Exp:-

```
<jsp:plugin type = "applet" code = "MyApplet.class"
 codebase = "/" >
    <jsp:params>
        <jsp:param name = "myparam"
         value = "122" />
    </jsp:params>
<jsp:fallback><b> unable to load applet </b>
    </jsp:fallback>
</jsp:plugin>
```

JavaScript

-> JavaScript is an object-based language

-> JavaScript is an example of a scripting language that is embeded in HTML document.

-> Javascript is a light weight and cross-platform.

-> Javascript is not compiled but translated. Javascript translator is responsible to translate javascript code.

Uses of Javascript:-

-> Javascript is used to create interactive websites. It mainly used for
   -> client side validation
   -> dynamic dropdown menus

-> Displaying date and time

-> Displaying group windows and dialog boxes (alert dialog box, prompt dialog box, confirm dialog box etc)

-> Displaying clocks etc.

Exp:
```
<h2> Welcome to JavaScript </h2>
<script>
  document.write("Hello World");
</script>
```

- JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag & external javascript file.

① code between body tag

```
<html>
 <head>
  <title> JS Exp </title>
 </head>
 <body>
    <script type = "text/javascript">
    alert("Hello World");
   </script>
 </body>
</html>
```

Note: The document.write() function is used to display dynamic content through javascript.

(2) Code between head tag

```
<html>
 <head>
 <script type = "text/javascript">
     function msg()
     { alert ("Hi Ram");
     }
 </script>
</head>
<body>
  <P> welcome to javaScript </P>
  <form>
  <input type = "button" value = "click"
     onclick = "msg()"/>
  </form>
  </body>
</html>
```

To call function, you need to work on event. Here we are using onclick event to call msg() function

(3) External JavaScript:

-> We can create external javascript file and embeded it in many html pages.

-> It provides code re-usability because single javascript file can be used in several html pages

-> An external javaScript file must be saved by .js extension. It is recommended to embed

all JavaScript files into a single file. It increases the speed of the webpage

→ src attribute is used to include Javascript file into html page in script tag.

Exp:

```
function hello()
{
    alert("Hello Ram");
}
```

hello.js

index.html

```
<html>
  <head>
  <script type="text/javascript" src="hello.js"
    >
  </script>
  </head>
 <body>
    <p> Welcome to JavaScript </p>
    <form>
     <input type="button" value="click"
      onclick="msg()" />
    </form>
 </body>
</html>
```

→ Note: hello.js Javascript file included into index.html page. It calls the Javascript function on button click.

JavaScript Basics.

(i) JS Comment: JS comments are as equal as Java comments.

(ii) JS variable:

A JavaScript variable is simply name of storage location. There are two types of variables in JS: (i) local variables (ii) Global variables.

Rules to declare JS variables:

(i) Name must starts with a letter (a to z or A to Z), underscore (_), or dollar ($) sign.

(ii) After first letter we can use digits (0 to 9)

Exp: value1.

(iii) JS variables are case sensitive.

Exp:
```
<script>
    var x = 10;
    var y = 10;
    var z = x + y;
    document.write(z);
</script>
```

JS Global variables:

→ A JavaScript global variable is declared outside the function or declared with window object. It can be accessed from any function.

Exp: <script>

```
var value = 50; // global variable
function a()
{
  alert (value);
}
function b()
{ alert(value);
}
</script>
```

Note: To declare JavaScript global variable inside function, you need to use window object

Exp:

```
function m()
{
  window.value = 100; /*declaring global
    variable by window object */
}
function n()
{
  alert(window.value); /*accessing global
  variable from other function */
}
```

Note: When you declare a variable outside the function, it is added in the window object internally, you can access it through window object also.

Exp:
```
<script>
    var value = 90;
    function a()
    {
        alert(window.value);
    }
</script>
```
// accessing global variable

JavaScript DataTypes:

→ JavaScript provides different Data Types to hold different types of values. There are two types of data types in JavaScript

(i) Primitive data type

(ii) Non-primitive (reference) data type.

→ JavaScript is a dynamic type language, means you don't need to specify the variable because it is dynamically used by JavaScript engine. You need to use var here to specify the data type. it can hold any type of value such as numbers, strings etc.

Exp: var a = 40; // holding number

var b = "Ram"; // holding string.

JavaScript primitive data types:

→ There are five types of primitive data types in JavaScript.

(i) String: Represents sequence of characters.
Exp: "hello"

(ii) Number: Represents numeric value ex: 10

(iii) Boolean: Represents boolean value either true or false

(iv) undefined: represents undefined value.
exp: var a;

(v) Null: represents null i.e no value at all.

JavaScript non-primitive data types:

→ There are following non-primitive data types are as follows:

(i) Object: represents instances through which we can access members.

(ii) Array: represents group of similar values.

(iii) RegExp: represents regular expression.

JavaScript Operators:

→ Javascript operators are symbols that are used to perform operations on operands.

→ There are following types of operators in Javascript.

(i) Arithmetic operators

(ii) Comparision operators

(iii) Bitwise operators

(iv) Logical operators

(v) Assignment operators

(vi) Special operators.

List of arithmetic operator.

operator	description	exp
+	addition	10 + 20 = 30
-	substraction	20 - 10 = 10
*	Multiplication	10 x 20 = 200
/	division	10/2 = 5
%	modules (remainder)	20% 10 = 0
++	increment	var a = 10; a++; now a = 11
--	decrement	var a = 10; a--; now a = 9.

List of comparision operators:

Operator	description	Exp.
==	is equal to	10 == 20 = false
===	Identical (equals and same type)	10 === 20 = false
!=	not equal to	10 != 20 = true
!==	not identical	20 !== 20 = false
>	greaterthan	20 > 10 = true
<	less than	20 < 10 = false.
<=	less than or equal to	20 <= 10 = false

List of JavaScript Bitwise operators:

(i) & -> Bitwise AND - (10==20 & 20==30) = false

(ii) | -> Bitwise OR (10==20 | 20==30) = false

(iii) ^ - Bitwise XOR (10==20 ^ 20==33) = false

(iv) ~ - Bitwise NOT (~10) = -10

(v) << - Bitwise leftshift (10<<2) = 40

(vi) >> - Bitwise right shift (10>>2) = 2

(vii) >>> - Bitwise right shift with zero (10>>>2) = 2

List of JavaScript logical operators:

(i) && -> Logical AND - (10==20 && 20==33) = false

(ii) || -> logical OR - (10==20 || 20==33) = false

(iii) ! -> Logical NOT - !(10==20) = true

List of Assignment Operators:

(i) = assign 10+10 = 20

(ii) += add and assign var a = 10; a+=20; now a = 30;

(iii) -= substract and assign var a=20; a-=10; now a = 10;

(iv) *= multiply and assign var a=20; a*=10; now a=200

(v) /= divide and assign var a=10; a/=2 now a=5;

(vi) %= modulus and assign var a=10; a%=2 now a = 0

List of JavaScript special operators:

(i) (?:) conditional operator returns value based on the condition it is like if else

(ii) , - comma operator allows multiple expressions to be evaluated as single statement

(iii) delete - delete ~~object~~ operator deletes a property from the object

(iv) in - in operator checks if object has the given property

(v) instanceof - checks if the object is an instance of given type

```
function c(){}
var o = new c();
o instance of c -> true
```

(vi) new - creates an intance (object)

(vii) typeof - checks the type of object

(viii) void - it discards expression's return value

(x) yeild - checks ~~this~~ what is returned in a generator by the generator's iterator.

If-else statement:

-> If-else statements in JavaScript as equal as in Java.

-> Switch statement as in javaScript also equal as in Java

Loops In JavaScript:-

-> There are four types of loops in JavaScript.

(i) for loop
(ii) while loop
(iii) do-while loop
(iv) for-in loop

Note: for, while and do while loops are as equal as in Java.

for-in loop:-

-> The for in statement iterates over the enumarable properties of an object, in original insertion order. For each distinct property, statement can be executed.

-> Loop through the properly of an object

-> The block code of inside the loop will be executed once for each property.

Note: Do not use the for/in statement to loop through arrays where index order is important. Use for loop instead of for/in

Exp:
```
<script>
var person = { fname: "Ram", lname: "Krish",
               age: 25 };
var text = ""; var x;
for (x in person)
{ text += person[x] + " "; }
```

</script>

o/p: Ram Krish 25

JavaScript functions.

-> JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

-> There are mainly two advantages of JavaScript functions.

(i) Code reusability: We can call a function several times so it save coding.

(ii) Less Coding: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript function Syntax:

```
Sig: function functionName ([arg1, arg2, ....
                              argn])
     {
        //code to execute
     }
```

① Exp:

```
<script>
    function msg() {
    alert("Hello");
    }
</script>
<form>
<input type="button" onclick="msg()" value=
```

"call function" />
</form>

② function arguments:

Ex: <script>

function getSquare(number)
{
alert (number x number);
}

</script>

<form>
<input type = "button" value = "click"
onclick = "getSquare(4)" />
</form>

③ function with return value

<script>
function getName ()
{
return "Ram";
}
</script>

<script>
document.write (getInfo());
</script>

Note: all the examples should be written in HTML ~~or~~ ~~javascript~~ inside head or body tag.

JavaScript Objects:-

→ A JavaScript object is an entity having state and behavior (property and method).

→ JavaScript is an object based language. Everything is an object in JavaScript.

→ JavaScript is template based not class based. Here, we don't create class to get the object but, we direct create objects.

There are three ways to create objects in JS

(i) By object literal

(ii) By creating instance of object directly (using new keyword)

(iii) By using an object constructor (using new keyword)

Object Literal:

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

property and value is separated by :(colon)

Exp:
```
<script>
    emp = {id:102, name:"Ram", salary:9000}
    document.write(emp.id+" "+emp.name+" "+emp.salary);
```

```
</script>
```

o/p: 102 Ram 9000

Instance of Object:

sig: var ObjectName = new object();

here new keyword is used to create object.

Exp:

```
<script>
    var emp = new Object();
    emp.id = 102;
    emp.name = "Ram";
    emp.salary = 9000;
    document.write(emp.id+" "+emp.name);
</script>
```

o/p: 102 Ram

Object Constructor:

-> You need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

-> The this keyword refers to the current object.

Exp:

```
<script>
    function emp(id, name, salary)
    {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }
    e = new emp(102, "Ram", 9000);
    document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

Defining method in JavaScript Object.

-> We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

Exp:

```
<script>
    function emp (id, name, salary)
    {
        this.id = id;
        this.name = name;
        this.salary = salary;
        this.changeSalary = changeSalary;
        function changeSalary(otherSalary)
        {
            this.salary = otherSalary;
        }
    }
    e = new emp (102, "Ram", 9000);
    document.write(e.id + "  " + e.name + "  " + e.salary);
    e.changeSalary(12000);
    document.write(e.id + "  " + e.name + "  " + e.salary);
</script>
```

O/P: 102 Ram 9000 102 Ram 12000

JavaScript Array:

-> JavaScript array is an object that represents a collection of (similar type of) elements.

Note: You can add elements of same type or different types.

-> There are three ways to construct array in JS

(i) array literal

sig: var arrayName = [value1, value2, ..., valueN];

(ii) Instance of array using new keyword.

sig: var arrayName = new Array( );

arrayName[0] = value;
arrayName[1] = value;
arrayName[N] = value

(iii) JavaScript array using constructor:

-> You need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

sig: Var arrayName = new Array(arg1, arg2, ... argn);

Exp:

```
<script>
    var arr1 = ["Hi", "Hello"]; //array (literal
    Var arr2 = new Array(); //using Instance
    arr2[0] = "Hi";
```

```
arr2[1] = "hello"; // using

var arr3 = new Array("Hi", "Hello");
// using constructor.
/* All three arrays (arr1, arr2, & arr3)
are store values as Hi, Hello in index
0 and 1 */

/* To access arrays in JS for loops
is useful. Array index starts with 0 and ends
with n-1 (: where n is length of array)
arrayName.length is return size of array */

for(i=0; i<arr1.length; i++)
{
    document.write(arr1[i] + "<br>");
}
</script>
```

o/p Hi
Hello

JavaScript strings:

-> JavaScript String is an object that represents a sequence of characters.

-> There are two ways to create strings in JavaScript.

(i) By String literal
(ii) By String object (using new keyword).

-> String literals is created using double quotes.

sig: var stringName = "stringValue";

-> string object: (using new keyword)

-> sig: var stringName = new String("stringvalue");

Exp:-

```
<script>
    var a = "hello";
    var b = new String("hi");
    document.write(a + " " + b);
</script>
```

JavaScript String methods:

-> List of JavaScript String methods

(i) charAt(index)
(ii) concat(str)
(iii) indexOf(str)
(iv) lastIndexOf(str)
(v) toLowerCase()
(vi) toUpperCase()
(vii) slice(beginIndex, endIndex)
(viii) trim

1) charAt(index):

this method returns the charater at the given index.

```
var str = "javascript";
str.charAt(2); // it returns v
```

(2) Concat (str):

→ The JS String concat(str) method concatenates or joins two strings.

Exp: var s1 = "Hi";
var s2 = "hello";
var s3 = s1.concat(s2);
// s3 holds a value "Hi hello"

(3) IndexOf (str)

→ This method returns index position of given string

Exp: var s1 = "I am an Indian";
var n = s1.indexOf("am");
// n holds value 2

(4) lastIndexOf (str)

→ This method returns last index position of given string.

Ex: var s1 = "I found its useful but its not for all";
var n = s1.lastIndexOf("its");
// n holds value of 23

(5) toLowerCase():

→ this method returns the given string in lowercase letters.

var s1 = "I am"
Var s2 = s1.toLowerCase()
// s2 holds a value "i am"

(6) toUpperCase():

-> This method returns the given string in uppercase letters.

```
Exp: var s1 = "hello";
     var s2 = toUpperCase(s1);
     // s2 holds a value "HELLO";
```

(7) slice(beginIndex, endIndex):

-> This method returns the part of string from given beginIndex to endIndex in slice() method, beginIndex is inclusive and endIndex is exclusive.

```
Exp: var s1 = "abcdefg";
     var s2 = slice(2,5);
     //s2 holds "cde" value
```

(8) trim()

-> This method removes leading and trailing whitespaces from the string.

```
Exp: var s1 = "  hello hi  ";
     var s2 = s1.trim();
     //s2 holds "hello hi" value
```

JavaScript Date Object:

-> The JavaScript date object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

→ You can use four(4) varient of date constructor to create date object.

Date()
Date(milliseconds)
Date(dateString)
Date(Year, month, day, hours, minutes, seconds, milliseconds)

Important methods of date object are:

(i) getFullYear() :- It returns the year in 4 digits. ex: 2017, it's a new method and suggested than getYear() which is now deprecated.

(ii) getMonth() : It returns the month in 2 digits from 0 to 11. So its better to use getMonth()+1 in your code.

(iii) getDate(): It returns the date in 1 or 2 digit form 1 to 31.

(iv) getDay(): It returns the day of week in 1 digit from 0 to 6 (0 = Sunday, 6 = Saturday)

(v) getHours(): It returns the hour of current time

(vi) getMinutes(): It returns the minute of current time

(vii) getSeconds(): It return the seconds of current time

(viii) getMilliseconds(): It returns the Milliseconds of current time

Exp: Current Date and Time :

```
<script>
    var today = new Date();
    document.getElementById('txt').innerHTML = today;
</script>
```

o/p: Current Date and time: Thu Aug 31 2017 09:54:09 GMT +0530 (India standard Time)

Another Exp:

```
var date = new Date();
var day = date.getDate(); //31
var month = date.getMonth(); //10
var year = date.getFullYear(); //2017
Var h = date.getHours(); //9
var m = date.getMinutes(); //55
var s = date.getSeconds(); //42
```

JavaScript Math Object:

→ The JavaScript math object provides several constants and methods to perform mathematical operations. Unlike date object, it does n't have constructors.

Math functions:

(i) Math.sqrt(n): It returns the square root of given number.

Exp: Math.sqrt(9) => 3.

(ii) Math.random() : this function returns the random number between 0 to 1

Exp: var a = Math.random();// 0.41840

(iii) Math.pow(m,n) : this function return the m^n

Exp: var a = Math.pow(2,4); //16

(iv) Math.floor(n) : this method returns the lowest integer for given number

Exp: var b = Math.floor(3.7); //3

(v) Math.ceil(n) : this method returns the largest integer for give number.

Exp: var c = Math.ceil(5.6); // 6

(vi) Math.round(n) : this method returns the rounded integer nearest for the given number.

If fractional part is equal or greater than 0.5, it goes to upper value 1 otherwise lower value 0.

Exp: var c = Math.round(3.7); //4

var d = Math.round(3.2); //3

(vii) Math.abs(n) : this method returns the absolute value for the given number.

Exp: var c = Math.abs(-4); //4

JavaScript Number:

-> The JavaScript number object enables you to represent a numeric value. It may be integer or floating point. A JavaScript number object follows IEEE standard to represent the floating-point numbers

Number Constructor:

Sig: var n = new Number(value);

Exp: var n = new Number(5); // n=5

-> if value can't be converted to number, it returns NaN(Not a Number) that can be checked by isNaN() method.

-> you can directly assign a number to variable.

```
var x = 12; // integer value 12
var y = 10.2; // floating point value is 10.2
var z = 13e4; // exponent value is 130000
var a = new Number(16); // integer value by
                        //number object value is 16
```

JS Number Constants:

Constant	description
MIN_VALUE	returns the largest minimum value
MAX_VALUE	return the largest maximum value
POSITIVE_INFINITY	returns positive infinity, overflow value
NEGATIVE_INFINITY	return negative infinity, overflow value
NaN	represents "Not a Number" value

JavaScript number methods:

(i) toExponential(x) : display exponential value

(ii) toFixed(x) : limits the numbers of digits after decimal value

(iii) toPrecision(x) : formats the number with given number of digits

(iv) toString() : converts number into string.

(v) valueOf() : converts other type of value into number.

Exp:

```
var a = 77.1234;
a.toExponential(2); // 7.71e+1
a.toFixed(); // returns 77 note rounded no fractional part
a.toFixed(2); // returns 77.12
a.toPrecision(2); // returns 77
a.toPrecision(4); // returns 77.12
var a.toString(); // displays '77.1234'
```

JavaScript BOM:- (Browser Object Model)

-> The Browser Object Model (BOM) is used to interact with the browser.

-> The default object of browser is window. i.e. you can call all the functions of window by specifying window or directly.

exp: window.alert("Hello") is same as alert("Hello");

→ You can use lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth etc.

Note: The document object represents an html document. It forms DOM (Document Object Model)

Methods of window object:-

(i) alert(); → displays the alert box containing message with Ok button

exp: alert("hello") or window.alert("hello");

(ii) confirm(); → displays the confirm box containing message with Ok and cancel buttons.

(iii) prompt(); → display dialog box to get input from the user.

(iv) open(); → opens the new window

(v) close(); → closes the current window

(vi) setTimeOut(); → performs action after specified time like calling function, evaluating expression etc.

Exp: <script>

function msg

(a) var v = confirm("Are you sure?")

/* v holds true if user press ok and false if user press cancel button"*/

(b) var a = prompt("who are you");

/* a holds value which is entered by user"*/,

open("http://www.google.co.in");

// it opens given url in new window

setTimeOut(alert("After 3 sec"), 3000);

// alert message will come after 3 sec.

JavaScript History Object:

-> The Javascript history object represents an array of URLs visited by the user.

-> By using this object you can load previous, forward or any particular page.

-> History object is window property so it can be accessed by window.history or history.

Property of JavaScript History Object

-> There ~~are~~ is a only one property of history object.

length : It return the length of the history URLs.

Exp: history.length;

Methods of JavaScript history object:

(i) forward() -> loads the next page.

Exp: history.forward();

(ii) back() -> loads the previous page

Exp: history.back();

(iii) go() -> loads the given page number.

Exp: history.go(2); // for next 2nd page

history.go(-2); // for previous 2nd page.

JavaScript Navigator Object:

-> The JavaScript navigator object is used for browser detection. It can be used to get browser information such as appName, appCode-Name, userAgent etc.

-> Navigator object is window property so it can be accessed by window.navigator or navigator.

Properties of JS Navigator Object:

→ There are many properties of navigator object that returns information of the browser.

(i) appName → It returns the name

(ii) appVersion → It returns the version

(iii) appCodeName → It returns the code name.

(iv) cookieEnabled → returns true if cookie enabled otherwise false.

(v) userAgent → returns the useragent

(vi) language → returns the language. It's supported in NetScape and firefox only

(vii) userLanguage → returns the user language. it is supported in IE only

(viii) plugins → It returns the plugins. It's supported in Netscape and firefox only

(ix) SystemLanguage → return the system language It is supported in IE only.

(x) mimeTypes[] → returns the array of mime type. Supported in Netscape and firefox only.

(xi) platform → returns the platform exp: win32

(xii) online → returns true if browser is online otherwise false.

Exp: document.writeln(navigator.appCodeName);
O/P: Mozilla

Navigator methods in JavaScripts:

(i) javaEnabled() -> checks if java is enabled
(ii) taintEnabled() -> checks if taint is enabled. It is deprecated since JavaScript 1.2

-> both the functions returns boolean value.
-> taint is used to check whether data validations are enable or not

JavaScript Screen Object:-

-> The JavaScript screen object holds information of browser screen. It can be used to display screen width, width, height etc.
-> Screen object is the window property so it can be accessed by window.screen or screen.

Properties of JavaScript Screen Object:-

(i) width -> returns the width of the screen
(ii) height -> returns the height of the screen
(iii) availWidth -> returns the available width
(iv) availHeight -> returns the available height
(v) colorDepth -> returns the color depth
(vi) pixelDepth -> returns the pixel depth.

Exp: document.writeln(screen.width);
O/P: 1366

JS Document Object Model: (DOM).

-> The Document Object Model represents the whole html document.

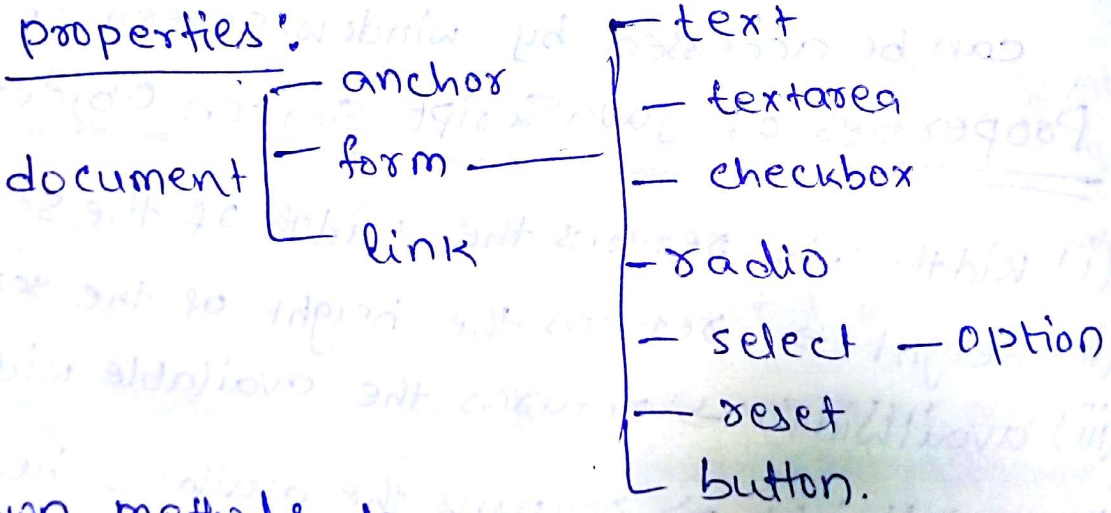
-> When html document is loaded in the browser, it becomes a document object. It is the root element that represents the html document.

-> It has properties and methods. By help of document object, we can add dynamic content to our web page.

-> Its is the object of window so we can use as window.document or document.

-> We have seen following properties and methods in DOM in previous classes.

properties:



Known methods to you.

-> getElementsById()
-> getElementsByName()
-> getElementsByTagName()
-> getElementsByClassName()

Unknown ~~Ahead~~ to you methods in JS :-

->(i) write("String") -> It writes the given string on the document

(ii) writeln("String") -> It writes the given string on the document with newline character at the end.

sig: to access value from form field.

sig: document.formName.attributeName of input text.value.

JavaScript InnerHTML:

-> The innerHTML properly can be used to write the dynamic html on the html document.

-> It is used ~~to~~ mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

Exp:

```
<html> <body> Hello
<script type="text/javascript">
~~function~~
document.body.innerHTML = "Hi";
</script>
</body>
</html>
```

-> In above example content of the body tag replaced to Hi from Hello dynamically using JavaScript innerHTML property.

JavaScript InnerText :-

-> The innerText property can be used to write the dynamic text on the HTML document.

-> Here text will not be interpreted as html text but a normal text.

-> It is used to mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.

Exp:

```
<html>
<body>
<script type="text/javascript">
function validate()
{
    var msg;
    if (document.myForm.userpass.value.length > 5)
    { msg="good"; }
    else {
        msg = "poor"; }
    document.getElementById('mylocation').
    innerText = msg;
}
</script>
<form name = "myform" >
<input type="password" value = "" name = "userpass"
```

```
onkeyup = "validate()" >
strength: <span id = "mylocation"> no strength
</span>
</form>
</body>
</html>
```

JavaScript form Validation:

-> It is important to validate the form submitted by the user because it can have inappropriate value. so validation is must.

-> The JavaScript provides you the facility to validate form ~~client~~ on the client side so processing will be fast then server-side validation so most of the web developers prefer JavaScript form validation.

-> Through JavaScript we can validate name, password, email, date, mobile number etc fields.

Exp: This example validate user input whether given input is number or not.

```
<html>
<head>
    <title> check input </title>
</head>
<body>
    <h2> Enter your registration number </h2>
```

```
<form name = "myform" action = "#">
<input type = "text" name = "text1" />
<li> *Enter numbers only</li> <br>
<input type = "submit" name = "submit"
        value = "submit"
onclick = "allnumeric(document.myform.
        text1)" />
</form>
<script src = "allnumbers.js">
</body>
</html>
```

allnumbers.js

```
function allnumeric(input)
{
    var numbers = /^[0-9]+$/;
    if(input.value.match(numbers))
    {
        alert("valid number");
        document.myform1.text1.focus();
        return true;
    }
    else
    {
        alert("invalid number");
        document.myform1.text1.focus();
        return false; } }
```

HTML/DOM Events for JavaScript

-> HTML or DOM are widely used in Javascript code.

following events are available in in Javascript.

(i) onclick : occurs when element is clicked.

(ii) ondblclick: occurs when element is double-clicked.

(iii) onfocus: occurs when an element gets focus such as button, input, textarea etc.

(iv) onblur: occurs when form looses the focus from an element.

(v) onsubmit: occurs when form is submitted.

(vi) onmouseover: occurs when mouse is moved over an element.

(vii) onmouseout: occurs when mouse is moved out from an element (after moved over)

(viii) onmousedown: occurs when mouse button pressed over an element.

(ix) onmouseup: occurs when mouse is released from an element (after mouse is pressed)

(x) onload: occurs when document, object, or frameset is loaded.

(xi) onunload: occurs when body or frameset is unloaded.

(XII) onscroll : occurs when document is scrolled.

(XIII) onresized: occurs when document is resized

(XIV) onreset: occurs when form is reset

(XV) onkeydown: occurs when key is being pressed

(XVI) onkeypress: occurs when user presses the key

(XVII) onkeyup: occurs when key is released.

PHP

Introduction:-

-> PHP is a recursive acronym for "PHP: Hypertext Preprocessor". It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embeded into HTML.

-> PHP is a server-side scripting language.

-> PHP scripts are executed on the server.

-> PHP supports many databases (MySQL, informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC etc).

-> PHP is an Open source Software.

-> PHP is free to download and use.

-> PHP runs on different platforms (Windows, Linux, etc)

-> PHP is compatible with almost all servers used today (Apache, IIS, etc)

-> PHP is easy to learn and runs efficiently on the server side.

-> PHP code is enclosed in special start and end processing instructions <?php and ?> that allow you to jump into and out of "PHP mode".

Exp:-

```
<html>
  <head>
    <title> Example </title>
  </head>
  <body>
    <?php
      echo "Hi I am a PHP script!";
    ?>
  </body>
</html>
```

-> to run this code you need to save file with .php extension and store it in /opt/lampp/htdocs folder :

htdocs folder is a root folder for php.

-> In htdocs folder you can create different folders for each and every project.

→ PHP comments are as like as Java Comments.

PHP variables::

-> All variables in PHP starts with a $ sign symbol.

→ PHP automatically converts the variable to the correct datatype, depending on its value.

-> Variable name should contain alpha-numeric characters, underscores.

→ variable name should not start with number, and variable name should not contain spaces.

Exp:

```
<?php
    $txt = "HelloWorld!";
    $x = 16;
?>
```

PHP Concatenation

-> The concatenation operator (.) is used to put two string values together.

→ To concatenate two strings variables together, use the concatenation operator.

```
<?php
    $a= "Hi";
    $b = "How";
    echo $a . $b;
?>
```

PHP Operators:

-> Operators are used to operate on values.

There are four classifications of operators.

(i) Arithmetic : +, -, x, /, %, ++, --

(ii) Assignment : =, +=, -=, x=, /=, %=,

(iii) Comparison : ==, ===, !=, <> (not equal), !==, >, <, >=, <=

(iv) Logical : and, or, xor, &&, ||, !

(v) increment/decrement

PHP Conditional Statements:

-> following conditional statements are available in PHP

(i) if

(ii) if...else

(iii) if...elseif...else

(iv) switch

Exp:
```
<?php    // example of if..else
    $a = 5;  $b = 10;
    if ($a==$b)
    {
      echo "equal";
    }
    else
      { echo "notequal"; }
?>
```

Exp ② <?php //Example of if...elseif...else

```
$a = 5; $b=10; $c=15;
if($a< $b)
{
   echo "$a. "is lessthan to ". $b;
}
else if ( $a == $c)
{
   echo $a. " and ". $b." are
        equal";
}
else
   {
      echo "you are in else stmt";
   }
?>
```

Exp 3 <?php //Example of switch

```
$a=2;
switch ($a)
{
  case 1:
     echo "one";
     break;
  case 2:
     echo "two";
     break;
  Case 3:
     echo "three";
     break.
  default:
     echo "default";
     break;
?> }
```

PHP Arrays.

-> There are three kinds of Arrays.

(i) Numeric array: an array with numeric index

-> A numeric array stores each array element with a numeric value.

-> array() function is used to create arrays in PHP.

-> Two ways to create numeric arrays.

(i) $Students = array("Ram", "Krishna");

. here Students is an array name where values Ram, & Krishna stored in a$students[0], $Students[1] respectively.

(ii) Index can be assigned manually

$cars[0] = "Volvo";

$cars[1] = "BMW";

Array Length:

-> count() function is used to return the length of an array.

Exp: echo count($Students);

Array Exp:

<?php

$x = array(1,2,3,4);

Note: Array can store any data type values.

Array exp:

```
<?php
    $a = array("Ram", "Krishna");
    $len = count($a);
    for($i=0; $i<$len; $i++)
    {
        echo $a[$i];
        echo "<br>";
    }
?>
```

(ii) PHP Associative Arrays:

-> Associative arrays are arrays that use ID_ each ID key is associated with a value

-> When storing data about specific named values, a numerical array is not always the best way to do it.

-> With associative array we can use values as keys and assign values to them.

Exp:

```
<?php
    $ages = array("peter"=>32, "Ram"=>45);
    echo $ages['peter'];
?>
```

O/P: 32

// $ages['peter'] = 32; we can create array like this also

Multidimensional Arrays:-

-> In multidimensional array, each element in the main array can also be an array.

-> And each element in the sub-array can be an array, and so on.

Exp:-
```
<?php
$stu = array("Ram" => array("B131234", "211"),
"Krishna" => array("B131235", "211"));
echo "Id of RRam is". $stu['Ram'][0];
?>
```
o/p: Id of Ram is B131234

PHP Loops:-

->There are four types of PHP loops

(i) while loop
(ii) do...while loop
(iii) for loop
(iv) foreach : loops through a block of code for each element in an array.

Exp:
```
<?php
$i = 1;
while ($i < 5)
{
    echo "While <br>";
}
$j = 0;
```

```
do
{
    $j++;
    echo "do while <br>";
}while($j<5);

for($k=0; $k<2; $k++)
{
    echo "for loop <br>";
}

/* foreach sign:  foreach ($array as $value)
                  {
                    code to be executed;
                  }
*/

$arr = array("one","two","three");
foreach ($arr as $value)
{
  echo $value."<br>";
}
?>
```

O/P:

```
while
while
while
while
dowhile
dowhile
dowhile
dowhile
dowhile
forloop
forloop
one
two
three
```

PHP Functions.

Sign:
```
function functionName()
{
    code to be executed;
}
```

Exp:
```
<?php
function writeName()
{
    echo "Ram";
}
echo "my name is ". writeName();
?>
```

Exp:2
```
<?php
function writeName($name)
{
    echo $name;
}
writeName("Ram");
?>
```

PHP data types

(i) integer : exp: $a = 5;

(ii) String : exp: $b = "hello";

(iii) floating point numbers (or) doubles:
exp: $c = 1.2;

(iv) boolean : exp: $d = true;

(v) array : exp: $arr = array(1, 2);

(vi) NULL : exp: $e = NULL;

(vii) Object:

-> An object is a data type that not only allows storing data but also information on, how to process that data.

-> An object is a specific instance of a class which serves as templates for objects. Objects are created based on this template via the new keyword.

-> Every object has properties and methods corresponding to those of its parent class. Every object instance is completely independent, with its own properties and methods, and thus can be manipulated independently of other objects of the same class.

Exp:

```
<?php
    class greeting
    {
        public $str = "Hello World";
        function show_greeting()
        {
            return $this->str;
        }
    }
    $message = new greeting();//object creation
    var_dump($message);
?> o/p:
```

(viii) PHP resources:

-> A resource is a special variable, holding reference to an external resource.

-> A resource variable typically holds special handlers to opened files and database connections.

Exp:
```
<?php
    //open a file for reading
    $handle = fopen("note.txt", "r");
    var_dump($handle);
    echo "<br>";
    //connect to MySQL server with default setting
    $link = mysql_connect("localhost", "root", "");
    var_dump($link);
?>
```

PHP strings:-

-> String exp: `$a = "hello";`

String function:-

(i) strlen(): This function is used to find length of a string.

(ii) str_word_count(): This function counts the number of words in a string.

(iii) str_replace(): It replaces all occurances of the search text within the target string.

(iv) Notes You can

(iv) strrev() : Its reverse a string.

Exp: <?php

```
$a = "hello";
echo strlen($a); //o/p: 5
$b = "how are you?";
echo str_word_count($b); //op: 3
echo str_replace("how", "who", $b);
//o/p: who are you?
/* Note: you can add fourth argument to string replace function to know how many times the string replacement was performed */
//echo str_replace("how", "who", $b, $count);
str_replace("how", "who", $b, $count);
echo "The text was replaced ".$count."times";
echo strrev($a); //o/p: olleh
?>
```

Reading data from forms using PHP:

-> To access the value of a particular form field you can use the following superglobal variables.

-> These variables are available in all scopes throughout a script.

(i) $_GET :- It contains a list of all the field names and values sent by a form using the get method (ie via the URL parameters)

(ii) $_POST : Contains a list of all the field names and values sent by a form using the post method (data will not visible in the URL)

(iii) $_REQUEST : Contains the values of both the $_GET and $_POST variables as well as the values of the $_COOKIE superglobal variable.

Content-form.php ↓

exp:-
```
<htm>
  <head>
    <title> Contact form </title>
  </head>
  <body>
    <h2> Contact us </h2>
    <P> Please fill in this form and
        send us </P>
    <form action = "process-form.php"
          method = "post" > <br>
      Name: <input type = "text"
      name = "name" id = "inputName" >
      <br> Email : <input type = "text" name =
      <br> "email" id = "inputEmail" >
      <br>Subject : <input type = "text" name
        = "Subject" id = "inputSubject" >
      <br> Message: <textarea name = "message" id = "inputmsg"
```

rows = "5" cols = 50></textarea>

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

</form>

process-form.php :-

```
<html>
  <head>
    <title> Contact form </title>
  </head>
  <body>
    <h1> Thank You </h1>
    <P> Here is the information you have
        submited : </P>
    <ol>
    <li> <em> Name : </em> <?php echo
    $_POST["name"] ?> </li>
    <li><em> Email : </em> <?php echo $_POST
    ["email"] ?> </li>
    <li> <em> Subject : </em> <?php echo
    $_POST["subject"] ?> </li>
    <li><em> Message : </em> <?php echo
    $_POST["message"] ?> </li>
  </ol>
</body>
</html>
```

PHP MySQL

-> To connect MySQL database using php we need to use mysqli_connect() function.

-> This function will take three arguments as localhost, sql username, sql password.

sig: mysqli_connect("localhost", "username", "password").

exp: mysqli_connect("localhost", "root", "");
in above example sql username is root and password is empty ("").

-> To execute query using PHP we need to use mysqli_query() function.

mysqli_query function will take two parameters one is sql connection reference and another one is sql query statement reference.

```
Exp: $link = mysqli_connect("localhost", "root", "");
     $sql = "CREATE DATABASE hello";
     mysqli_query($link, $sql);
```

Note: mysqli(improved mysql).

-> we have another way to connect mysql server using PDO(PHP data object).

-> MySQLi & PDO both offer an object oriented

API but MySQLi also offer a procedural API. which is relatively easy for beginers to understand.

① Sig: to connect mysqli procedural way:

```
$link = mysqli_connect("hostname", "username",
        "password", "database");
```

Exp: `$link = mysqli_connect("localhost", "root", "", "demo");`

② Sig: to connect mysqli object oriented way.

```
$mysqli = new mysqli("hostname",
        "username", "password", "database");
```

Exp: `$mysqli = new $mysqli("localhost", "root", "", "demo");`

③ Sig to Connect mysqli PDO (PHP data object) way:

Sig:

```
$pdo = new PDO("mysql:host=hostname;
dbname=database", "username", "password");
```

Exp:

```
$pdo = new PDO("mysql:host=localhost",
dbname=demo", "root", "");
```

Exp: how to use prepared statement in PHP MYSQLi? procedural way:-

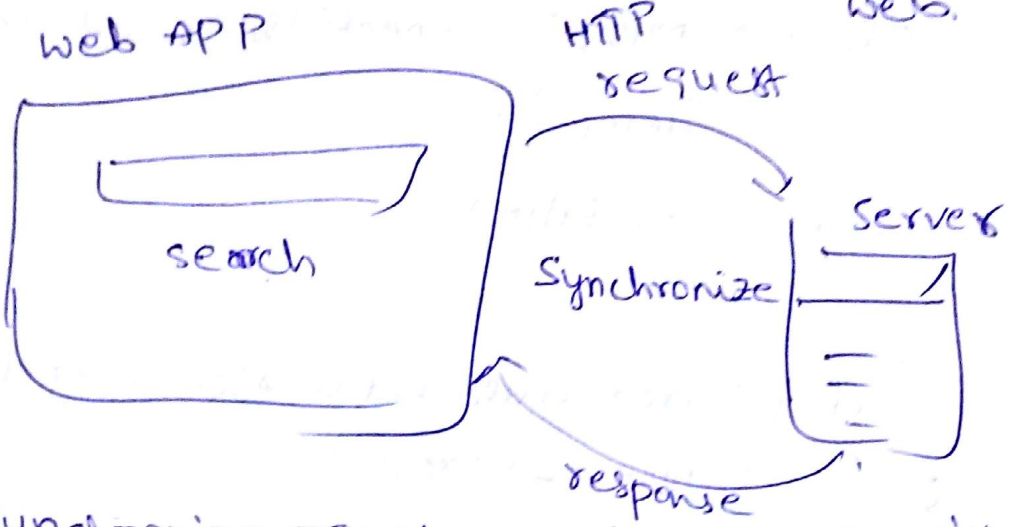
sol:

```
<?php
$link = mysqli_connect("localhost", "root", "",
        "demo");
if ($link == false)
{
    die("Error could not connect". mysqli_
        connect_error());
}
$sql = "Insert into persons( firstname, lastname,
    email ) VALUES (?, ?, ?)";
if ( $stmt = mysqli_prepare($link, $sql))
{
    mysqli_stmt_bind_param ($stmt, "isss",
        $firstname, $lastname, $email);
    $firstname = "Ram";
    $lastname = "Krishna";
    $email = $"ram@gmail.com";
    mysqli_stmt_execute ($stmt);
    echo "Records inserted successfully";
}
else
    {
        echo "could not prepare sql query :$sql".
    } mysqli_error($link);
    mysqli_stmt_close($stmt); mysqli_close($link);
?>
```

AJAX

Ajax]
Asynchronize Javascript and XML

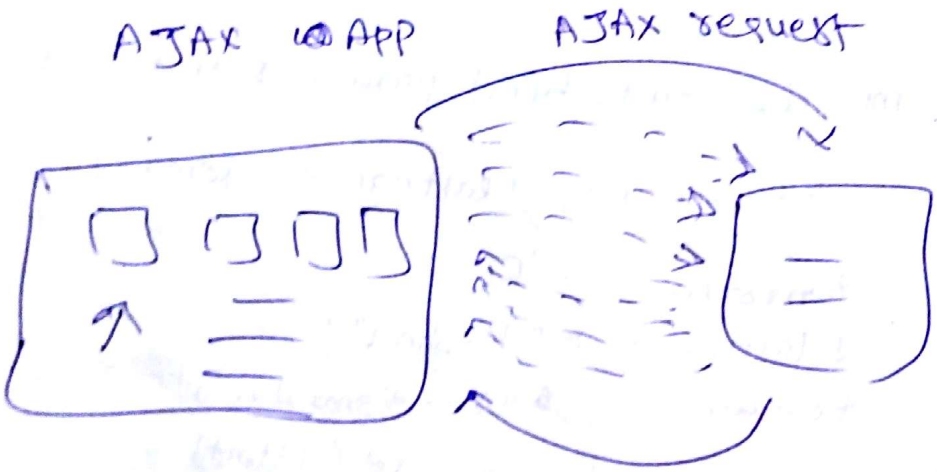
Normal web → (or) traditional web.

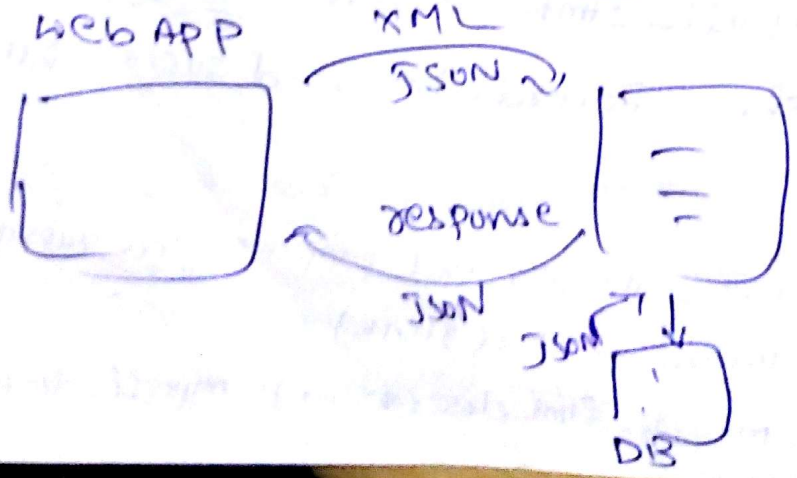


Synchronize means user has to wait till he gets response.

→ but how it works fB & twitter without reloading page

→



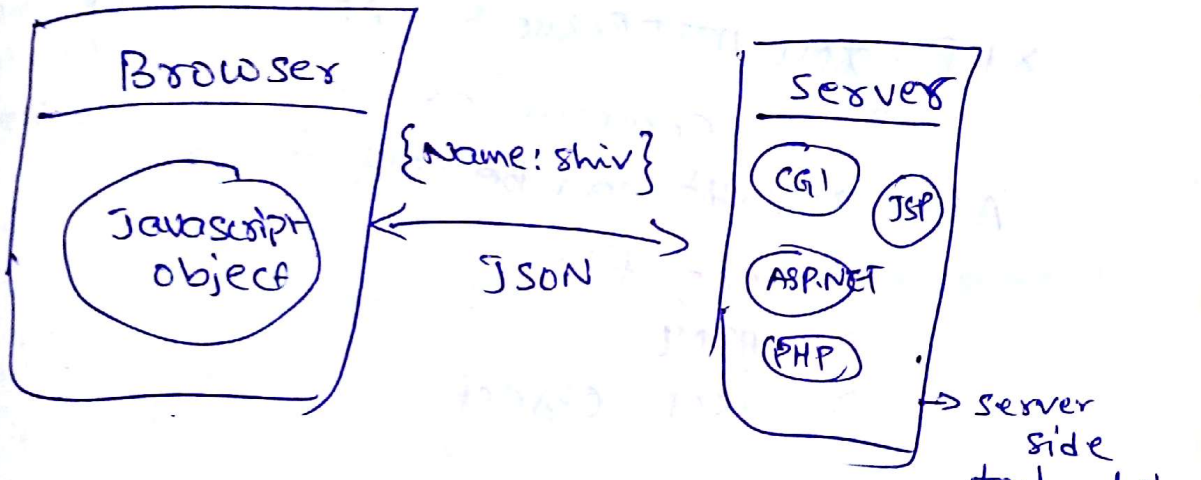


Ruby basic
JQuery basic

what is JSON:

→ JavaScript object Notation.

→ JSON is simple data exchange format which helps to communicate between javascript & server side technologies.



→ json format:

```
{
    name : value, name : value
}
```

Exp: `<script language = "javascript">`

```
{
  "CustomerCode" : "101",
  "CustomerName" : "Rani"
}
```

if you make it as javascript variable it becomes javascript object.

Exp: `var x = { "customercode" : "101", "customerName" : "Rani" }`

`alert (x.customercode)` => 101.

JQuery : Its an addon to Javascript

founder : John Resig.

Ajax is not a single technology but its a group of technologies.

XHR - XML HTTP Request (API) handle the communication between the client and server.

Ajax request can be any format.

- -> text file
- -> HTML
- -> JSON object

PHP File functions

1) fopen ($file, mode) → $include_path, $context are optional
2) fclose ($file)
3) fwrite ($file, "data", $length) → optional.
4) file_exists ($file)
5) fgets ($file)
6) fgetc ($file)
7) feof ($file)
8) copy (sourcefile, destfile)
9) unlink ($file)
10) file_get_contents ($file)

select from mysql

-> $sql = select * from student;

-> ~~$res = $conn->query($sql);~~

$res = mysqli_query($conn, $sql);

```
if ( $res ->num_rows >0)
{
    while ($row = result -> fetch_assoc())
    {
      echo  $row['id']. $row['name'];
    }
}
else
  {
      echo "0 results";
  }
}
```

files ref: https://www.guru99.com/php-file-processing.html

To read files in PHP:-

```
$file = fopen("hi.txt", "r");
if($file)
{ while ($line =fgets($file)) !== false)
    {
      echo $line;
    }
```

```
        fclose($file);
    }
else
    {
        echo "error opening a file";
    }
```

another way to read file using eof

```
$file = fopen("hi.txt", "r")
if($file)
{
    while (!feof($file))
    {
        $line = fgets($file);
        echo $line;
    }
    fclose($file);
}
else
{
    echo "error opening a file";
}
```