;
echo "<br>";
echo $y;
?>
</body>
</html>
```

o/p:- Hello world!
Hello world!

Eg:- string functions:

```
<html>
<body>
<?php
echo strlen("Hello world");
echo strrev("Hello world");
echo str_word_count("Hello world");
echo strpos("Hello world", "world");
echo str_replace("world", "Dolly", "Hello world!");
?>
</body>
</html>
```

2. Integer:-

An Integer data type is a non-decimal number b/w -2,147,483,648 to 2,147,483,647

Eg:-
```
<?php
$x = 5985;
var_dump($x);
?>
```

o/p:- int(5985)

$x - Integer

var_dump() - It return the data type & value

3. Flow:-

Arrays:- Array is a special variable, which can hold more than one value at a time

In php, there are 3 types of arrays

1. Indexed arrays
2. Associative arrays.
3. Multidimensional arrays.

echo & print both are same
↓ ↓
has no return value. has a return value
↓ ↓
can take multiple parameters one argument
echo is faster than Print

Php Indexed Arrays:- There are two ways to create indexed arrays

1. The index can be assigned automatically.
   $cars = array("Volvo", "BMW", "Toyota");
2. The index can be assigned manually.
   $cars[0] = "Volvo";
   $cars[1] = "BMW";
   $cars[2] = "Toyota";

Eg:-
```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like". $cars[0]. ", ". $cars[1]. "and". $cars[2]. ", ";
?>
```

→ Length of an Array- count function.

Eg:-
```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

→ Loop through an Indexed array

Eg:-
```
<?php
$cars = arrays("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
FOR ($x = 0; $x < arrlength; $x++)
{ echo $cars[$x]; echo "<br>"; } ?>
```

php Associative Arrays :-

Associative arrays are arrays that use named keys that you assign to them

→ There are 2 ways to create an associative array.

1. `$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");` (or)
2. `$age['Peter'] = "35";`
   `$age["Ben"] = "37";`
   `$age['Joe'] = "43";`

Eg:-

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is ". $age['Peter']. " years old.";
?>
```

→ Loop through an Associative Array.

```
<?php
$age= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
$arrlength = count($age);
for($x=0; $x<$arrlength; $x++)
{
echo $age[$x];
 echo "<br>";
}
?>
```

diff b/w echo & Print

```
<html>
<body>
<?php
echo "<h2>PHP is fun!</h2>
echo "Hello world <br>";
?>
</body>
</html>
```

Print

```
<?php
Print "<h2>PHP is fun</h2>";
Print "Hello world <br>";
?>
```

Php Multidimensional Arrays:- It is an array in which each element can also be an array and each ē in the sub-array can be an array or further contain array within itself & so on.

```
<?php
$contacts = array(array("name" => "shilpa",
        "email" => "s@gmail.com"),
    array("name" => "N", "email" => "N@gmail.com");
echo "S' Email_id is: ". $contacts[0]["email"];
?>
```

O/P:-

Viewing Array structure & Values:- By using 2 statements

1. Var_dump() (or) Print_r()

Eg:-
```
<?php
$cities = array("London", "Hyd", Delhi");
Print_r ($cities);
?>
```

O/P

Array ([0] => London[1] => Hyd[2] => Delhi)

Eg 2:-
```
<?php
$cities = array("London", "Hyd", Delhi");
Var_dump ($cities);
?>
```

O/P:- array(3) { [0] => string(6) "London" [1] => string(5) "Hyd" [2] => string(8) "Delhi" }

Operators:- operators are used to perform operations on variables & values. PHP divides the operators in different ways.

1. Arithmetic operator
2. Assignment operator.
3. Comparison operator
4. Increment / Decrement operator
5. Logical operators.
6. String operators,
7. Array operators.

1: Arithmetic operators:- Arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

```
<html>
<head>
<title> Arithmetical operators </title>
</head>
<body>
<?php
$a = 42;
$b = 20;
$c = $a+$b;
echo "Addition operation Result : $c <br/>";
$c = $a-$b;
echo " substration operation Result : $c <br/>";
$c = $a*$b;
echo " Multiplication operation Result : $c <br/>";
$c = $a/$b;
echo "Division operation Result : $c <br/>";
$c = $a%$b;
echo "Modulus operation Result : $c <br/>";
$c = $a++;
echo "Increment operation Result : $c <br/>";
```

```
$c = $a--;
echo "Decrement operation Result:$c <br/>";
?>
</body>
</html>
```

2. Assignment operator:- It is used to numeric value to write a value to a variable. The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right

```
<html>
<head>
<title> Assignment operator</title>
</head>
<body>
<?php
$a=42;
$b=20;
$c=$a+$b;
echo "Addition operation Result : $c <br/>";
$c+=$a;
echo " Add AND Assignment operation Result : $c <br/>";
$c-=$a;
echo " subtract AND Assignment operation Result : $c <br/>";
$c*=$a;
echo " Multiply AND Assignment operation Result : $c <br/>";
$c/=$a;
echo " Division AND Assignment operation Result : $c <br/>";
$c%=$a;
echo "Modulus AND Assignment operation Result : $c <br/>";
?>
</body>
</html>
```

3. Comparison operator :- PHP comparison operators are used to compare two values (number or string).

```
<!DOCTYPE html>
<html>
<head>
<title> Php comparison operators </title>
</head>
<body>
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z);
echo "<br>";
var_dump($x === $z);
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x !== $z);
echo "<br>";
var_dump($x < $y);
echo "<br>";
var_dump($x > $y);
echo "<br>";
var_dump($x <= $y);
echo "<br>";
var_dump($x >= $y);
?>
</body>
</html>
```

identical

not equal

not identical

less than or equal

greater than or equal

4. Increment/Decrement operators:-

```
<html>
<head>
<title> Increment/Decrement operator </title>
</head>
<body>
<?php
$a=12;
$b= ++$a; pre-increment
echo "a =".$a." , b =".$b;
echo "<br>";
$b = $a++; - post-increment
echo "a =".$a." , b =".$b;
echo "<br>";
$b = --$a; - pre-decrement
echo "a =".$a." , b =".$b;
echo "<br>";
$b = $a--; - post-decrement
echo "a =".$a." , b =".$b;
echo "<br>";
?>
</body>
</html>
```

5. Logical operator:-

```
<html>
<head>
<title> Logical operator </title>
</head>
<body>
<?php
```

```
$a=true;
$b=false;
$c=true;
// AND example:-
var_dump($a and $b); -
var_dump($a and $c);
echo "<br>"; (or)
var_dump($a && $b); -T
var_dump($a && $c); -f
echo "<br>";
// or example
var_dump($a || $b); -T
var_dump($a || $c); -f
echo "<br>";
// Xor example
var_dump($a xor $b); -T
var_dump($a xor $c); -f
echo "<br>";
// NOT example
var_dump(!$a); - f
var_dump(!$b); -T
echo "<br>";
?>
</body>
</html>
```

6. String operators:

- concatenation - $txt1. $txt2
- = - assignment $txt1.= $txt2

(1)

```
<html>
<body>
<?php
$txt1 = "Hello";
$txt2 = "world!";
echo $txt1 . $txt2;
?>
</body>
</html>
```

O/P: Hello world

(2)

```
<?php
$txt1 = "Hello";
$txt2 = "world";
$txt1 .= $txt2;
echo $txt1;
?>
```

Array

+ union $x + $y -
== equality $x == $y
=== identify $x === $y
!= $x != $y
<> Inequality $x <> $y
!== Non-identity $x !== $y

There are 2 operation

1. . Concatenation - $txt1 . $txt2

2. .= Concatenation - $txt1 .= $txt2
assignment

1:- eg:
```
<html>
<body>
<?php
$txt1 = "Hello ";
$txt2 = "world!";
echo $txt1 . $txt2;
?>
</body>
</html>
```

O/P
Hello world!

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

O/P: Hello world

Control structures :-

① If statement : executes some code if one condition is true.

```
<?php
$t = date("H");
if ($t < "20")
{
echo "Have a good day!";
}
?>
```

② if-else-statement: It executes some code if a condition is true and another code if that condition is false

```
<?php
$t = date("H");
if ($t < "20")
{
echo "Have a good day!";
}
else
echo "Have a good night!";
}
?>
```

```
<?php
$a = 20;
$b = 30;
if ($a == $b)
{echo "a and b are equal";
}
else
{
echo "a and b are not equal";
}
?>
```

③ if-else if statement :- It executes different codes for more than 2 con

```
<?php
$t = date("H");
if ($t < "10")
{
echo "Have a good morning!";
}
elseif ($t < "20")
{
echo "Have a good day!";
}
else
{
echo "Have a good night!";
}
?>
```

Switch statement :- different actions based on different conditions

→ Select one of many blocks of code to be executed

```
<?php
$favcolor="red";
Switch($favcolor)
{
case "red";
echo "your favorite color is red!";
break;
case "blue";
echo "your favorite color is blue!";
break;
case "green";
echo "your favorite color is green!";
break;
default:
echo "your favorite color is neigther red blue, or green!";
}
?>
```

while loop.

```
<?php
$x=1;
while($x<=5)
{
echo "The number is:"$x<br>;
$x++;
}
?>
```

do-while loop.

```
<?php
$x=1;
do
{
echo "The number is:"$x<br>;
$x++;
}
while($x<=5);
?>
```

for Loop:-

```
<?php
for($x=0;$x<=10;$x++)
{
echo "The number is;$x<br>";
}
?>
```

Functions:- A function is a block of statements that can be used repeatedly in a program.

→ A function will not executed immediately when a page loads.

→ A function will be executed by a call to the function.

→ There are 2 types of functions.

1. User-defined functions. ② Built-in functions

P:-

Function without Arguments:

```
<?php
function writemsg()
{
echo "Hello world!";
}
writemsg();
?>
```

② with Arguments:

```
<?php
function familyname($fname, $year)
{
echo "$fname Born in $year<br>";
}
familyname("Sri", "1985");
familyname("Sai", "1988");
familyname("Shiva", "1989");
?>
```

Php Default Argument value:

```
<?php
function setHeight($minheight=50)
{
echo "The height is :$minheight<br>";
}
setHeight(350);
setHeight();
setHeight(130);
setHeight(25);
?>
```

Function Returning value

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

File Handling:- Php has several functions for creating, reading, uploading & editing files

1. **Readfile():** The readfile() reads a file and writes it to the o/p screen

① <?php
echo "This is my read operation";
?>
Save : sample.txt

② <?php
echo readfile("sample.txt");
?>
Save : simple.php

③ create (or) open file :- fopen() function is used to create a file and open a file

Eg:- <?php
$myfile = fopen("newfile.txt", "w");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>

④ closing a file
<?php
$myfile = fopen("sample.txt", "r");
fclose($myfile);
?>

⑤ append() :- <?php
$myfile = fopen("file1.txt", "a");
$file2 = "Good morning";
fappend($myfile, $file2);
fclose($myfile);
?>

⑥ Deleting a file :-
<?php
$myfile = "sample.txt";
unlink($myfile);
?>

→ check if the file is deleted or not
<?php
$myfile = "file.txt";
if(unlink($myfile)
{
echo "Delete the file". $myfile;
}
else
{echo "Not deleted ". $myfile;
} ?>

Textboxes and Radio buttons:

PHP form Handling: $_GET & $_POST are used to collect form data

```
<!DOCTYPE HTML>
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="Submit">
</form>
</body>
</html>
```

Name : [ ]
E-mail : [ ]
[Submit]

→ when the user fills out the form above & clicks the Submit button, the form data is sent with the HTTP post method

→ To display the submitted data you could simply echo all the Var

⇒ The "welcome.php" looks like this

```
<html>
<body>
welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["Email"]; ?>
</body>
</html>
```

O/P: welcome Shilpa
your e-mail add is shilpa@

⇒ The same result could also be achived using the HTTP Get method.

```
<html>
<body>
<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="Email"><br>
<input type="Submit">
</form>
</body>
</html>
```

⇒ and "welcome_get.php" looks like this

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?><br>
Your E-mail address is :<?php echo $_GET["email"]; ?>
</body>
</html>
```

GET	POST
1. Information sent from a form with the GET method is visible to everyone	1. Information sent from a form with the POST method is invisible to others
2. GET also has limits on the amount of characters to send the limitation is 2000 characters.	2 has no limits on the amount of information sent

Form Validation:- It contains various i/p fields, required and optional text fields, radio buttons, & a submit button.

eg:- Name: [ ]
E-Mail: [ ]
Website: [ ]
Comment: [ ]
Gender: Female Male Other
[Submit]

Name: Required + Must only contain letters & whitespace
E-mail: Required + Valid E-mail address (@)
Website: optional. It must contain a Valid URL
Comment: optional.. Multi-line i/p filed
Gender: Required must select one

Text fields;

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40">
</textarea>
```

Radio Buttons:-

```
Gender: <input type="radio" name="gender"
value="female">female
<input type="radio" name="gender" value="male">male
```

<input type="radio" name="gender" value="other">Other

Connecting to database (MySQL as reference)

MySQL is the most popular database system used with PHP.

What is MYSQL:.

→ MySQL is a database system used on the web
→ It runs on a server.
→ It is idea for both small & large applications.
→ Very fast, reliable, and easy to use.
→ It uses standard SQL. It compiles on a number of platforms.

MySQL database are stored in tables. A table is a collection of related data, and it consists of columns & rows.

MySQL db using.

1. MySQLi extension (the "i" stands for improved).
2. PDO (PHP data objects)

PDO	MYSQLi
1. It will work on 12 different db systems	1. It will work only on MYSQL db
2. PDO Makes the process easy. only have to change the connection string & afew queries	2. you will need to rewrite the entire-code.-queries includy
3. Object-oriented	3. object-oriented & a Procedural API.

Both support Prepared Statements. Prepared Statements protect from SQL injection, & are very important for web application security

Eg:- MYSQLi (object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
```

```
//create connection
$conn = new
mysqli($servername, $username, $password);
// check connection
if ($conn -> connect-error)
   {
die ("Connection failed : ".$conn->connect_error);
}
echo "connected successfully";
 ?>
```

close the connec

```
$conn->close();
```

MYSQLi Procedural

```
<?php
$servername="localhost";
$username ="username";
$password = "password";
$conn = Mysqli_connect($servername, $username, $password);
if(!$conn)
{
die("Connection failed: ". mysqli_connect_error());
}
echo " Connected successfully ";
?>
```

close the connection

```
mysqli_close($conn);
```

PDO:

```
<?php
$servername = "localhost";
$username = "username";
$password= "password";
try
{ $conn = new PDO ("mysql:host=$servername;dname=myDB",
                   $username, $password);
```

close the connection

```
$conn=null;
```

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

create db.

Create a db:- A db consists of one or more tables. create db statement is used to create db in MySQL

object-oriented

```
//create db
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql===TRUE)
  {
echo "Database created successfully";
  }
else
  {
echo "Error creating database: " .
      $conn->error;
  }
$conn->close();
?>
```

Procedural

```
//create db
$sql = "CREATE DATABASE MyDB";
if (mysqli_query($conn, $sql))
  {
echo "Database created successfully";
  }
else
  {
echo "Error creating database:" .
mysqli_error($conn);
  }
mysqli_close($conn);
?>
```

PDO

```
$sql = "CREATE DATABASE myDBPDO";
//use exec() because no results are returned
  $conn->exec($sql);
  echo "Database created successfully<br>";
    }
catch(PDOException $e)
  {
echo $sql . "<br>" .
  $e->getmessage();
  }
$conn=null;
?>
```

Cookie:- A cookie is used to identify a user

Create Cookie with php:-

Syntax:

```
Setcookie(name, value, expire, path, domain, secure, httponly);
```

Only the name parameter is required. All other parameters are optional

php create/Retrieve a cookie:

Modify is same op diff

```
<?php
$cookie_name = "User";
$cookie_value = "John Doe";
Setcookie($cookie_name, $cookie_value, time() +(86400 * 30), "/ ");  ∴ 86400=1 day
 ?>
<html>
 <body>
 <?php
 if(!isset ($_COOKIE[$cookie_name]))
   {
 echo 'Cookie named'". $cookie_name." 'is not set!";
   }
    else
    {
     echo "Cookie'". $cookie_name."' is set! <br>";
    echo "Value is: ". $_COOKIE [$cookie_name];
     }
    ?>
<P><strong>Note:</strong> you might have to reload the page to see the value of the cookie.</p>
   </body>
   </html>
```

Delete a cookie

```
<?php
setcookie("User", "", time()-3600);
 ?>
<html>
```

```
<?php
echo "Cookie 'User' is deleted.";
 ?>
<body>
<html>
```

Check if cookies are Enabled

```
<?php
setcookie("test_cookie", "test", time()+360, '/');
?>
<html>
<body>
<?php
if(count($_COOKIE) > 0)
    {
  echo "Cookies are enabled.";
    }
  else {
    echo "Cookies are disabled.";
      }
      ?>
      </body>
      </html>
```

Sessions. A session is a way to store information (in variable) to be used across multiple page. Unlike a cookie, the information is not stored on the user computer.

Start PHP Session

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variable
$_SESSION["favcolor"] = "green";
$_SESSION["favanima"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

PHP session variable

echo "favorite color is " . $_SESSION["favcolor"];

print_r($_SESSION);

Modify

```
$_SESSION["favcolor"] = "Yellow";
print_r($_SESSION);
```

Destroy

```
session_unset();
session_destroy();
echo "All session variables are now removed & the session is destroyed";
```

Expressions:- An expression is a combination of values, variables, operators, and functions that results in a value.

Ex:- $val = 2+3*4;

Precedence order	operator	Associativity
1.	() [] ->	Left to right
2.	++ -- (unary) ! ~ * and size of	Right to left
3.	* / %	Left to Right
4.	+ -	Left to Right
5.	<< >>	Left to Right
6.	< <= > >=	Left to Right
7.	== !=	Left to Right
8.	& (bitwise AND)	Left to right.
9.	^ (bitwise XOR)	Left to right
10	| (bitwise OR)	Left to right

Strings:- String is a collection of characters.

→ In php the String is denoted within a double quote

→ Concatenation is the only one operator used in String it is denoted by dot.

→ Strings are treated as the array of characters. The first position of the character is indexed as '0'.

String operators:- There are 2 types

1. The concatenation operator:- Using the . operator we can concatenate 2 strings.

```
<?php
$str1 = "Good";
$str2 = "Morning";
echo $str1 . $str2;
?>
```

o/p:- goodMorning

2. Concatenation Assignment:-

Using .= operator we can use concatenation assignment. In this operator, the concatenated string is assigned to a variable as a value.

```
<?php
$str1 = "Hello";
$str2 = "world";
$str1 .= $str2;
echo $str1;
?>
```

Handling file uploads:- PHP help in uploading a file. for uploading a file in your php configuration file php.ini search for file_uploads directive and set it on. i.e file_uploads = on

php $_FILES : The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp, file name and errors associated with file.

1. $_FILES['filename']['name']:- It returns file name
2. $_FILES['filename']['type']: It returns MIME type of the file
3. $_FILES['filename']['size']: It returns the size of the file (in bytes)

4. $_FILES['filename']['tmp_name']:- It returns temporary file name of the file which was stored on the server

5. $_FILES['filename']['error']:- It return error code associated with this file.

uploadform.html

```
<!DocType html>
<html>
 <body>
<form action="upload.php" method="post" enctype="multipart/
                                        form-data">
 Select file:
<input type="file" name="fileToUpload"/>
<input type="Submit" value="upload Image" name="submit"/>
  </form>
  <body>
  </html>
```

-> The form method must be post and form also needs the attribute: enctype="multipart/form-data" which specifies which content-type to use when submitting the form.

uploader.php:-

```
<?php
$target_path = "c:/";
$target_path = $target_path. basename($_FILES['fileUpload']['name']);
if(move_uploaded_file($_FILES['fileUpload']['tmp_name'], $target_path))
  {
  echo "file uploaded successfully!";
}
else
  {
  echo "Sorry, file not uploaded, please try again!";
} ?>
```

UNIT-II

XML

→ that defines set of rules for encoding docu

→ XML stands for extensible Markup Language. much like HTML

→ XML was designed to store & transport data

→ XML was designed to describe data

→ XML tags are not predefined in XML. you must define "your own tags".

→ XML is self describing

XML prolog

<? XML version = "1.0" encoding = "UTF-8" ?>

→ XML was designed to both human & machine readable.

→ XML uses a DTD (document type definition) to formally describe the data

SGML

XML	HTML
1. XML is not a replacement for HTML	1. HTML & XML were designed with diff goals.
2. XML was designed to describe data & to focus on what data is	2. HTML was designed to display data & to focus on how data looks
3. XML is about describing information	3. HTML is about displaying information
4. XML allows own tags & own document structure	4. Pre-defined tags & Predefined document structure

XML Tags :-

```
<?XML version="1.0"?>
<note>  — root
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>  } end of root
```

<root>
<child>
<subchild>...
</child>

O/P:
<sub>

child 1's <child2>

Note
To: Tove
From: Jani
Reminder
Don't forget me this weekend.

→ XML was designed to carry data - with focus on what data is.
→ XML simplifies data sharing.
→ It simplifies data transport
⊕ XML tags are "Case-sensitive"
② XML tags must be closed in an appropriate order.
→ In XML opening & closing tags must be written with the same case like: <message>This is wrong </Message>
<message>This is correct</message>

Adv of XML:
→ Truly portable Data
→ Easily readable by human user.
→ Very expressive, flexible & customizable.
→ Many additional standards & tools
→ Widely used & supported

Subject.xml
<?xml ve
<CSE>
<unit>PHP<
</CSE>

XML attribute & value:

Attributes are part of XML tags/ē's. An ē's can have multiple unique attributes. Attributes give more information about XML ē tags

Syn: <element_name attribute="" attribute="">
content... </element_names>

Attribute Type:
1. String Type:
2. Tokenized Type.
3. Enumerated Type

eg college.XML
<college>
<student>
<name>shilpa</name>
<rollno>-
</student>
<faculty>
<name>
<branch>
</fa.
</HOD
</college>

Attribute Rule:
→ An attribute name must not appear more than once in the same start-tag or empty element tag

→ An Attribute must be declared in the DTD using an Attribute-List Declaration

→ Attribute value must not contain direct or indirect entity reference to external entities.

⇒ XML coment starts with <!--- & ends with --->

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<!--- Students grades are uploaded by months --->
<class_List>
<Student>
    <name> Revansh </name>
    <grades> A </grades>
</Student>
</class_List>
```

XML Comments Rules:

1. Comments cannot appear before XML declaration
2. Comments may appear anywhere in a document
3. comments must not appear within attribute value
4. Comments cannot be nested inside the other comments

XML encoding. It is the process of converting unicode characters into their equivalent binary representation. When the XML processor reads an XML document, it encodes the document depending on the type of encoding.

encoding Types: UTF-8 — UCS Transformation format
UTF-16 ↓ Universal character set.

The number 8 or 16 refers to the number of bits used to represent a character.

→ They are either 8 (one byte) or 16 (two bytes)

→ for the documents without encoding i/f, UTF-8 is set by default

DTD (Document type definition) :- A DTD is to define the structure of the legal element & attribute of an xml document

Why use a DTD:-

→ With a DTD independent groups of people can agree on a standard DTD for interchanging data.

→ An application can use a DTD to verify that xml data is valid

Internal DTD declaration:

```
<?xml version = "1.0"?>
<!DOCTYPE note [                          root of e of their document is note
<!ELEMENT note (to, from, heading, body)>   e note defining that the note e must contain 4 e's "To, from, heading, body"
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to> Jani </to>
<from> Tony </from>
<heading> Remainder </heading>
<body> Don't forget my textbook </body>
</note>
```

External DTD declaration:

```
<?xml version = "1.0" encoding = "UTF-8">
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to> TOVE. </to>
<from> Jani </from>
<heading> Remainder </heading>
<body> Don't forget my textbook </body>
</note>
```

note.dtd

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

PCDATA: PCDATA means Parsed character data.

→ Think of character data as the text found btw the start tag & the end of tag of an xml e.

→ PCDATA is text that will be parsed by a parser. The text will be remined by the parser for entities & markup.

→ Tag inside the text will be treated as markup & entities will be expanded.

→ However, Parsed character data should not contain any &, < or > characters these needs to be represented by the &amp, &lt, &gt, entities respectively

CDATA: means character data.

→ is text that will not be parsed by a parsed, Tags inside the text will not be treated as markup & entities will not be expanded

DTD: Document Type definition [elements – #PCDATA
attributes – #CDATA

→ It is used to define the structure of a XML documents & the legal e's & attributes.

→ DTD is text based document with dtd extension

→ DTD contains element Declarations
attribute "
entity reference "

DTD are 2 types: 1. Internal (we can create in same file) eg: XML & DTD
2. External (we can create diff files 1. XML 2. DTD

Syntax: <!DOCTYPE root-element-name [ ] >
<!ELEMENT root-element-name (childs) >
<!ELEMENT child-element-name (#PCDATA) >
↓
we can store the data

1 :- Internal Save Internal : xml student.

```
<? xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [   (root)
<! ELEMENT student (name, rollno, branch)>   : child e's
<! ELEMENT name (#PCDATA)>
<! ELEMENT rollno (#PCDATA)>
<! ELEMENT branch (#PCDATA)>
]>
<student>
<name> shilpa </name>
<rollno> 123 </rollno>
<branch> CSE </branch>
</student>
```

structure

xml code

O/P :-
```
<student>
<name> -- </name>
<rollno>
<branch>
</student>
```

2 External :- 2 types [Private – SYSTEM, Public – PUBLIC]

Here DTD is 1 file & xml is 1 file

↓

works with multiple files

emp.dtd Private

```
<!DOCTYPE employee [
<! ELEMENT employee (empNO, name, Salary)>
<! ELEMENT empno (#PCDATA)>
"  name (#  )>
"  Salary (#  )>
```

Public

```
<!DOCTYPE employer PUBLIC
"-//ABHI//1.0//EN" "emp.dtd"
```
vendor name, ver, Lang

Employee.xml

```
<? xml version="1.0" encoding="UTF-8"?>
<! DOCTYPE employee SYSTEM "emp.dtd">
<employee>
<empno>
<name>
<salary>
</employee>
```

we follow the structure exactly so we are using xml validate & buddy

you are not allowed to use special characters like &, <, >, " if you want these characters use &amp, &lt, &gt, " respectively

```
<sub> c&s </sub>
<sub> c&amp;s </sub>
```

Types of e's

1. Text-only e's (It allows only text data)
2. child-only " (" " " child e's)
3. Mixed e's (It allows text + child e's)
4. Empty e's (not allows any data + child e's) → we can apply attributes
5. Any e's (text data + childs + Mixed + empty)

we store using External DTD → child only e's

save: Types.DTD.

root ↑ child ↑

③
```
<!ELEMENT studentinfo (student*)>
<!ELEMENT student (#PCDATA | name | rollno)*> → Mixed e's.
```

Text only e's
```
<!ELEMENT name (#PCDATA)>
    "    rollno ( " )
```

* - cardinality operator
↓
we can store many ops

Stude.XML

```
<!DOCTYPE studentinfo SYSTEM "Types.DTD">
<studentinfo>
    <student>
        The student name is <name> shilpa </name> and his
        rol no is <rollno> 123 </rollno>
    </student>
</studentinfo>
```

⑤ Any e's

④ empty e's :-

→ we can't store child & any type of data
→ we can apply the attributes

Eg: (Empr.XML)
```
<!DOCTYPE employees [
<!ELEMENT employees (employee*)>
<!ELEMENT employee EMPTY>
]>
<employees
    <employee></employee>
</employees>
```
no spaces

XML Schema:- It is commonly known as XML Schema Definition (XSD)

→ It is used to describe the structure of an XML document, just like a DTD.

→ XML Schema defines the elements, attributes & datatypes

→ A xml document with correct syntax is called "well formed".

eg:-

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org">
<xs:element name="contact">
 <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:"  "  "  ="company"
      <xs:"  "  "  ="phone"
    </xs sequence>
    </xs complexType>
    </xs:element>
    </xs:schema
```

→ <xs:element name="contact"> defines the e called 'contact'

→ <xs:complexType>:- The "contact" e is a complexType

<xs:sequence> the is a sequence of e

XML Schema Data Types :- 1. simple Type 2) Complex Type

1:- The simple type allows you to have text based e's. It contains neither attributes, child e's & can't be left empty

Syn: <xs:element name="phone_numbers" type="xs:int" />

Syn: <xs element name="phone_num" type="xs:int" />

2:- The complex type allows you to hold multiple attributes & e's. It can contain additional sub e's & can be left empty

DTD

1. DTD stands for document object Model
2. DTD are derived from SGML Syntax
3. DTD doesn't support datatypes
4. DTD doesn't support namespaces
5. DTD doesn't define order for child Els
6. DTD is not extensible
7. DTD is not simple to learn
8. DTD Provides less control on XML structure

XSD

(5)

1. XSD stands for XML schema definition
2. XSD are implemented in XML syntax
3. XSD support datatypes for Els and attributes.
4. XSD support namespaces
5. XSD defines order for child e
6. XSD is extensible
7. XSD is simple to learn. because you don't need to learn new lang
8. XSD Provides more control on XML structure.

(2) Complex type

Syntax: <xs:element = "Address">

```
<XS:complexType>
<XS:sequence>
<XS:element name="name" type="xs:string" />
<XS:  "    "  ="company"  "  "  />
<XS:  "    "  = "phone"  "  "  />
</XS:sequence>
</XS:complexType>
</XS:element>
```

global Type

XML

```
<customer_dob>
2
</customer_dob>
```

XSD

```
<XS:element name=
"customer_dob"
type="XS:date" />
```

Types of character entity

1. Predefined character entity
2. Numbered " " — &#decimal, &#x Hexadecimal
3. Named " " — A' acute accent, Ugrave — Ù

1. <, > — &comp.
   &apos — Single q
   &gt — greater
   &Lt — Less than
   &quot — Double quote

XML - Parser : XML document → client APP → XML parser

Validating — xml4c (IBM, in C++), xml4j (IBM, in Java)

non-validating — openxml (Java), Lark (Java), xp (Java),

xml Attribute

```
<a hreef = "http://www.tutorialpoint.com/">Tutorialspoint!</a>
```

hreef → attribute name; "http://www.tutorialpoint.com/" → attribute value

→ Same attribute can't have two values in a syntax. bcz the attri 'b' is specified twice

```
<a b="x" c="y" b="z"> ... </a>
```

→ Attribute names are defined without quotation marks,
→ Attribute values are appear in quotation marks

Not Allowed character	Replacement entity	character Description
<	≪	
>	>	
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Attribute Type

1. String Type
2. Tokenized Type
3. Enumerated Type

DOM: Document object model

→ The DOM defines a standard for accessing documents like XML & HTML.

→ DOM is Memory-based, this making it traversable & editable

→ DOM is not language-specific, nor is it platform-specific

XML DOM:

1. A standard object model for XML
2. A standard programming interface for XML
3. platform & language independent

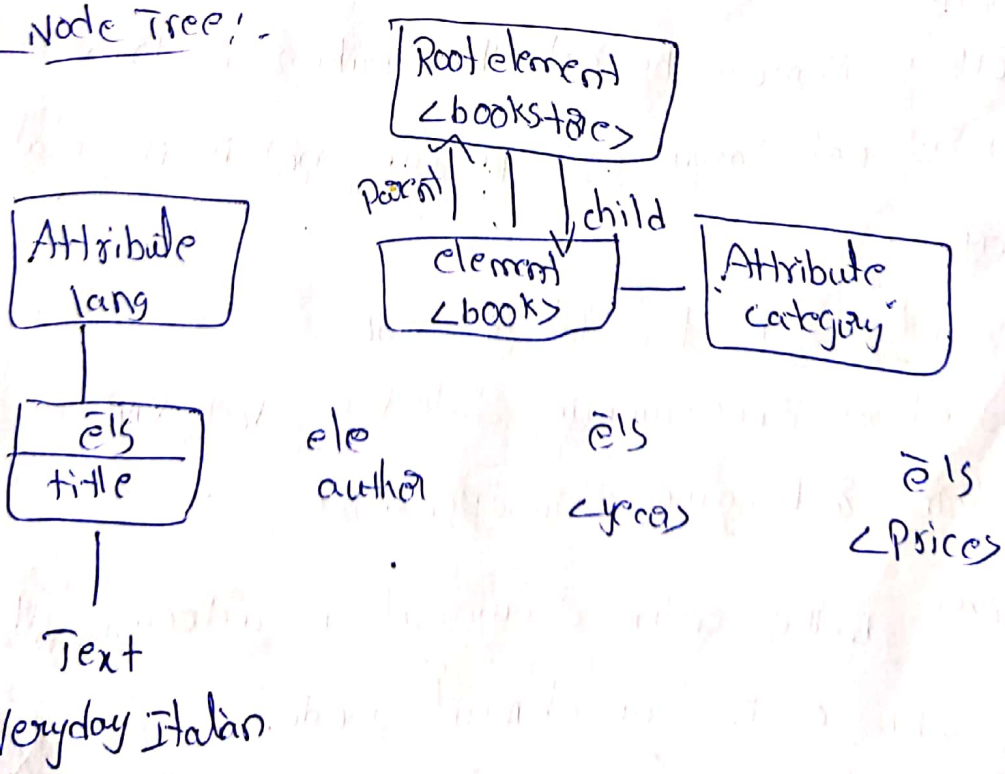
DOM Nodes: 1. The entire document is a document node.

2. Every XML e is an element node.
3. The Text in the XML e's are text nodes

DOM example

```
<?xml version="1.0" encoding = "ISO-8859-1"?>
    <bookstore>
        <book category ="cooking">
        <title lang = "eng">Everyday italian</title>
        <author> Giada De Laurentiis </author>
         <year> 2005</year>
         <price> 30.00 </price>
        </book>
      <book category="web" cover="Paperback">
         <title lang ="en"> Learning XML</title>
          <author> Erik. T. Ray </author>
          <year> 2003 </year>
          <price> 39.35</price>
          </book>
         </bookstore>
```

XML DOM Node Tree:-



XML DOM Methods:

-> x.getElementsByTagName(name) - get all el's with a specified tag name

-> x.appendChild(node) - insert a child node to x.

-> x.removeChild(node) - remove a child node from x.

DOM eg:-

```
txt=xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue
```

-> xmlDoc - the XML DOM object

-> getElementsByTagName("title")[0] - the first <title> element.

-> childNodes[0] - the first child of the <title> element (the text node)

-> nodeValue - the value of the node

NodeTree example: Parser

```
<html>
<head>
<script type="text/javascript" src="loadxmlstring.js"></script>
</head>
<body>
<script type="text/javascript">
text="<book>"
text=text+"<title>everyday italian</title>";
text=text+"<author>
+"</year>
+"</book>";
xmlDoc=loadXMLString(text);
x=xmlDoc.documentElement.childNodes;
for(i=0;i<x.length;i++)
```

```
{
document.write(x[i].nodeName);
do:  "  (": ");
"  "  (x[i].childNodes[0].nodeValue);
"  "  ("<br/>");
}
</script>
</body>
</html>
```

O/P

title: —
auth —
year.

DOM → web
SAX → Eve

DOM & SAX Parsers in java :-

DOM	SAX
1. It loads whole xml document in memory	1. It loads a small part of the xml file in memory
2. DOM ~~faster~~ Parser is faster than SAX bcz It accesses whole XML doc	
3. It works on DOM	3. event based

XML Parsers → read the XML & create a way for pgm's to use XML

XML Parsers : An xml parser is a slw library or package that provide interfaces for client application to work with an xml document.

XML document → API → XML Parsers → client Appli

Types of Parsers : 1. DOM
2. SAX

DOM	SAX
1. Document object model	1. Simple API for XML
2. It reads the entire document	2. It reads node by node
3. DOM is useful when reading small to medium size xml file	3. It is used when big XML file needs to be parsed
4. It is tree based parser	4. event based parser
5. DOM is little slow as compared to SAX	5. SAX is faster than DOM
6. DOM can insert & delete node	6. Can't insert & delete node

event Methods

<catalog> → start docum.
<catalog> — root els
<title> — start els
<title> shilpa <Hi
↓
characters
</catalog> → end element
↓
end document

Examples:- DOM TO parse the XML file

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
Public class Parsing_DOMDemo
{
static Public void main(String[] args)
{
try
{
```

DOM: It contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM:- A DOM Parser creates an internal structure in memory which is a DOM & the client application get itf of the original XML document by invoking methods on this document object

Advantages.

1. It supports both read & write operations & the API is very simple to
2. It is prefered when random access to widley separated Parts of a document is required.

Dis: 1. It is memory inefficient.

2. It is comparatively slower than other Parsers

SAX: It is event based API & less intuitive

Features: It does not create any internal structure

→ clients does not know what methods to call, they just override the methods of the API & place his own code inside method

Adv:

1. It is simple & memory efficient
2. It is very fast & works for huge documents.

Dis:

1. It is event based API is less intuitive
2. clients never know the full itf bcz the data is broken into pieces

D

```
System.out.print("Enter the name of XML document");
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
String file_name = input.readLine();
File fp = new File(file_name);
if (fp.exists())
{
 try
 {
DocumentBuilderFactory factory_obj =
 DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory_obj.newDocumentBuilder();
InputSource ip_src = new InputSource(file_name);
Document doc = builder.parse(ip_src);
System.out.println(file_name + " is well_formed!");
 }
 catch (Exception e)
  {
System.out.println(file_name + "isn't well_formed!");
    System.exit(1);
 }}
 else
  System.out.print("file not found");
 }}
 catch(IOException ex)
  {
 ex.PrintStackTrace();
 }}}
```

Student1.xml

```
<Student>
<Roll_NO>10</Roll_NO>
<Personal_Info>
  <Name>Path</Name>
<Address>Pune</Address>
<phone>123456</phone>
</Personal_Info>
<class>second</class>
<subject>M4</subject>
<Marks>100 <- this will be the case for crag
</Student>
```

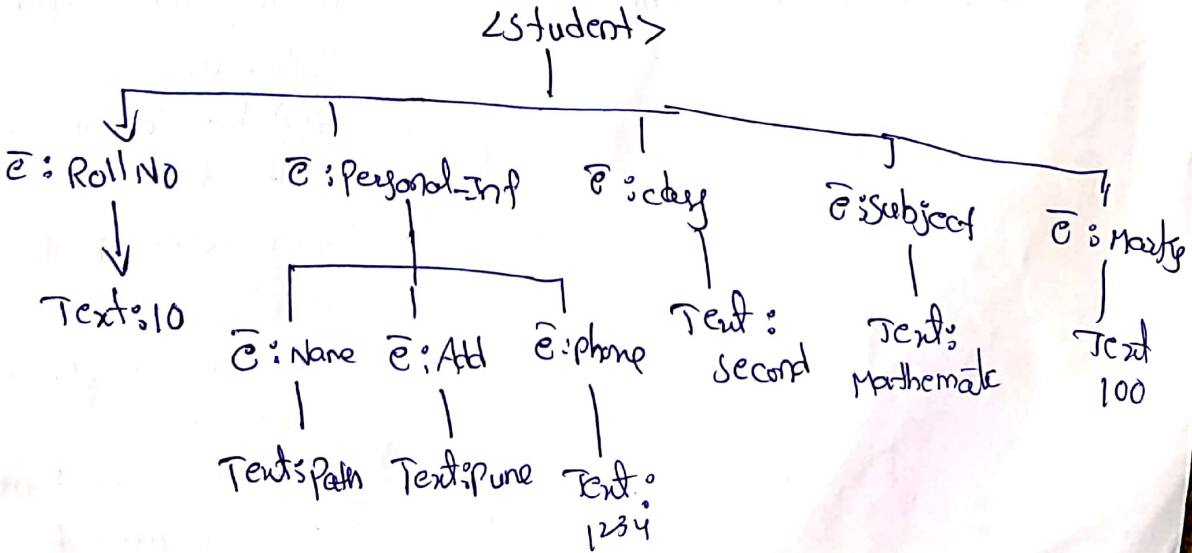
⇒ SAX Parser in Java example Program

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
Public class Parsing_SAXDemo
{
Public static void main(String[] args) throws IOException
{
try
{
System.out.print("Enter the name of xml document");
BufferReader input = new BufferReader(new InputStream Reader
String file_name = input.readline();          (System.in));
File fp = new file(file_name);
if(fp.exists())
{
try
{
```

```
XMLReader reader = XMLReaderFactory.createXMLReader();
  reader.parse(file_name);
  System.out.println(file_name+" is well-formed");
  }
catch(Exception e)
  {
System.out.println(file_name+"is not well-formed");
  System.exit();
  }}
  else
  {
  System.out.println("file is not present "+file_name);
  }}
  catch(IoException ex)
    {ex.printStackTrace();}
  }}
```

XHTML :- An XHTML document consists of three main parts.

1. DOCTYPE declaration
2. <head> section
3. <body> section



HTML	XHTML
1. The HTML tags are case insensitive. Hence <body> or <BODY> or <Body> are treated as one & the same.	1. The XHTML is case sensitive and all the tags in XHTML document must be written in lower case.
2. We can omit the closing tags sometimes in HTML document.	2. For every tage there must be a closing tag is not given There are two ways by which we can mention the closing tags or <a href="shilpa.html" />
3. In HTML the attribute value is not always neccessary to quote the attribute value. In fact numeric attribute values are rarely quoted in HTML.	3. In every XHTML docum the attribute value must b quoted.
4. In HTML there are some implicit attribute value	4. In every XHTML the attr values must be specified explicitly.
5. In HTML even if we do not follow the nesting rules strictly it does not cause much difference.	5. In XHTML document the nesting rules must be strictly followed. → A form e can not contain another form e. → An Anchor e does not Cont another form e → List e can't be nested in list e's. → If there are 2 nested e's inner e must be enclosed before closing the outer e

Servlets

Servlet:- Servlet is a technology which is used to create a web appli

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

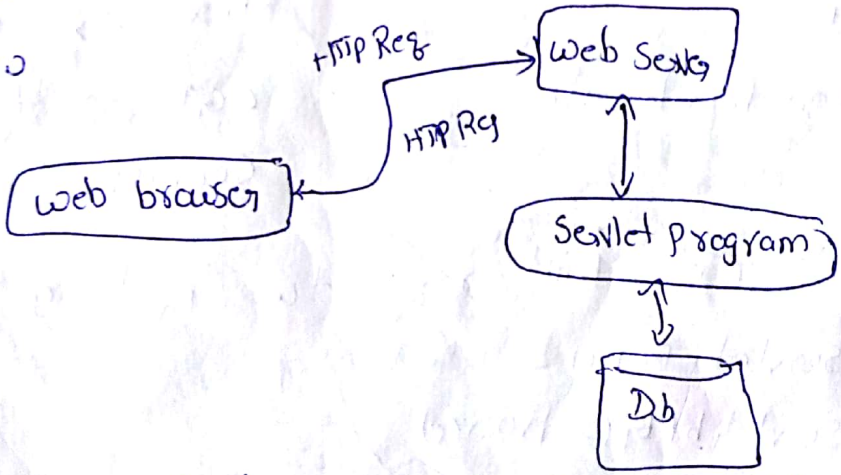
→ Servlet is a web component that is deployed on the server to create a dynamic web page.

→ A web application can be described as collection of web pages. & when we call it dynamic, it simply means that the web pages are not same for all the users. web pages would be generated on server side based on the req made by client

Properties of Servlets:-

1. Servlets work on the server-side
2. Servlets capable of handling complex request obtained from web server.

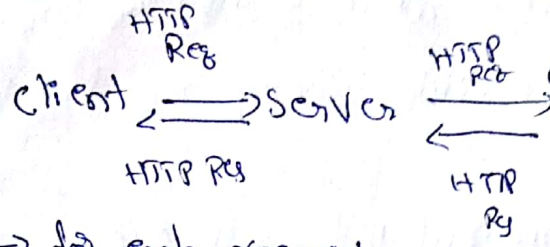
Servlet Architecture:



→ Servlet technology is robust & scalable bcz of java lang before Servlet, CGI scripting language was common as a server-side programming lang.

→ Many Interfaces & classes in the Servlet API such as servlet GenericServlet HttpServlet ServletReq, ServRes.

CGI:- It an external application which is written by using any of the Programming language like C or C++ this is repo for processing client req & generating dynamic content.



Features of CGI
1. Portable
2. Efficient & scalable
3. Reusable

→ for each request new process will be created

→ In CGI server has to create & destroy the process for every request.

Servlet	CGI
1. Servlets are portable & efficient	1. CGI is not portable
2. In Servlets sharing of data is possible	2 Sharing of data is not possible
3. Directly communicate with the web server	3 Can't directly Communi-cate with the web Ser
4. Servlets are less expensive than CGI.	4. CGI are more expensive than Servlets
5. Servlets can handle the Cookies	5. CGI can't handle the Cookies

→ Servlet API's (I'll's built from two packages

1. javax.servlet (Basic) – Protocol indepent - Generic servlet
2. javax.servlet.http(Advanced) – HttpServe

disadv CGI:- 1. If the number of clients, it takes more time for sending the response.
2. For each req, it starts a Process, and the web server is limited to start process
3. It uses platform dependent lang like C, C++

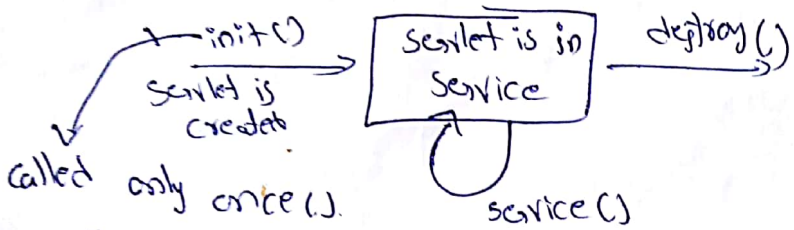
1. Better Performance:- It creates a thread for each req, not process
2. Portability:- because it uses Java language
3. Robust
4. Secure

~~Servlet life cycle:-~~

~~1. Servlet class is~~

→ Generic Servlet class has an abstract service() method. which means the Subclass of GenericServlet should always override the service() method

Servlet life cycle:-



→ Public void init() throws ServletException {

1) The servlet is initialized by calling the init() method
2) The servlet calls service() method to process the client's req
3) The servlet is terminated by calling the destroy() method

It is the main method to perform the actual task (i.e web server)
The service() method is handle to the req & response

→ The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc) & calls doGet, doPost, doPut, doDel

Syn:- Public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException {

}

→ doGet() & doPost() are most frequently used methods.

doGet():

Syn: Public void doGet(HttpServletRequest request, HttpServRes res) throws SerEx, IOEx

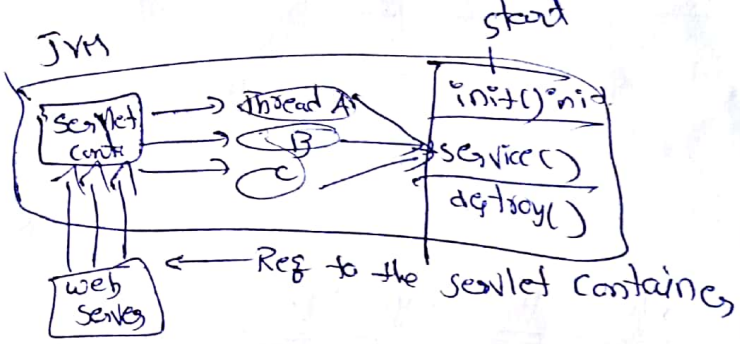
{

}

doPost():

Syn:-

destroy(). It is called only once at the end of the life cycle of a servlet



Syn:- Public void destroy()

{

}

init method Syntax:-

Public void init(ServletConfig config) throw ServletException

destroy():

Public void destroy()

Servlet Deployment:-

For creating a servlets you need Java platform so first install JDK in your system then setup environment variable for JDK.

Eg:- For JDK 1.8 environment variable value is

c:\Program files\Java\JDK 1.8.0\bin;

after JDK setup Install Apache Tomcat (or) any other server in your pc & set environment variable for server. for we need to set the servlet Jar file of your server. Jarfiles of different web server as

Jar file	Server
1. Servlet-api.jar	Apache Tomcat
2. weblogic.jar	weblogic
3. Javaee.jar	Glassfish
4. javaee.jar	JBoss

Eg:- we commonly use Apache Tomcat server Tomcat Jar file location in with reference Tomcat 8.

apache-tomcat-8.0.28\common\lib\servlet-api.jar

Creating a Servlet:-

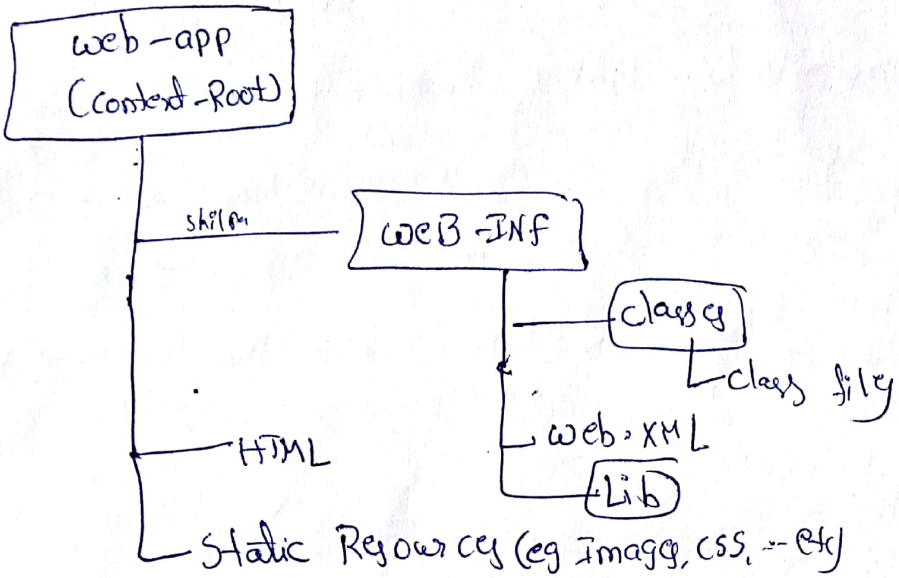
Servlet

1. By implementing Servlet Interface
2. By inheriting GenericServlet class (or)
3. By inheriting HttpServlet class

-> Mostely we are using HttpServlet because it provide Http request specific method such as doGet(), doPost(), doHead() etc

1. create a directory structure
2. Create a servlet
3. Compile the servlet
4. Create a deployment descriptor
5. Start the server & deploy the Project
6. Access the servlet

In Tomcat Server you need to deploy the Servlet class into web-app folder. In web-app you can create your own folder which represents the project which is displayed in url. Common structure for Tomcat source file is,



method - first letter small
words - S

DemoServlet.java:

```
import javax.servlet.http.*;        // class, HttpServReq
import javax.servlet.*;             // package
import java.io.*;                   // Package name, Initial Version, folder
Public class DemoServlet extends HttpServlet{   // class, obj
Public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
{
res.setContentType("text/html");    // setting the content type
PrintWriter pw = res.getWriter();   // class, object, method, get the stream & write data
pw.println("<html><body>");         // method
pw.println("welcome to servlet");
pw.println("</body></html>");
pw.close();                         // closing the stream
}}
```

Servlet API :- It provides classes & Interfaces to develop web based applications.

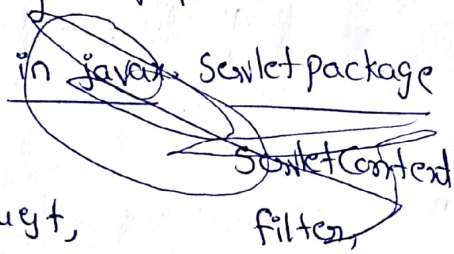
Package :- Servlet API containing two java packages are used to develop the Servlet Programs, they are.

1. javax.servlet
2. javax.servlet.http

1: javax.servlet package contains list of interfaces & classes that are used by the Servlet & web container. These classes & interfaces are not specific to any Protocol.

2: javax.servlet.http:. package contains list of classes & interfaces to define http Servlet Programs. This package are used to interact with browser using http protocol.

Interfaces in javax.servlet package



Servlet,
ServletRequest,
ServletResponse,
RequestDispatcher,
ServletConfig,
ServletContext
Filter,
FilterConfig,
FilterChain
ServletRequestListner
ServletReqAttributeListner
ServletContextListner
ServletContextAttributeListn

classes :- GenericServlet
ServletInputStream
servletoutputStream
ServletRequestwrapper
ServletResponsewrapper
ServletRequestEvent
ServletContextEvent
ServletRequestAttributeEvent
ServletContextAttributeEvent
ServletException
UnavailableException

Javax.Servlet Package Interfaces & classes

Interfaces

1. Servlet – Interface defining all life cycle methods
2. ServletRequest – Reading
3. ServletResponse – writing
4. RequestDispatcher
5. ServletConfig – Initialization Parameters
6. ServletContext – Interface events can be logged
7. SingleThreadModel
8. filter
9. FilterConfig
10. FilterChain
11. ServletRequestListener
12. ServletReqAttributeListener
13. ServletContextListener
14. ServletContextAttributeListener

classes

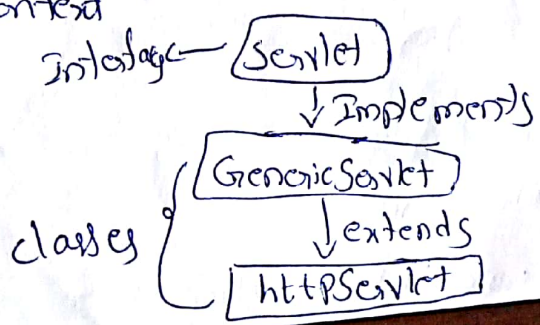
1. GenericServlet
2. ServletInputStream
3. ServletOutputStream
4. ServletRequestWrapper
5. ServletResponseWrapper
6. ServletRequestEvent
7. ServletContextEvent
8. ServletRequestAttributeEvent
9. ServletContextAttributeEvent
10. ServletException
11. UnavailableException

Interfaces in

1. HttpServletRequest
2. " " Response
3. HttpSession
4. HttpSessionListener
5. HttpSessionAttributeListener
6. HttpSessionActivationListener
8. HttpSessionContext

classes

1. HttpServlet
2. Cookie
3. HttpServletRequestWrapper
4. " " Response "
5. HttpSessionEvent
6. HttpSessionBindingEvent
7. HttpUtils



~~javax.Servlet~~

By implementing Servlet interface :

Eg:- Public class myServlet implements Servlet

{

}

By extending GenericServlet class

Public class myServlet extends GenericServlet

{

}

By Extending HttpServlet class

Public class myservlet extends HttpServlet

{

}

Methods of Servlet interface.

Method	Description
1. Public void init(ServletConfig config)	Initializes the servlet. It is th life cycle method of servlet & invok by the web container only once
2 Public void service(ServletReq req, ServletResponse response)	Req
3. Public void destroy()	is invoked only once & indicate that servlet is being destroyed
4. Public ServletConfig get GetServletConfig()	return the object of ServletCon
5. Public String getServletInfo()	returns info about servlet su

Eg:- implementing Servlet interface

```
Public class myServlet implements Servlet
{
}
Public void destroy()
{
}
Public void init(ServletConfig sc)
{
}
Public ServletConfig getServletConfig()
{
}
Public String getServiceInfo()
{
}
Public void Service(ServletReq req, ServletResponse res) throws IO Ex
                                                          Serv
{
}
```

1. The Servlet Interface:- This interface defines all the life cycle methods such as init(), destroy(), & service(). All the servlets must implement this interface.

Method	Description.
1. void init(ServletConfig s)	- This method is called for initialising the servlet. The initialization Parameters is obtained from the ServletConfig interface.
2. void destroy()	- This method is called when the servlet has to be unlo
3. ServletConfig getServletConfig()	- This method is used to obtain the initialis -ation parameters.
4. void service(ServletRequest req, ServletResponse res)	- This method is used to implement the service that a servlet should provide. The clients req is processed and a response is given
5. String getServletInfo()	This method is used to obtain description of the servlet.

2. The ServletConfig Interface: This interface is used to obtain the initialisation Parameters. Various methods that are used by the interface are

method	Description.
1. String getServletName()	- This method is used to obtain the name of the servlet.
2. String getInitParameter (String p)	- This method returns the value of the Parameter p.
3. Enumeration getInitParameter names()	- This method returns the names of initialisation Parameters.
4. ServletContext getServlet Context()	- This method returns the context for the servlet

Difference b/w Generic Servlet & HTTP Servlet

	Generic Servlet	HttpServlet
1.	Generic Servlet is a direct Subclass of Servlet interface	1. HttpServlet is a subclass of Generic Servlet
2.	Generic Servlet is protocol independent. It handles all types of protocol like http, smtp, ftp.	2. HttpServlet is protocol dependent. It handles only http protocol
3.	It supports only service() method	3. It supports public void service() doGet, doPost(), doDelete() doPut()
4.	It belongs to javax.servlet package	4. javax.servlet.http Package
5.	It is a must to use service() method as it is a callback method	5. Being service() is non-abstract, it can be replaced by doGet() or doPost() or

The javax.servlet Package.

Interfaces & classes

Interfaces	Description
1. Servlet	- This interface defines all the life cycle methods.
2. ServletConfig	- This interface obtains the initialization Parameters
3. ServletContext	- Using this interface the events can be logged.
4. ServletRequest	- This interface is useful in reading the data from the client request.
5. ServletResponse	- This interface is useful in writing the data to the client response.

3. The ServletContext Interface

Method	Description
1. object getAttribute(String attribute_name)	- The value of the attribute attribute_name in the current session is returned
2. void SetAttribute(String attribut_name, object value)	- The attribute_name is passed to the object value.
3 String getServerInfo()	- This method returns the information about the server.
4. String log(String str)	writes the str to the servlet log
5. String getMimeType(String file)	It returns the MIME type of the file

Reading

4. The ServletRequest Interface:

It helps in gathering information about the client. Various methods in ServletRequest are

Method	Description
1. objectgetAttribute(String attribute_name)	- The value of the attribute attribute_n in the current session is returned
2. int getContentlength()	- It returns the content size of the request
3. String getContentType()	- It returns the type of the request.
4. getProtocol()	- Returns the name of the Protocal used
5. getReader()	- This method is useful for reading the text from the request
6. getServerName()	- It returns the name of the Server on which the servlet is running

classes

class	Description
1. GenericServlet	This class implements the Servlet & ServletConfig interface.
2. ServletInputStream	This class provides the i/p stream for reading the client's request
3. ServletO/p Stream	This class provide the O/p stream for writing the client's response
4. ServletException	It is used to raise the exception when an error occurs.

1. GenericServlet class:- This class is useful for implementing the life cycle methods. It implements the Servlet & ServletConfig interface It is called "Generic" because it does not assume that the protocol it will process will be HTTP. If we want to write our own protocol, then we must extend this class.

2. The ServletI/PStream:- servletIPstream class extends the InputStream This class provide an i/p stream which is used by a servlet for reading the data from a client request. For reading the byte from a client the method used is readLine.

3. The ServletOutputStream class. It extends the O/P. This class Provide an o/p. stream. which is used by a servlet for writing the data for a client reponse. It also define Print and Println methods

Reading Servlet Parameters :-

Servlets handling from data parsing automatically using the following methods.

1. getParameter()
2. getParameterValues()
3. getParameterNames()

1:- you call request.getParameter() method to get the value of a form parameter

2:- Call this method if the parameter appears more than once and returns multiple value. eg: checkbox.

3:- Call this method if you want a complete list of all parameters in the current request

Name.html

```
<form action="Serv1" method="GET">
<input type="text" name="firstname"/>
<input type="text" name="lastname"/>
<input type="submit" name value="click"/>
</form>
```

HelloWorld.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class HelloWorld extends HttpServlet
  Private string mesage;
  {
Public Void init() throws ServletException
  {
   message = "Hello world";
  }
Public Void doGet(HttpServletReq re--.
  {
response.setContentType("text/html");
PrintWriter out=response.getWriter()
out.println(docType+"<html>\n"+
  "<body>\n"+"<b>firstName</b>
</b>\n"+request.getParameter
("firstname")"\n"+
  "<b>Lastname</b>\n"+
  request.getPar("Lastname")
  "\n"+
  "</body>
  </html>");
```

FirstServlet.java:

```
import java.io.*;
import javax.Servlet.*;
import javax.Servlet.http.*;
Public class FirstServlet extends HttpServlet
{
Public Void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IoException, ServletException
{
response.setContentType("text/html");
PrintWriter out= response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>My First Servlet</title>");
out.println("<body>");
out.println("<h1>welcome</h1>");
out.println("</body>");
out.println("</html>");
}
}
```

web.xml

```
<Servlet>
<Servlet-name>FirstServlet</Servlet-name>
<Servlet-class>FirstServlet</Servlet-class>
</Servlet>
<Servlet-mapping>
<Servlet-name>FirstServlet</Servlet-name>
<url-pattern>/Servlets/servlet/FirstServlet</url-pattern>
</Servlet-mapping>
```

R It allow you read the names & values of parameters that are include in a client request: A web page is defined in PostParameters.html and a servlet is defined in PostParameterServlet.java.

PostParameters.html

```
<html>
<body>
<center>
<form name="form1" method="post" action="http://localhost:8080/servlets-
    example/servlet/PostParametersServlet">
<table>
<tr>
    <td><B>Employee</td>
    <td><input type=textbox name="e" size="25" value="">
    </tr>
    <tr>
      <td><B>phone</td>
    <td><input type=textbox name="p" size="25" value=""></td>
    </tr>
    </table>
    <input type=submit value="submit">
    </body>
    </html>
```

PostParametersServlet.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
Public class PostParameterServlet extends GenericServlet
{
Public void service(ServletRequest request ServletResponse response)
    throws ServletException, IOException
    {
```

```
// Get Print writer.
PrintWriter pw = response.getWriter();
// Get enumeration of Parameter names.
Enumeration e = request.getParameterNames();
// Display Parameter names & values.
while(e.hasMoreElements())
    {
    String pname = (String)e.nextElement();
    pw.print(pname + " = ");
    String pvalue = request.getParameter(pname);
    pw.println(pvalue);
    }
    pw.close();
    }}
```

→ Compile the servlet. Next, copy it to the appropriate directory, and update the web.xml file as previously described

1. Start Tomcat
2. Display the web page in a browser
3. Enter an employee name & phone number in the text fields
4. Submit the web page.

Reading Initialization Parameters

DemoServletInit.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class DemoServletInit extends HttpServlet
 {
 Public void doGet(HttpServletRequest request, HttpServletResponse reppa)
      throws ServletException, IOException
      {
   response.setContentType("text/html");
   PrintWriter PW = response.getWriter();
  ServletConfig config = getServletConfig();
  String name = Config.getInitParameter("name");
  PW.Print("Name is : " + driver);
   PW.close();
  }
 }
```

web.XML

```
<Servlet>
<Servlet-name>D</Servlet-name>   (alias na)
<Servlet-class>DemoServletInit</Servlet-class>
<init-Param>
<Param-name>name</Param-name>
<Param-value>Shilpa</Param-value>
</init-Param>
</Servlet>
<Servlet-mapping>
<Servlet-name>D</Servlet-name>
<url-patten>/Demo</url-patten>
</Servlet-mapping>
```

O/P

name is Shilpa

Handling Http Request & Response

The HttpServlet class provides specialized methods that handle the various types of Http Requests.

Methods are:
doDelete(), doget(), doHead(), dooptions(), dopost(), doput() & doTrace().

Handling HTTP GET Requests.

A servlet that handly an Http GET request. The servlet is invoked when a form on a web page is submitted. The example contains 2 files. A web page is defined in ColorGet.html & Servlet is defined in ColorGetServlet.java

```
<html>
<body>
<center>
<form name="Form1" action="http://localhost:808/servlets-
        example/servlet/ColorGetServlet">
  <B>Color:</B>
  <select name="color" size="1">
  <option value="Red">Red</option>
  <option value="Green">Green</option>
  <option value="Blue">Blue</option>
  </select>
  <br>
  <input type=Submit value="Submit">
  </form>
  </body>
  </html>
```

The source code for ColorGetServlet.java is shown in the below. The doGet() method is overridden to process any HTTP GET req that are sent to this servlet. It uses the getParameter() method of HttpServletRequest to obtain the selection that was made by the user.

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

Public class ColorGetServlet extends HttpServlet
  {
  Public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
      {
      String color = request.getParameter("color");
      response.setContentType("text/html");
      PrintWriter pw = response.getWriter();
      pw.println("<B>The selected color is :");
      pw.println(color);
      pw.close();
      }
```

Handling HTTP POST Requests

A web page is defined in colorPost.html so

Servlet " " " ColorPostServlet.java

colorPost.html

```
<html>
<body>
<center>
<form name = "form1"
  method = "post" action = "http://localhost:8080/servlets-example/servlet/
    ColorPostServlet">
  <B> color</B>
<Select name = "color" size = "1">
<option value = "Red"> Red</option>
```

```
<option value="Green">Green</option>
<option value="Blue">Blue</option>
</select>
<br>
<input type=Submit value="Submit">
</form>
</body>
</html>
```

ColorPostServlet.java

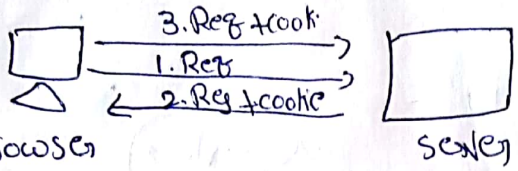
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class ColorPostServlet extends HttpServlet
{
Public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
String color = request.getParameter("color");
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>The selected color is");
pw.println(color);
pw.close();
}}
```

Using Cookies: A cookie is a small piece of information that is persisted b/w the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path & domain qualifiers, a max age, & a version number

How Cookie works

By default, each request is considered as a new request. In cookie technique, we add cookie with response from the Servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus we recognize the user as the old user.



Types of cookie: 2 types

1. Non-Persistent cookie
2. Persistent cookie

1: Non-Persistent Cookie: It is valid for single session only. It is removed each time when user closes the browser.

2: Persistent Cookie: It is valid for multiple session. It is not removed each time when user closes the browser. It is removed only if user logout or signout

Creating Cookie.

```
Cookie ck = new Cookie("user", "shilpa"); //creating cookie object
response.addCookie(ck); //adding cookie in the response
```

(name/key, value)

Delete Cookie

```
Cookie ck = new Cookie("user", ""); //deleting value of cookie
ck.setMaxAge(0); //changing the max age to 0 seconds
response.addCookie(ck); //adding cookie in the response
```

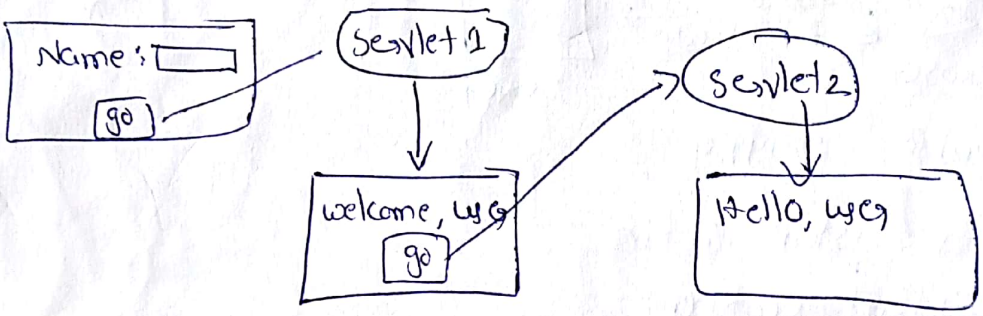
(60*60*24)

How to get Cookies :

```
Cookie ck[] = request.getCookies();
for(int i=0; i<ck.length; i++)
{
out.print("<br>"+ck[i].getName()+" "+ck[i].getValue(); // Printing
                                                        name & value of cookie
}
```

Simple example of Servlet Cookie

We are storing the name of the user in the cookie object & accessing in another servlet. As we know well that session corresponds to the particular user. So if you access it from to many browsers with diff values, you will get the diff value



index.html

```
<form action="Servlet1" method="post">
Name : <input type="text" name="userName" /> <br />
<input type="submit" value="go" />
</form>
```

FirstServlet.java

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

Public class FirstServlet extends HttpServlet
{
Public void doPost(HttpServletReq req)
{
```

```
try {
response.setContentType("text/html");
PrintWriter out=response.getWriter();
String n=request.getParameter("username");
  out.println("welcome "+n);
Cookie ck=new Cookie("uname",n); - cookie object
 response.addCookie(ck); // adding cookie
//creating submit button
out.print("<form action='servlet2'>");
out.print("<input type='submit' value='go'>");
 out.print("</form>");
 out.close();
    }
 catch(Exception e) {
   System.out.println(e); }
        } }
```

SecondServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class SecondServlet extends HttpServlet {
Public void doPost(HttpServletRequest req, HttpServ
                                                    )
 {
 try {
response.setContentType("text/html");
PrintWriter out=response.getWriter();
Cookie ck[]=request.getCookies();
out.print("Hello"+ck[0].getValue());
 out.close(); }
```

```
catch(Exception e) {
    System.out.println(e); }
}
```

O/P

Name: shilpa (Raj N

Go

welcom. shilps

Go

Hello shilpa

web.xml

```
<web-app>
<servlet>
<servlet-name>s1</servlet-name>
<servlet-class>FirstServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>
</web-app>
```

Cookie:

File	Description
1. AddCookie.html	- Allows a user to specify a value for the cookie named MyCookie
2. AddCoolsieServlet.java	- Process the submission of AddCookie.html
3. GetCoolkieServlet.java	- Display Cookie values.

AddCookie.html

```
<html>
<body>
<center>
<form name="form1" method="Post" action="http://localhost:8080/
servlets-examples/servlet/AddCookieServlet">
<B> Enter a Value for MyCookie</B>
<input type=a textbox name="data" size=25 value="">
<input type=Submit value="submit">
</form>
</body> </html>
```

The source code for AddCookieServlet.java is shown. It gets the value of the parameter named "data". It then creates a Cookie object that has the name "MyCookie" & containing the value of the "data" parameter.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class AddCookieServlet extends HttpServlet
  {
Public void doPost(HttpServletReq req HttpServletRes rs ) throws
                                    ServletException, IoException
  {
  // GET Parameter from HTTP Request
  String data = request.getParameter("data");
  // Add Cookie to HTTP response
  response.addCookie(cookie);
  // write output to browser.
  response.setContentType("text/html");
  PrintWriter pw = response.getWriter();
  pw.println("<B> MyCookie has been set to");
    pw.println(data);
    pw.close();
  }}
```

⇒

```
Public class GeetCookiesServlet extends HttpServlet{
Public void doGet(HttpServletReq                    )
// Get cookie from header of HTTP Request
  Cookie[] Cookies = request.getCookies();
// Display these cookies.
  response.setContentType("Text/html");
PrintWriter pw = response.getWriter();
  pw.println("<B>");
```

```
for(int i=0; i<cookie.length; i++)
  {
  String name = Cookie[i].getName();
  String value = Cookie[i].getValue();
    pw.println("name=" +name+ ";value=" +value);
      }
      pw.close();
    }
  }
```

Session Tracking:- Session simply means a particular interval of time.

Session Tracking is a way to maintain state(data) of an user. It is also known as session management in servlet.

Http Protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

A session can be created via the getSession() method of HttpServletRequest. An HttpSession object is returned.

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class DataServlet extends HttpServlet
  {
Public void doGet(Http.
  // Get the HttpSession object
  HttpSession hs = request.getSession(true);
  // Get writer.
  response.setContentType("text/html");
  PrintWriter pw = response.getWriter();
  pw.print("<B>");
```

```
//Display date/time of last accg
Date date=(Date)hs.getAttribute("date");
 if(date!=null)
 {
 pw.print("Last access: "+date+"<br>");
 }
date=new Date();
hs.setAttribute("date",date);
pw.println("current date:"+date);
}}
```

Cookie Ex:

1 set cookie: Cookie firstname = new Cookie("f_name", request.getParameter("first_name"));

firstname.setmaxAge(60*60*24);

response.addCookie(firstname);

2. Read Cookie getCookie

```
Cookie ck[] = request.getCookie();
out.println(ck[0].getName());
out.println(ck[0].getValue());
```

op f_name
EEC

⇒ addCookie() method of HttpResponse object is used to add Cookie to response object.

⇒ getCookie() method of HttpRequest object is used to return Cookie from browser

Session

1. The data remains on server-side
2. Unlimited size of data as per as server capability
3. It can store any type of data
4. How long the data should remain on server is not fixed
5. Session destroys after session timeout or logout
6. There is less data-traveling over the nlw
7. More secure mechanism to session tracking

Cookie

1. The data is on client side.
2. Limited support for data handling
3. It can store only text type of data
4. We can set the time for which data can remain on client side
5. Session remains on client machine
6. All cookie need to travel each time client side req to server.
7. It is less secure

UNIT-IV

Introduction to JSP: The Anatomy of a JSP page, JSP Processing, Declarations, Directives, Expressions, code Snippets, implicit objects, Using Beans in JSP Pages, Using Cookies and Session for Session tracking, Connecting to database in JSP.

Introduction to JSP:-

Servlet adds HTML code inside Java code

JSP Java code - Insert HTML

→ It stands for Java Server Pages.
→ It is a server side technology.
→ It is used for creating web application.
→ It is used to create dynamic web content.
→ In this JSP tags are used to insert Java code into HTML Page
→ It is an advanced Version of Servlet Technology.
→ It is a web based technolgy helps us to create dynamic and platform independant web pages.
→ In this, Java code can be inserted in HTML/XML Pages or both.
→ JSP is first converted into servlet by JSP container before Processing the client's request.

* Jsp Pages are more advantageous than Servlet.
  → they are easy to maintain
  → JSP are extended Version of Servlet.

Advantages of using JSP:-

1. It does not require advanced knowledge of JAVA.
2. It is capable of handling exceptions
3. Easy to use and learn. 8. High Perform & scalability
4. It can tags which are easy to use and understand
5. Implicit objects are there which reduces the length of code.
6. It is suitable for both JAVA & non JAVA programmes
7. JSP is build on Java Technology, So it is platform independent

Disadvantages of using JSP

1. Difficult to debug for errors. 3. Easy to maintain & code
2. First time access leads to wastage of time.

Features of JSP:

1. **Coding in JSP is easy:-** As it is just adding JAVA code to HTML/XML
2. **Reduction in the length of code:-** In JSP we use action tags, custom tags
3. **Connection to Db is easier:-** It is easier to connect website to db and allows to read & write data easily to the db.

JSP Syntax:-

1. **Declaration Tag:** It is used to declare Variable.

Syntax:- <%! Dec Var %>

Eg:- <%! int Var=10; %>

2. **Java scriplets:-** It allows us to add any number of JAVA code, Variable and expressions.

Syntax:- <% Java code %>

3. **JSP Expression:-** It evaluate and convert the expression to a string

Syntax:- <%= expression %>

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

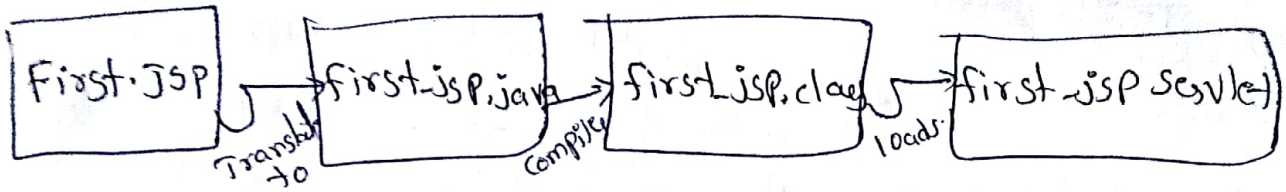
4. **JAVA Comments:-** It contains the text that is added for information which has to be ignored.

Syntax:- <%--- JSP comments %>

→ **Why JSP is preffered over Servlets**

→ JSP provide an easier way to code dynamic web page
→ JSP does not require additional files like java class file, web.xml
→ Any change in the JSP code is handled by web container

-> JSP Pages are converted into Servlet by the web Container. The container translates a JSP Page into servlet class source (.java) file and then compiles into a Java Servlet class.



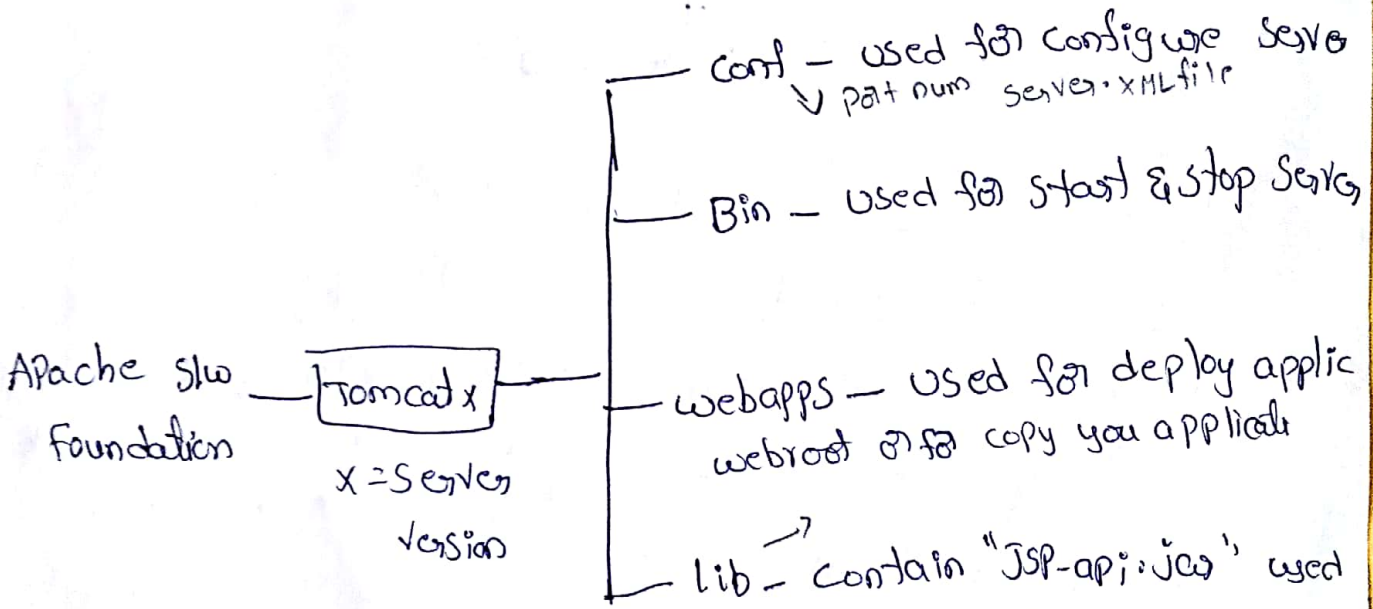
Setclasspath for JSP:-

Installation of JSP:- Download tomcat from the following url

https://tomcat.apache.org/download-70.cgi

"c:\Program file\apache Software foundation\tomcat 6.0"

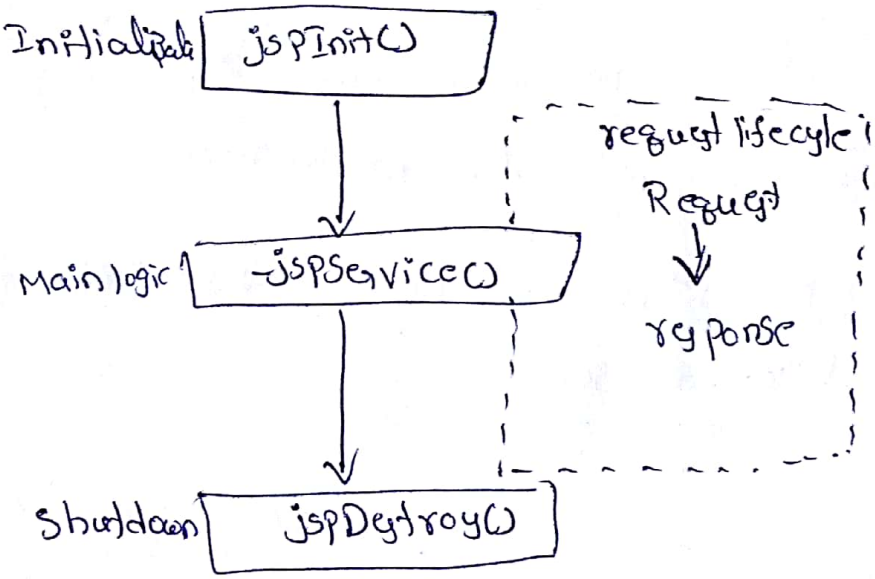
Hierarchy of Tomcat Server:



Life-cycle of JSP:-

A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to translate a JSP into servlet.

1. Translation of JSP page to servlet code.
2. Compilation of servlet to bytecode.
3. Loading Servlet class
4. Creating Servlet instance.
5. Initialization by calling jspInit() method.
6. Request processing by calling _jspService() method.
7. Destroying by calling jspDestroy() method.

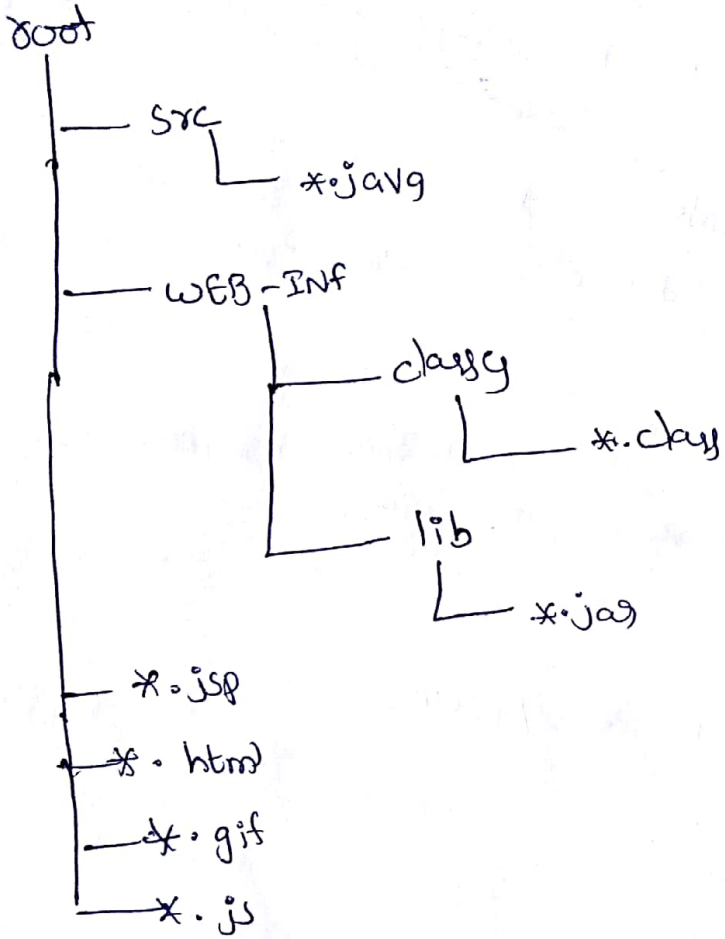


Difference between Servlet and JSP:

Servlet	JSP
1. Servlet is faster than JSP	1. JSP is slower than Servlet because it first translate into Java code then compile.
2. Servlets are the Java Programs that can be compiled to generate dynamic content	2. JSP is a scripting language that generate dynamic content
3. Servlet is a Java code	3. JSP is tag based approach
4. Servlets have no facility of calling Java Bean	4. JSP can directly call Java beans

5. coding of Servlet is harden than JSP.	5. coding of JSP is easier than servlet because it is tag base.
6. Servlet accept all Protocol request.	6. JSP will accept only http Protocol request
7. There is no facility of creating custom tags	7. JSP can build custom tags.

Directory structure of JSP:-



First JSP Program

```
<html>
<body>
<%. out.print("Hello world"); %>
</body>
</html>
```

O/P

hello world

Run jsp program:-

1. Deploy jsp code in root directory.
2. Start server

Syntax:- http://localhost:portnumber/RootDirectory/jspfile

Ex:- http://localhost:2015/jspapplication/shilpa.jsp

Anatomy of a JSP Page:-

→ JSP page is a simple web page which contains the JSP elements and template text.

→ The template text can be scripting code such as Html, xml or a simple plain text.

→ Various JSP elements can be action tags, custom tags, JSTL library elements. These JSP elements are responsible for generating dynamic contents

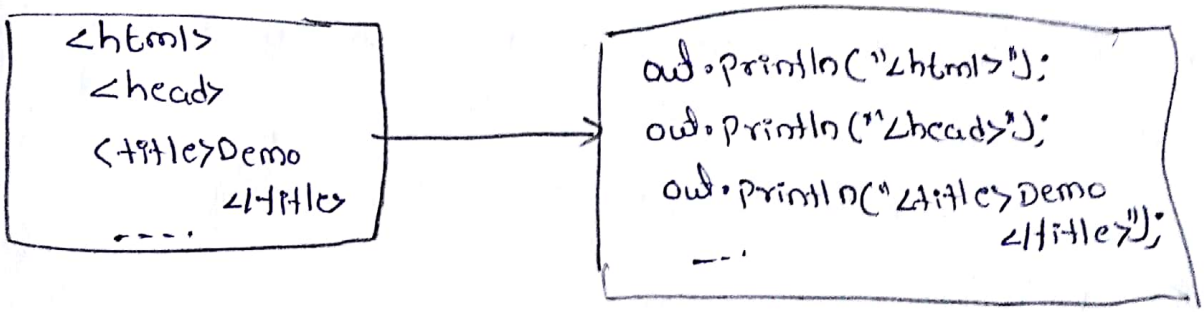
```
<%@ page language="java" contentType="text/html" %>
<%@ page import="java.util.Date;" %>
  <html>
  <head>
  <title> First A JSP Program </title>          } Template Text
  </head>
  <body>
  <%
    out.println("Helo B.Tech CSE students");
    out.println("<br><br>");
    out.println("welcome to JSP Programming");   } JSP elements
    out.println(new Date().toString());
  %>
  <center> Have a nice day </center>             } Template Text
  <br>
  </body> </html>
```

JSP / Custom / Action / Library

when Jsp request gets processed template text & Jsp (4) elements are merged together and sent to the browser as response.

Jsp Processing:- Jsp Pages can be Processed using Jsp Container only.

1. client makes a request for required Jsp page to the server. the server must have Jsp container so that Jsp request can be Processed. for instance: Let the client make a request for xyz.jsp Pg.
2. On receiving this request the Jsp container searches and then reads the desired Jsp Page. Then this Jsp Page is straightaway converted to corresponding Servlet. Basically any Jsp page is a Combination of template text & Jsp element. Every template text is translated into corresponding println statement.



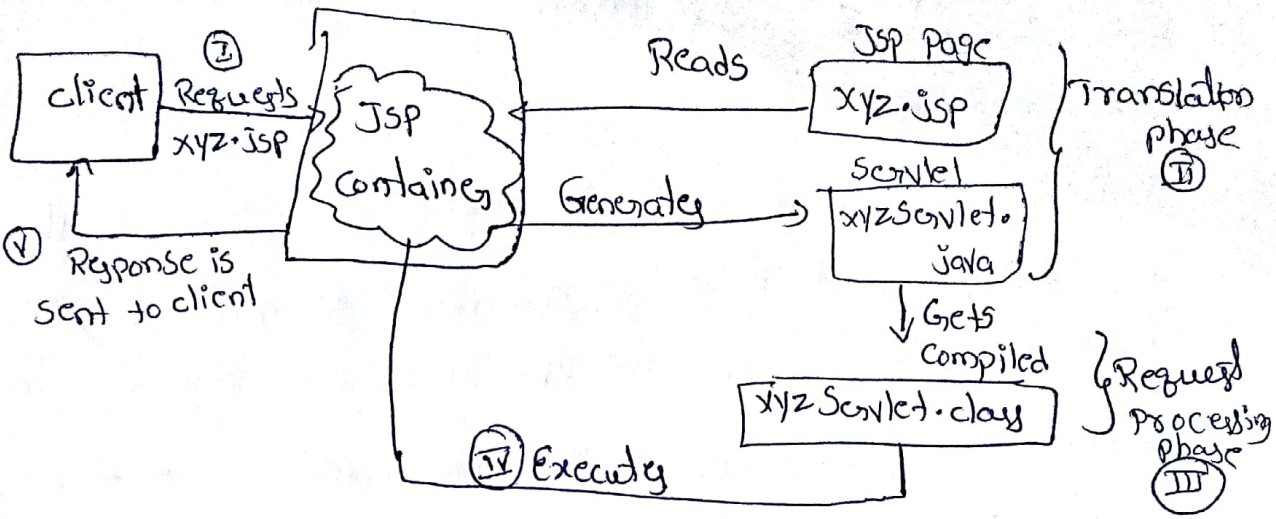
Every Jsp element is converted into corresponding Java code. This phase is called translation phase. The output of translation phase is a Servlet.

for eg:- Our xyz.jsp gets converted to xyzServlet.java

3. ~~This Servlet is then~~
3. Basically any Jsp Page is a combination of template text & Jsp elements.

4. This Servlet is then compiled to generate the Servlet class file. Using this class the response can be generated. This phase is called "request processing phase."

5. The Jsp container thus execute the servlet class-file
6. A requested page is then returned to the client as a response.



Jsp Declarations:-

At the end a Jsp page is translated into servlet class. So when we declare a variable or method in Jsp inside Declaration Tag, it means the declaration is made inside the servlet class but outside the service

```
<%! declaration %>
```

Example:-

```
<html>
 <head>
  <title>My first Jsp Page</title>
 </head>
 <%!
    int count=0;
 %>
  <body>
      Page Count is:
        <% out.println(++count); %>
  </body>
 </html>
```

In the above code. we have used the declaration tag to declare variable count.

Servlet Declaration Tag

```
Public class hello_jsp extends HttpServlet
{
int count=0;
Public void _jspService(HttpServletRequest request, HttpServletResponse res)
                                    throws IOException, ServletException
  {
  PrintWriter out = response.getWriter();
   response.setContentType("text/html");
   out.write("<html><body>");
    out.write("Page count is:");
    out.Print(++count);
    out.write("</body></html>");
```

Ex:

```
<%@ Page language="java ContentType="Text/html"%>
<%!
   String msg = "Hello";
 %>
<%! Public String Myfunction(String msg)
  {
   return msg;
   } %>
<html>
 <head>
  <title> use of method</title>
  </head>
  <body>
<% out.Print("Before function call "+msg); %>
  <br/>
After function call : <%= Myfunction("Publications") %>
  </body>
  </html>
```

O/P

Before function call: Hel
After function call: Technical Publi

JSP Directive: JSP directive control the processing of entire JSP Page.

→ The jsp directive are message that tells the web container how to translate a JSP Page into the corresponding servlet.

3 Types of Directive

1. Page directive 2. Include directive 3. Taglib directive

Syntax:- <%@ directive attribute="value" %>

1. Page directive:- The page directive define the attribute the apply to the entire JSP page.

Syn:- <%@Page attribute="value" %>

Attributes of JSP page directive

1. import
2. ContentType
3. extends
4. language
5. Info
6. buffer
7. isELIgnored
8. isThreadSafe
9. autoFlush
10. Session
11. Page encoding
12. Error Page
13. isErrorPage

Import attribute: The import attribute is used to import class, interface of all the members of a package

```
<%@page import="java.io.*" %>
<%@ page import="java.util.Date" %>
Today is : <%= new Date() %>
</body>
</html>
```

Content Type: It define the MIME (Multipurpose Internet Mail Extension) type of the HTTP response. The default value is "text/html; charset=ISO-8859-1".

Syn: <%@ page contentType="Text/html" %>

3. Extends:- The extends attribute defines the Parent class that will be inherited by the generated Servlet.

4. language:- The language attribute specifies the scripting langu. used in the JSP Page. The default value is "Java".

5. Info :- This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

Syn:- <%@Page info = "Composed by Shilpa" %>

The webcontainer will create a method getServletInfo() in the resulting Servlet.

Eg:-
```
Public String getServletInfo()
{
return "composed by Shilpa";
}
```

② Include Directive:- Include directive ~~basically allows us to use Custom tags in JSP.~~ is used to copy the content of one JSP Page to another

Eg:-
```
<%@include file="calculate.jsp" %>
<html>
<body>
<%@include file="header.html" %>
Today is <%= java.util.Calendar.getInstance().getTime() %>
</body>
</html>
```

③ Taglib directive:- It is used to define a tag library that defines many tags. We use the TLD(Tag library Descriptor) file to define the tags

Syn:- <%@ taglib uri = "mytaglib" prefix = "Sampletag" %>

Eg:- <html>
<body>
<%@ taglib uri="http://www.javapoint.com/tags" prefix="mytag" %>
<mytag:CurrentDate/>
</body>
</html>

JSP Expressions:- Expression tag is used to find the result of the given expression and send that result back to the client browser

⇒ Each Expression tag of JSP will be internally converted into out.print statement. In JSP out is an implicit object

Syntax:- <%=Statement %>

Ex:- <html>
<body>
<% ="welcome to jsp" %>; → do not end with ; in case of expression tag
</body>
</html>

⇒ To display the current time we have used the getTime() method of Calendar class

<html>
<body>
Current Time <%=java.util.Calendar.getInstance().getTime() %>
</body>
</html>

Eg:- index.jsp

<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go">
</form>
</body>
</html>

welcome.jsp

<% ="welcome+" request.getParameter("uname") %>

Jsp Implicit objects There are 9 jsp implicit objects.

These objects are created by the web container that are available to all the jsp pages.

object	Type
1. out	JspWriter
2. Request	HttpServletRequest
3. Response	HttpServletResponse
4. Config	ServletConfig
5. application	ServletContext
6. Session	HttpSession
7. PageContext	PageContext
8. Page	object
9 Exception	Throwable.

1. **out Implicit object**:- This object used for writing any data to the buffer.

Syntax:- `PrintWriter out = response.getWriter();`

Example:-

```
<html>
<body>
<%
out.print("Today is:"+java.util.Calender.getInstance().
getTime());
%>
</body>
</html>
```

O/P

Today is Mon April 15, 13:21:20 IST 2015

2. **Request Implicit object**:- The JSP request is an implicit object of type HttpServletRequest i.e created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc

It can also be used to set, get & remove attribute from the jsp request scope.

Eg:- 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);
%>
```

Response Implicit object : In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServlet Response is created by the web container for each jsp request.

It can be used to add or manipulate response such as redirect response to another resource, send error etc. where we are redirecting the response to the Google

Eg:- index.html

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="Submit" value="go"><br/>
</form>
```

welcome.jsp

```
<% response.sendRedirect("http://www.google.com");
%>
```

JSP config implicit object :- object type is ServletConfig this object can be used to get initialization Parameter for a Particular JSP Page.

It is used to get initialization parameters from the web.xml file

index.html

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br>
</form>
```

web.xml

```
<web-app>
<servlet>
<servlet-name>Home</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>
<init-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</parameter value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>/Home</servlet-name>
<url-pattern>welcome</url-pattern>
</servlet-mapping>
</web-app>
```

welcome.jsp

```
<% out.print("welcome "+request.getParameter("uname"));
String driver=config.getInitParameter("dname");
out.print("driver name is ="+driver);
%>
```

JSP application implicit object :- is an object of type ServletContext for all jsp in a web application, there must be a single application object we can share the data from one JSP to any other JSP in the web application.

The instance of ServletContext is created only once by the web container when application or project is deployed on the server

index.html :

```
<form action = "welcome.jsp">
<input type = "text" name = "uname">
<input type = 'Submit" value = "go"><br>
</form>
```

welcome.jsp :

```
<% out.print("welcome "+request.getParameter("uname"));
String driver = application.getInitParameter("dname");
out.print(" driver name is = "+driver);
%>
```

web.xml :

```
<web-app>
<servlet>
<servlet-name>S</servlet-name>
<jsp-file>/welcome.jsp</servlet-class>
</servlet>
<servlet-mapping>
<Servlet-name>S</Servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>
<Context-Param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>
</web-app>
```

Using Beans in JSP:-

Bean are reusable components. These actually instances of Java class with getter and setter methods. It helps in separation of business and presentation logic.

JavaBeans are simple classes that are used to develop dynamic web pages. by using separate java classes instead of using java code in a Jsp page. Using JavaBeans it is easy to share objects between multiple webpages.

Java Bean Properties:-

1. getPropertyName(); It is used to read the property. This method is called accessor.
2. SetPropertyName(); It is used to write the property. This method is called mutator.

UseBean Tag:-

<jsp:UseBean> Tag is used to instantiate a JavaBean or to locate existing bean instance and assign it to a variable name

Syntax:- <jsp:useBean> action tag:

```
<jsp:usebean id="bean name" scope="scope name"
class="packageName.className" beanName="PackageName.
      className" type='PackageName.className"/>
```

where

id: It represents variable name assigned to id attribute of UseBean tag.

Scope:- It represents scope in which bean instance has to be located. Scopes may be Page, request, session, & application. Default scope is page.

Page:- It indicates that bean can be used within Jsp Page until Page sends response to client or

forwards request to another resource.

Request:- It indicates bean can be used from any JSP Page that Processing same request until JSP Page sends response to the client.

Session:- It indicates that bean can be used from any JSP Page involved in the same session. The page in which we create bean must have page directive with session = "true".

Application:- It indicates that bean can be used from any JSP page in the same application

class:- It takes class name to create a bean instance if bean instance not present in given scope. It should have no argument constructor and should not be an abstract class.

beanName:- It takes class name or expression.

Type:- It takes a class or interface name which can be used with class or beanName attribute. It is attribute can be used with or without class or beanName.

Example Program:-

Creation of Bean: StudentBean.java

```
Package javabean.net.jsp.beans;
Public class StudentBean implements java.io.Serializable
{
Private String name = null;
Private int enrollment_no = 0;
Private String address = null;
Private double cpi = 0.0;
Public StudentBean()
  {
  }
```

```
Public String getName()
 {
 return name;
 }
Public int getEnrollment_no()
 {
 return enrollment_no;
  }
  Public String getAddress()
   {
   return address;
  }
  Public double getCpi()
   {
  return cpi;
 }
Public void setName(String name)
 {
   this.name = name;
 }
Public void setEnrollment_no(int enrollment_no)
   {
   this.enrollment_no = enrollment_no;
 }
 Public void setAddress(String address)
  {
    this.address = address;
  }
  Public void setCpi(double cpi)
    {
    this.cpi = cpi;
    }
  }
```

JSP Program

```
<html>
<head>
<title> BEAN and JSP Example </title>
```

→ Cont in Pag-(11)-backside

Write a Jsp program to validate user name & password

```
<html>
<head>
<title> Login using JSP </title>
</head>
<body>
<h1>LOGIN FORM</h1>
<form action="checkLogin.jsp">
<table>
<tr>
<td> Username : </td><td> <input name = "username"
size =15 type = "text" />
</td>
</tr>
<tr>
<td>Password : </td>
<td> <input name = "password" size =15 type = "text" />
</td>
</tr>
</table>
<input type = "Submit" value = " Login" />
</form>
</body>
</html>
```

Step 2: The Jsp code that validate the username & pw

```
<%
String username = request.getParameter("username");
String password = request.getParameter("password");
out.println("checking login<br>");
```

```
if(username == null || password == null)
  {
out.print("Invalid parameters");
}
if(username.toLowerCase().trim().equals("admin")
&& password.toLowerCase().trim().equals("admin"))
    {
     out.println("welcome" + username);
     session.setAttribute("username", username);
    }
else
{
 out.println("Invalid username and password");
}
  %>
```

```
</head>
<body>
<jsp:useBean id="Students" class="StudentBean">
<jsp:setProperty name="Students" property="name" value="SS"/>
<jsp:setProperty name="Students" property="enrollment_no" value="10"/>
<jsp:setProperty name="Students" property="address" value="Hyd"/>
<jsp:setProperty name="Students" property="cpi" value="9.2"/>
  </jsp:useBean>
  <p> Student Name:
<jsp:getProperty name="Students" property="name"/>
  </p>
<p> Student Enrollment no:
<jsp:getProperty name="Students" property="enrollment_no"/>
  </p>
<p> Student Address
<jsp:getProperty name="Students" property="cpi"/>
  </p>
  </body>
  </html>
```

Using Cookie and Session for Session Tracking:-

Cookie: Cookie are the small text files that are stored in the client's computer

→ They are basically used to keep track of the users who browse the web. the information stored in the cookie is generally name, age, id, city ...

→ The server script sends a set of cookie to the browser the browser stores the information on the local machine and make use of this information next time when the browser is browsing the web.

Implementation of Cookie in JSP:

Create an html file to read the values from the user,

Input.html

```
<html>
<body>
<form method="post" action="CreateCookie.jsp">
Username <input type="text" name="name">
City<input type="text" name="city">
<input type="Submit" name="Submit" value="Submit">
</form>
</body>
</html>
```

CreateCookie.jsp:

```
<%@ page language="java" import="java.util.*" %>
<%
String name=request.getParameter("name");
    String city=request.getParameter("city");
Cookie namecookie=new Cookie("name",name);
Cookie cityCookie=new Cookie("city",city);
response.addCookie(namecookie);
response.addCookie(cityCookie);
namecookie.setMaxAge(60*60*24);
citycookie.setMaxAge(60*60*24);
%>
<h3>
<a href="Readcookie.jsp"> click here to continue..</a>
</h3>
```

ReadCookie.jsp

```
<h3>Reading the cookie</h3>
<%
Cookie[] cookieg= request.getCookieg();
for(int i=0; i<cookieg.length; i++)
{
out.Println("Cookie_Name"+Cookieg[i].getName()+"<br>");
out.Println("Cookie_value"+ cookieg[i].getValue()+"<br>");
}
%>
<a href= "DeleteCookie.jsp"> click here to Delete the cookieg!!</a>
```

DeleteCookieg.jsp:

```
<%
Cookie namecookie= new Cookie("name","");
nameCookie.setMaxAge(0);
namecookie.setValue("");
response.addCookie(namecookie);
Cookie citycookie= new Cookie("city","");
citycookie.setMaxAge(0);
citycookie.setValue("");
response.addCookie(citycookie);
%>
<h3> Cookieg is deleted </h3>
<a href ="Readcookie.jsp"> click here to check the deletion.. </a>
```

OIP

UserName : S city : Hyd Submit

click here to Continue

Reading the cookieg

Cookie_name: name

Cookie_value : S

Cookie-Name: city Hyd

click here to Delete the cookie

If we click for deletion of the cookies, then the cookie gets deleted

Cookie is Deleted!!

click here to check the deletion

Session Tracking:- This method of keeping track of all the information between server and browser using Session-ID is called session tracking.

CounterDemo.jsp

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<head>
<title> Hello </title>
</head>
</body>
<c:set var="first_cnt" scope="session" value="${first_cnt+1}"/>
<c:set var="second_cnt" scope="application" value="${second_cnt+1}"/>
<h3>
<p> WELCOME</p></h3>
The session count of this page is ${first_cnt}
The application count of this page is ${second_cnt}
</body>
</html>
```

Session methods:-

1. getAttribute: It returns stored value from session object it return null if no value is associated with name.
2. SetAttribute:- It associates a value with name
3. RemoveAttribute:-
4. getAttributeNames:-

5. getId: It return the unique id in the session
6. IsNew: It determine if session is new to client
7. getCreationTime: It returns time at which session was created
8. getLastAccessedTime: It returns time at which session was accessed last by client
9. getMaxInactiveInterval:- It gets max amount of time session in seconds that access session before being invalidated.
10. setMaxInactiveInterval: It sets maximum amount of time session in seconds b/w client req before session being invalidated.

Connecting to database in jsp

The database is used for storing various type of data which are huge and has storing capacity in gigabytes. JSP can connect with such databases to create and manage the records.

1. The database named books is created.
2. The Table in the books database is books_info which is as given below.

ISBN	Name	author
1	DSF	Tannebaum
11	DC	Kelvin
22	OS	Peterson
42	Unix Programming	Bach
55	JavaServerPage	Hary Bergsho
57	DAA	Coleman

input.JSP

```
<%@ Page language="java" import="java.sql.*"%>
<%@ Page import="java.io.*"%>
<%@ Page import="java.sql.Connection"%>
<%@ Page import="java.sql.DriverManager"%>
<%@ Page import="java.sql.ResultSet"%>
<%@ Page import="java.sql.ResultSetMetaData"%>
<%@ Page import="java.sql.Statement"%>
<%
Connection conn=null;
ResultSet rs=null;
Statement stmt=null;
Class.forName("com.mysql.jdbc.Driver").newInstance();
conn=DriverManager.getConnection("jdbc:mysql://localhost:
        3306/books","root","System");
out.write("Connected to mysql");
stmt=conn.createStatement();
rs=stmt.executeQuery("select * from books_info");
%>
    while(rs.next())
    {
  %>
<tr>
<td><%=rs.getString("ISBN")%></td>
<td><%=rs.getString("name")%></td>
<td><%=rs.getString("author")%></td>
</tr>
<%
}
rs.close();
stmt.close();
conn.close();
%>
```

```
</table>
</center>
</body>
</html>
```

O/P:- Connected to mysql

ISBN	BookName	Author
1	DSF	Tannenbaum
11	DC	KELVIN
22	OS	PETERSON
42	UNIX	BACH
55	Java	Hans Bergstan
57	DAA	coleman.

Code Snippets: The code that appears between the <% and %> ddimitery is called a scriplet. Scriplets are nothing but java code enclosed within <% and %> tags.

Templatetext.jsp

```
<%@ page language="java" contentType="text/html"%>
<html>
<head>
<title> snippets</title>
</head>
<body bgcolor="gray">
<h1>Hi</h1>
<h2> Hello</h2>
<li> welcome</li>
<p>
<% out.print("JSP is equal to HTML and JAVA");%>
</p>
</body>
</html>
```

UNIT-V

Client Side Scripting:

Introduction to Javascript, Javascript language, declaring variable, scope of variable, functions, event handler (onclick, onsubmit), DOM from validation simple AJAX application.

Introduction to Javascript:- Javascript is a scripting language most often used for client-side web development.

→ client-side refers to operations that are performed by the client (in our case the client is the browser) in a client-server relationship.

→ Javascript was designed to add interactivity to HTML pages

→ A scripting language is a lightweight programming language

→ A Javascript consists of lines of executable computer code

→ A Javascript is usually embedded directly into HTML pages

→ Javascript can be used to create cookies

→ Simple web applications such as calculators, calenders can be developed using Javascript

→ Javascript can be read & write HTML elements

→ Javascript can be used to validate data

→ Javascript originally Livescript

Example program.

```
<html>
<head>
<title>My first Javascript</title>
</head>
<body>
</center>
<script type="text/javascript">
```

<body> Inline
<head> Internal
<External> .js file

document.write("welcome to first page of Javascript");

</script>

</body>
</html>

function is used to display dynamic content through JS

⇒ <script> - Tag tells the browser where the scripting starts (<script type = "text/javascript">) & ends (</script>). Type attribute indicates the type of SL

⇒ document.write :-

The word document.write is a standard Javascript command for writing o/p to a page

Comments in Javascript :

→ The Javascript allows 2 kinds of comments

1) Single line Comment → denoted as //
2) Multiline Comment → " " /* */

Eg :-

```
<html>
<head>
<title>my Javascript</title>
</head>
<body>
<center>
<script>
document.write("Below I have some comment statements");
//This is a single line Comment
/* we can enter multiple comments in javascripts */
</script>
</center>
</body>
</html>
```

→ difference btw Server Side Scripting and Client Side Scripting

1. Server Side Scripting :-

→ The Server Side Scripting is used to create the web page that provide some Service.

→ These scripts generally run on web Servers.

→ A users request is fulfilled by running a script directly on the web Server to generate dynamic HTML pages. The HTML is then Sent to the client browser.

→ Use: Processing of user request, accessing to database.

→ Ex: PHP, ASP.NET, C++, Java & C#

2. Client Side Scripting :-

→ The client Side Scripting is used to create the web page as a request & response to Server. These pages are displayed to the user on web browser (JS, Livescript, VB,

→ These Scripts generally run on web browser.

→ The processing of these scripts takes place on the end user Comp-uters. The source code is transfered from the web Server to the user Computer over the internet & run directly in the browser.

→ Use: Making interactive web Page, for interacting with temporary storage such as cookie & local storage, Sending request to Server & getting the response & displaying that response in web browser

→ Ex:- HTML, CSS, JavaScript (Primarily)

Advantages of Javascript :-

1. Javascript are easy to learn.
2. It is comparatively fast for user to execute.
3. Javascript support for creating large class of applications such as calculator, calendar, detecting number of visitor for the web site creating cookies, validating data and so on -
4. Java Script does not require compilation.

Keywords in Javascript :

1. break.	return	default	do	while
2. continue	throw	function	finally	instance of
3. delete	var	switch	if	
4. for	with	try	new	
5. in	case	void	this	
	else	catch	typeof	

Javascript History :-

→ Js orginal name is Livescript.

→ It is completly developed based on 'c' language syntax.

→ Livescript developed on Netscape corporation in 1990's

→ In 1995 Brendan Eich is a popular scientist in Netscape corporation. he renamed "Livescript" to "Js"

→ Js official name is ECMA script

↓

European Computer Manufacturer's Association

Functions.

→ Separate functions can be created for each separate basic. This helps in finding the bug from the program efficiently.

→ We can define the functions anywhere in the script either in head & body section or in both

How to create a function in javascript

1. Use the keyword function followed by the name of the function
2. After the function name, open & close parenthesis.
3. After parenthesis, open & close curly braces
4. within curly braces, write your line of code

Syntax:

```
function functionname()
{
line of code to be executed
}
```

Example:

```
<html>
<head>
<title> functions </title>
<script type="text/javascript">
function myFunction()
{
document.write("This is a simple function program.<br />");
}
myfunction();
</script>
</head> <body>
</body>
</html>
```

(or)

```
</script>
</head>
<body>
<center>
<script type="text/javascript">
document.write("This statement is before a function call");
document.write("<br>");
my-fun();
</script> </center> </body> </html>
```

Function with Arguments:- you can create functions with arguments as well. Arguments should be specified within Parenthesis.

Syntax:- function functionname(arg1, arg2)
{
lines of code to be executed
}

Example:-

```
<html>
<head>
<script type="text/javascript">
var count=0;
function countVowels(name)
{
for(var i=0; i<name.length; i++)
{
if(name[i]=="a"||name[i]=="e"||name[i]=="i"||name[i]
=="o"||name[i]=="u")
count = count+1;
}
document.write("Hello" +name+ "!!! your name has "+count
+ "vowels.");
}
var myName= prompt("please enter your name");
countVowels(myName);
</script>
</head>
<body>
</body>
</html>
```

Example:-

```
<html>
<body>
<script type = "text/javascript">
function my_func (var a, var b)
  {
    var c;
    c=a+b;
    return c;
  }
    my_func(10, 20);
document.write ("Result is +c");
</script>
</body>
</html>
```

Javascript Return a value:- you can also create JS function that return value. Inside the function, you need to use the keyword return followed by the value to be returned

Syntax:

```
function functionname(arg1, arg2)
  {
  line of code to be executed
    return val1;
      }
```

Example:-

```
<html>
  <head>
<script type="text/javascript">
function returnSum(first, second)
  {
var sum = first+second;
  var sum;
  }
```

```
var firstNo = 70;
var SecondNo = 30;
document.write("firstNo = " + " + secondNo + " = " + returnSum
                                                (firstNo, secondNo));
```

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

Event handlers (onclick, onsubmit):

when the page loads, it is called an event. when the user clicks a button, that click too is an event.

Onclick Event Type: This is the most frequently used event type which occurs when a user clicks the left button of this mouse you can put your validation, warning etc.

Example:

```
<html>
  <head>
<script type="text/javascript">
function sayHello()
 {
 alert("Hello world");
 }
</script>
</head>
<body>
<p> click the following button & see the result </p>
 <form>
<input type="button" onclick="sayHello()" value="SayHello" />
    </form>
    </body>
    </html>
```

events

- onchange
- onclick ✓
- onkeydown
- onload
- reset
- Select
- onSubmit ✓
- onmouseover ✓
- onmouseout ✓

onSubmit EventType :- onsubmit is an event that occurs when you try to submit a form :

The following example shows how to use onsubmit. Here we are calling a validate() function before submitting a form data to the webServer. If validate() function returns true, the form will be submitted, otherwise it will not submit the data

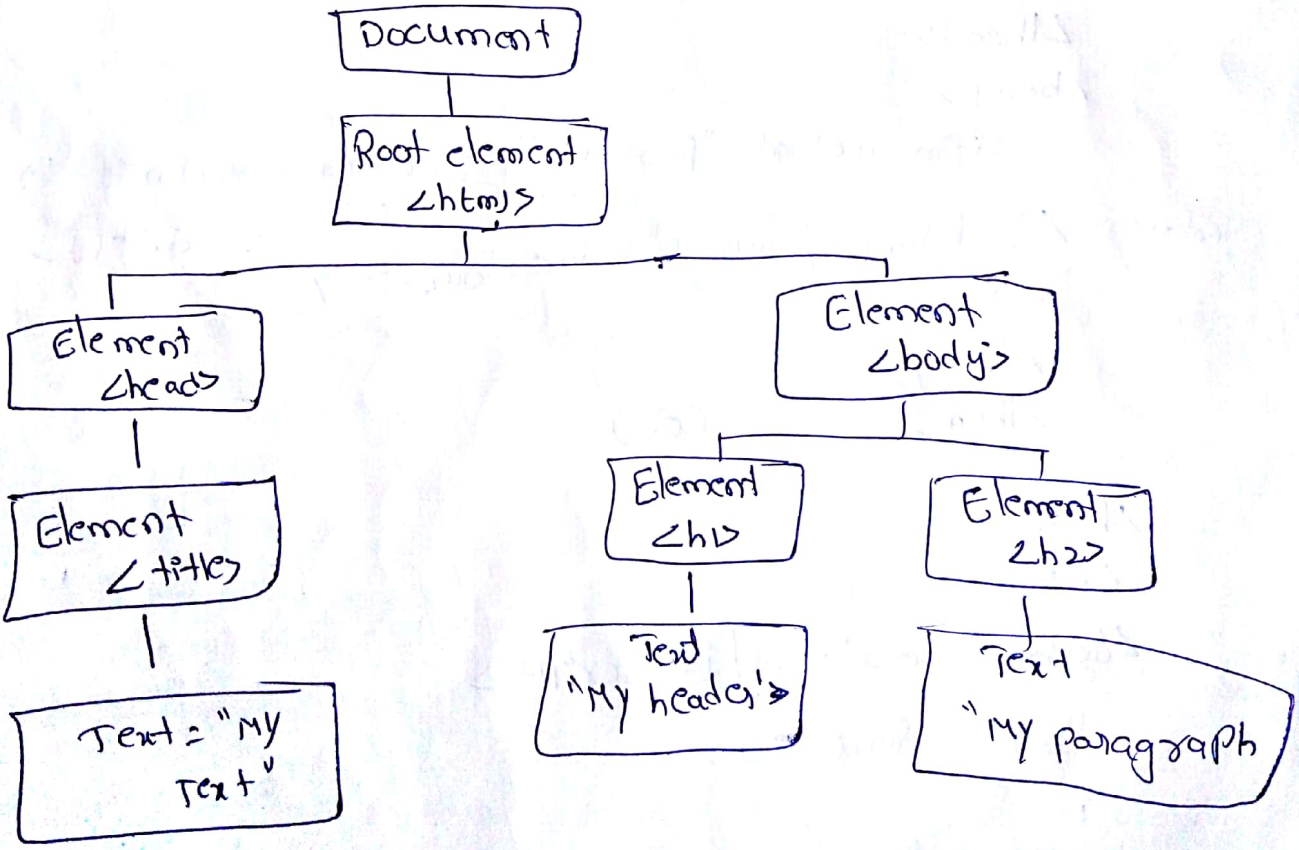
```
<html>
<head>
<script type="text/javascript">
<!--
  function validation()
  {
    all validation goes here... return either True or false
  }
//-->
</script>
</head>
<body>
  <form method="post" action="t.cgi" onsubmit="return validate()">
Enter name: <input type="submit" value="Submit" />
  </form>
</body>
</html>
```

(or)

```
<html>
<head>
<script type="text/javascript">
  function fun()
  {
alert("you have entered' + form1.text1.value);
  }
</script>
</head>
```

```
<body>
<center>
<h3>Number guessing game</h3>
</center>
<form name="form1" onsubmit=fun();>
<input type="Text" name="text1">
<input type="Submit" value="Submit">
</form>
</body>
</html>
```

Document object Model :- JavaScript can access all the elements in a webpage making use of DOM. In fact the web browser creates a DOM of the webpage when the page is loaded. The DOM Model is created as a tree of objects.



How to use DOM & Events: Using DOM, Javascript can perform multiple tasks. It can create new e's & attribute. change the existing e's & attribute & even remove existing e's & attribute. JS can also react to existing events & create a new events

1. getElementById: To access e's & attribute whose Id is set.
2. innerHTML: To access the content of an element.

```
<html>
<head>
<title> DOM </title>
<head>
<body>
<h1 id="one"> welcome</h1>
<p> This is the welcome message.</p>
<h2> Technology</h2>
<p> This is the technology section.</p>
<script type = "text/javascript">
    var text = document.getElementById("one").innerHTML;
    alert("The first heading is " +text);
</script>
</body>
</html>
```

getElementsByTagName Example: This method will return an array of all the items with the same tag Name.

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

```
<h1> welcome </h1>
<p> This is the welcome msg. </p>
<h2> Technology </h2>
<p id = "second"> This is the Technology section. </p>
<script type = "text/javascript">
    var paragraphs = document.getElementsByTagName("p");
    alert ("Content in the second paragraph is " + paragraphs[1].
                                                    innerHTML);
    document.getElementById("second").innerHTML = "The original
                                                    msg is changed.";
</script>
</body>
</html>
```

Event handler example.

1. CreateElement: To create New e.
2. Removechild: Remove an e.
3. you can add an event handler to particular e like this

```
document.getElementById(id).onclick = function()
{
    lines of code to be executed
}
```

(or)

```
document.getElementById(id).addEventListner("click",
                                    functionname)
```

Ex:-

```
<html>
<head>
<title> DOM </title>
</head>
```

```
<body>
<input type="button" id="btnclick" value="click Me!!" />
<script type="text/javascript">
  document.getElementById("btnclick").addEventListner(
                                    "click", clicked);
  function clicked()
  {
   alert("you click me!!!");
  }
</script>
</body>
</html>
```

getElementById:- The document.getElementById() method returns the element of specified id. document.getElementById() receive input value on the basis on i/p field id

Eg.

```
<html>
<body>
<head>
<script type="text/javascript">
  function square()
  {
var num = document.getElementById("number").value;
 alert(num*num);
  }
</script>
</head>
<form>
Enter NO: <input type="text" id="number" name="number" /><br />
<input type="button" value="square" onclick="square()" />
</form>
</body>
</html>
```

o/p: Enter NO: 4
square 16 } another
o/p will be displayed another page like 16

```
<html>
<head>
<body>
<script type="text/javascript">
function square()
{
var num=document.getElementById("number").value;
var result=num*num;
document.getElementById("answer").value=result;
}
</script>
</head>
<form>
Enter No: <input type="text" id="number" name="number" />
<input type="button" value="Square" onclick="square()" />
<input id="answer" />
</form>
</body>
</html>
```

O/P: Enter No: 4 — o/p will be displayed directly
Square : 16

1. DOM is an application Programming Interface for HTML & XML doc
2. with the DOM. Programmer can create & build documents, navigate their structure, and add, modify, & delete e's and
3. The DOM is an object-oriented representation of the web Page, which can be modified with a scripting lang such as JS
4. It can be used any Programming language

Event Handlers

onclick : Use this to invoke javascript upon clicking (a link, or form boxes)

onload : Use this to invoke javascript after the page or an image has finished loading. (Page loads on browser)

onmouseover : Use this to invoke javascript if the mouse pass by some link

onmouseout : Use this to invoke javascript if the mouse goes pass some link

onunload : javascript right after someone leaves this page

onclick Syntax:-

```
<input type = "button" name = 'test' value = "click me"
                                    onclick = ' inform()">
```

onload:

```
<onload = 'inform()'> - body
```

onunload - Body

onmouseover - Link, Button

onmouseout - " "

onSubmit - form

onclick - Button, checkbox, Submit

Password validation: Password.

```
<html>
<head>
<script>
function pass_validation()
{
var password = document.myform.password.value;
if (password == null || password == "")
 {
alert("Password can't be blank");
    return false;
    }
    else if (password.length < 8)
     {
    alert("Password must be at least 8 characters long!");
      return false;
     }}
    </script>
     </head>
     <body>
<form name = "myform" method = "post" onsubmit = "return
                    pass_validation()">
Password : <input type = "password" name = "password">
 <input type = "submit" value = "Submit">
   </form>
    </body>
     </html>
```

Javascript forms Validation:-

The Javascript provides the facility to validate the form on the client side so processing will be fast than server-side validation. So most of the web developers prefer client side from validation using JS.

Eg:-

```
<html>
<head>
<script>
function form_validation()
{
var name=document.myform.name.value;
// var x=document.forms["myform"]["name"].value;
if(name==null || name=="")   (logical or)
{
alert("Name can't be blank");
return false;
}
}
</script>
</head>
<body>
<form name="myform" method="post" action="register.php"
onsubmit="return form_validation()">
Name: <input type="text" name="name">
<input type="Submit" value="Submit">
</form>
</body>
</html>
```

O/P

Name:

Submit

Simple AJAX application:- AJAX stands for Asynchronous JS & XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS & JS.

→ AJAX is a web browser technology independent of web server s/w.

→ A user can continue to use the application while the client program requests information from the server in the background

AJAX is Based on Open Standards

1. Browser-based Presentation using HTML & CSS.
2. Data is stored in XML format & fetched from the server.
3. Behind-the-scene da

→ AJAX can't work independently. It is used in combination with other technologies to create interactive webpages.

Working of AJAX:- when user makes a request, the browser creates a object for the HttpRequest and a request is made to the server over an internet. The server processes this request and sends the required data to the browser. At the browser side the returned data is processed using Javascript and the web documents gets updated accordingly.

Web browser		Server
1. User Makes a request.		
2. XML Http request object is created	Request / Internet	4. Processes the Http request.
3. Sends Http request.	Response / Internet	5. Create a response and send data to browser.
6. Processes the returned data.		
7. Update web document.		

What is Script:- script is simply weekly typed Program

→ script is lightweight programming language (or) loosely typed

→ Semicolon not required, but variable declaration is requi

Adv

1. Easy to learn
2. Minimum knowledge bcz scripts ~~are~~ weekly loosely typed
3. Easy of complexity
4. HTML & CSS - static web page. whens you are adding JS we use the dynamic web pages.

Disadv:

1. JS is reading browsers. (It is maliciory Purposes)

Types

client-side The script which is running with the browser is called as client side scripting.

Server-Side: within the web server.

1. Apache,
2. Tomcat, IIS (Internet Iif Services)

Declaring Variably: Variably are used to store Value like (name:'Shilpa') or expression (sum = a+b).

Before using a variable, you first need to declare it, you have to use the keyword Var to declare a variable like

Syntax:- Var name;

Assign a value to the variable either while declaring the variable after declaring the variable.

Var name = "Shilpa"; (or) Var name;
name = "shilpa";

Ex:-

```
<html>
<head>
<title> Variables </title>
<script type="text/javascript">
var a = 10;
var b = 20;
var c = a+b;
document.write(c);
</script>
```

Local variable:

```
<script>
function abc();
{
var x = 10; // local variable
}
</script>
```

Global variable

```
<script>
var value = 10; // global
function a()
{
alert(value);
}function b()
{
alert(value);
}
```

⇒ Global variable through window object

```
function m()
{
window.value = 200; // declaring global variable by window object
}function n()
{
alert(window.value); //accessing global variable from other function
}
```

Way of using javascript

→ Between the <body> tag of html – Inline JS

→ Between the <head> tag of html – Internel JS

→ In .js file – External JS

Inline JS :- when js was written within the html e using attributes related to e's of the e then it is called

```
<html>
<form>
<input type="button" value="click" onclick="alert(Buttonclick")/>
</form>
</html>
```

o/p. [click]

Button clicked

Internal JS.

when the js was written within the section using e then it is called as internal js

```
<html>
<head>
<script>
function msg()
{
alert("welcome in js");
}
</script>
</head>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</html>
```

o/p

[click]

welcome in js

External JS.

```
<script type="text/js" src="message.js">
<form>
<input type="button" value="click" onclick="msg()"/>
```