Java Script is an Object-based scripting language which is light weight, cross platform and interpreted language. It is designed for creating network centric applications. It is very easy to implement because it is integrated with HTML.

→ Java script is not a compiled language, but it is a translated language. The 'java script translator (embedded in the browser) is responsible for translating the 'javascript code for the web browser.

→ Java Script is one of the three languages all web developers must learn: a) HTML to define the content of web pages. b) CSS to specify the layout of webpages. c) Java script to program the behavior of web pages.

→ Applications:- Java Script is used to create interactive websites. It is mainly used for a) Client side validation b) Dynamic drop-down menus c) Displaying date and time d) Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialogbox, and prompt dialog box). e) Displaying clocks. f) User Notifications

→ Advantages:- 1) A script or scripting language is a computer language with a series of commands within a file that is capable of being executed without being compiled. 2) Open source, allowing users to view and edit the script if needed. 3) Easy to port between different operating systems. 4) Easy to learn and write.

→ Java script was first known as Live Script, but Netscape changed its name to Java script, possibly because of the excitement being generated by Java. Java script made its first appearance in Netscape 2.0 in 1995 with the name Live Script. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

→ The merits of using Java script are – less server interaction, immediate feedback to the visitors, increased interactivity and richer interfaces.

→ Types of scripting languages:- There are two types of scripting languages. a) Client side b) server side scripting.

→ a) Client Side scripting languages:- The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the users computer over the internet and run directly in the browser.

→ The scripting language needs to be enabled on the client computer. Sometimes if a user is conscious of security risks they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when script is attempting to run.

→ b) Server Side Scripting languages:- The server side environment that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive websites that interface to databases or other data stores on the server.

Introduction to Javascript:- Javascript can be implemented using Javascript statements that are placed within the <script> ... </script> HTML tags in a webpage.

→ We can place the <script> tags, containing our Javascript, anywhere within the webpage, but it is normally recommended that we should keep it within the <head> tags. The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.

Syntax:-
```
<script ... >
    Javascript code
</script>
```

→ The script tag consists of two attributes. a) Language (It specifies what scripting language you are using i.e 'javascript'). b) Type (What is to be recommended to indicate the scripting language in use and its value should be sent to "text/javascript").

Example:-
```
<script language="javascript" type="text/javascript">
    Javascript code
</script>
```

→ Simple program for Javascript:-
```
<html>
  <body>
    <script language="javascript" type="text/javascript">
      document.write ("Hello Javascript!")
    </script>
  </body>
</html>
```

→ The code will produce the following result is -

Hello Javascript!

→ JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. JavaScript allows you to omit semicolon if each of your statements are placed on a separate line. Ex:- <script>

```
var1 = 10
var2 = 20
</script>
```

→ But when formatted in a single line as follows, you must use semicolons. <script>

```
var1 = 10; var2 = 20;
</script>
```

→ It is a good programming practice to use semicolons and the JavaScript is a case-sensitive language.

→ Using comments to prevent execution of code is suitable for code testing. Comments are used to add information about the code, warnings or suggestions so that end users can easily interpret the code.

→ JavaScript single line comment is represented by //.

Ex: a) // It is a single line comment

b) var x = 10+20; // addition of 10 and 20.

→ JavaScript multiline comment is represented i.e enclosed in /* and */.

Ex:- /* It is multiline comment
It will not be displayed */

→ The document.write() function is used to display dynamic content through JavaScript.

→ Places to put JavaScript code:- 1) Between the body tag of HTML 2) Between Head tag of HTML 3) In .js file (External JavaScript).

JavaScript DataTypes:- Java script provides different data types to hold different types of values. There are two types of data types. 1) Primitive Datatype 2) Non Primitive (Reference) Datatype

→ Javascript is a dynamic type language, means you don't need to specify type of the variable because it is dynamically used by Javascript engine.

→ You need to use var here to specify the datatype. It can hold any type of values such as numbers, strings etc

Ex: 1) var a=40; // Holding number

2) var b="Ajay"; // Holding String

→ Primitive DataTypes:- There are five types of primitive datatypes in Java script. They are as follows:

a) String ⇒ Represents sequence of characters. Ex: "hello"

b) Number ⇒ Represents Numeric values. Ex:- 100

c) Boolean ⇒ Represents boolean value either true or false

d) Undefined ⇒ Represents undefined value

e) Null ⇒ Represents null i.e. no value at all.

→ Non-Primitive DataTypes:- The non-primitive datatypes are as follows:

a) Object ⇒ Represents instance through which we can access members

b) Array ⇒ Represents group of similar values

c) RegExp ⇒ Represents Regular Expression.

→ Ex: Java Script objects are written with curly braces {}. Object Properties are written as name: value pairs, seperated by commas.

var person={firstname: "John", lastname: "Paul", age: 25};

→ Array items are seperated by commas. var colors=["red", "blue"];

→ JavaScript Variable:- A javascript variable is simply a name of storage location. There are two types of variables in JavaScript, local variable and global variable

→ There are some rules while declaring a JavaScript variable (also known as Identifiers) a) Name must start with a letter (a to z (or) A to Z), underscore (_), or dollar sign. b) After first letter we can use digits (0 to 9). c) JavaScript variables are case sensitive Ex: var x=10; var _y="Hi".

→ A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

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

→ A JavaScript global variable has global scope which means it can be defined anywhere in your JavaScript code.

```
Ex. <script>
      var x=50; //Global variable
      function a()
      { alert(value);
      }
    </script>
```

→ JavaScript Reserved words:- A list of all the reserved words in JavaScript are given in the following table. They can not be used as JavaScript variables, functions, loop labels, or any object names

abstract	~~catch~~char	delete	extends	function	instanceof
boolean	class	do	false	goto	int
break	const	double	final	if	interface
byte	continue	else	finally	implements	long
case	debugger	enum	float	import	native
catch	default	export	for	in	new

null	public	super	throw	try	volatile
package	return	switch	throws	typeof	while
private	short	synchronized	transient	var	with
protected	static	this	true	void	

fig: JavaScript Reserved words.

JavaScript Operators:- JavaScript operators are symbols that are used to perform operations on operands. There are the following types of operators in JavaScript

1) Arithmetic Operators:- These are used to perform arithmetic operations on the operands. Addition (+), Subtraction (-), Multiplication (*), Division (/), Modulus (%), Increment (++), Decrement (--).

Ex:-
```
<script>
    var x=10; var y=20;
    var z=x+y;
    document.write(z);
</script>
```

2) Comparsion (or) Relational Operators:- These are used to compare the two operands. Equal (==), Equal and of same type (===), Not Equal to (!=), Not Identical (!==), Greater than (>), Greater than or equal to (>=), less than (<), less than or equal to (<=).

3) Logical operators:- Logical AND (&&), Logical OR (||), Logical NOT (!).

4) Assignment operators:- Assign (=), Add and Assign (+=) etc.

5) Bitwise operators:- Bitwise AND (&), Bitwise OR (|), Bitwise XOR (^), Bitwise NOT (~), Bitwise Left Shift (<<), Bitwise right shift (>>), Bitwise Right shift with zero (>>>).

→ 6) Special operators:-

a) Conditional operator (? :): It returns value based on the condition. Ex: var x = (10<20) ? 10 : 20;

b) Comma operator (,):- It allows multiple expressions to be evaluated as single statement.

c) Delete operator :- It deletes a property from the object

d) new operator: It creates an instance of an object

e) in operator:- It checks if object has the given property.

f) instance of operator:- It checks if the object is an instance of given type

g) typeof operator: It is a unary operator ie placed before its single operand, which can be of any type. The "typeof" operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

Ex: var x=10;
var result = (typeof x == "string" ? "x is Numeric" : "x is string");
document.write(result); // x is Numeric

h) void operator:- It discards the expression's return value

i) yield operator:- checks what is returned in a generator by the generator's iterator.

→ If you re-declare a JavaScript variable, it will not lose its value var x=10; var x; then x is 10.

→ Ex: a) var x = 5+2+3; // x value is 10

b) var y = "John" + " " + "Mary"; // y value John Mary

→ If you put a number in quotes, the rest of the numbers will be treated as strings, and concatenated

Ex: a) var x = "5"+2+3; // output is 523.

b) var y = 2+3+"5"; // output is 55

Conditional statements:- These are used to perform different actions based on different conditions. There are four types of conditional statements. 1) If statement 2) If-else statement. 3) Else-if ladder 4) Switch statement

→ 1) Simple If statement:- It evaluates the content only if expression is true

Syn:-
```
if (condition)
{
// content to be evaluated
}
```

Ex:-
```
<script>
var a=10;
if (a>10){
document.write("a>10");
}
</script>
```
O/P:- a>10

→ 2) If-else statement:- It evaluates the content whether condition is true or false.

Syn:
```
if (expression){
//Content to be evaluated if condition is true
}
else {
//Content to be evaluated if condition is false
}
```

→ 3) Else-if ladder: It evaluates the content only if expression is true from several expressions.

Ex: `if (time<10) { greetings = "Good Morning"; }`
`else if (time<20) { greetings = "Good Evening"; }`
`else { greetings = "Have a nice day"; }.`

→ 4) Switch statement:- It is used to execute one code from multiple expressions.

Ex:
```
<script> var grade='B'; var result;
switch (grade) { case 'A' : result = "A Grade"; break;
                 case 'B' : result = "B Grade"; break;
                 default : result = "No Grade"; }
document.write(result); </script>   //output is B Grade
```

→ Looping Statements:- The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

→ There are four types of loops in JavaScript. 1) for loop 2) while loop 3) do while loop 4) for-in loop.

→ 1) for loop:- It iterates the elements for the fixed number of times. It should be used if number of iterations is known.

Syn:-
```
for(initialization; condition; increment)
{
  // code to be executed
}
```

Ex:-
```
<script>
for(i=1; i<=5; i++)
{
  document.write(i + "<br/>");
}
</script>   // output 1 2 3 4 5
```

→ 2) while loop:- It iterates the elements for the infinite number of times, It should be used if number of iterations is not known.

Syn:-
```
while(condition)
{
//code to be executed
}
```

Ex:
```
<script>
var x=11;
while(x<=15){
{ document.write(x+"<br/>");
  x++;
}
</script>  //outputs 11 12 13 14 15
```

→ 3) do-while loop:- Code is to executed at least once whether the condition is true or false.

Syn:
```
do
{
//code to be executed
}while(condition);
```

Ex:-
```
<script>
var y=21;
do
{
 document.write(i+"<br/>");
 y++;
}while(y<=25);
</script> //o/p is 21 22 23 24 25
```

→ 4) for-in loop:- It is used to iterate the properties of an object

Syn:-
```
for(variablename in object)
{ // code to be executed }
```

Jump Statements:- The "break" and "continue" statements are the only JavaScript statements that can "jump out of" a code block.

1) Break statement:- It is used to "jump out" of a switch statement. It can also be used to "jump out" of a loop. The break statements breaks the loop and continues executing the code after the loop (if any).

```
Ex: for (i= 0; i<10; i++)
    {
      if (i==3) break;
      text += " The Number is=" + i + "<br>";
    }
```

o/p:
The Number is 0
The Number is 1
The Number is 2

2) continue statement:- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
Ex: for(i= 1; i<=5; i++)
    {
      if(i==3) continue;
      text += " The Number is" + i + "<br>";
    }
```

o/p:
The Number is 1
The Number is 2
The Number is 4
The Number is 5

JavaScript Labels:- The continue statement (with or without a label reference) can only be used to skip one loop iteration. The break statement, without a label reference, can only be used to jump out of a loop or a switch. With a label reference, the break statement can be used to jump out of any code block Ex: a) break labelname;
b) continue labelname;

Javascript Functions:- A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again.

→ Adv:- 1) Code reusability (we can call a function several times so it save coding). 2) Less coding (It makes our program compact)

Syntax:-
```
function <functionname>([arg1, arg2, ... argN])
{
  // code to be executed
}
```

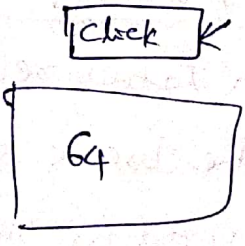
→ The most common way to define a function in Javascript is by using the "function" keyword, followed by a unique function name, a list of parameters (zero or more), and a statement block surrounded by curly braces.

→ Example: function has no arguments.

```
<script>
 function msg()
 { alert("hello! this is message");
 }
</script>
<input type="button" onclick="msg()"
        value = "Call function" />
```

→ We can call function by passing arguments.

Ex:-
```
<script>
 function getcube(number)
 {
   alert(number*number*number);
 }
</script>
<form> <input type="button" value="click"
          onclick="getcube(4)"/>
</form>
```



Function with return value:- We can call function that returns a value and we it in our program.

Ex:1)
```
<script>
function getInfo()
{
    return "hello how are you";
}
</script>
```

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

O/P:- hello how are you

Ex: 2)
```
var x = myFunction(4,3); // Function call
function myFunction(a, b) // Function Definition
{
    return (a*b); // function returns product of a and b
}                 // output is 12 i.e x=12
```

Java Script Objects:- A java script object is an entity having state and behavior (properties and methods). Ex: Pen, car, chair etc. Java script 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.

→ Creating objects in Javascript:- There are three ways to create objects. 1) By object literal 2) By creating instance of object directly (using new keyword) 3) By using an object constructor (using new keyword)

→ 1) By Object literal:- The syntax of creating object using object literal is given below:

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

Ex:-
```
<script> emp={id:102, name:"ravi", salary=20000}
    document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

→ 2) By creating instance of Object:- new keyword is used to create objects. var objectname = new Object();

Ex: <script>
var emp = new Object();
emp.id = 101; emp.name = "Ravi"; emp.sal = 50000;
document.write(emp.id + " " + emp.name + " " + emp.sal);
</script>

→ 3) By using an Object Constructor:- Here, we need to create function with arguments. Each argument value can be assigned in the current object by using "this" keyword (refers to the current object).

Ex: <script>
function emp(id, name, sal)
{
this.id = id; this.name = name; this.sal = sal;
}
e = new emp(103, "Vishal", 30000);
document.write(e.id + " " + e.name + " " + e.sal);
</script>

→ JavaScript Array:- JavaScript array is an object that represents a collection of similar type of elements. There are three ways to construct array in JavaScript. 1) By Array literal, 2) By creating instance of Array directly (using new keyword) 3) By using an Array constructor (using new keyword)

→ 1) By Array literal:- The syntax of creating array using array literal

Syntax: var arrayname = [value1, value2, ..., valueN];

Where values are contained inside [ ] and seperated by, (comma).

Ex: <script> var emp = ["Ravi", "Ramu", "Raju"];
for(i=0; i<emp.length; i++) {
document.write(emp[i] + "
"); }
</script>

2) JavaScript Array Directly:- The syntax of creating array directly.

Syn: Var arrayname = new Array();

Ex:
```
<script>
Var i; Var emp = new Array();
emp[0] = "Ravi"; emp[1] = "Raju"; emp[2] = "Ramu";
for(i=0; i<emp.length; i++)
{ document.write(emp[i] + "<br>");
}
</script>
```

3) JavaScript Array Constructor:- Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

Ex:
```
<script>
var emp = new Array("Ravi", "Raju", "Ramu");
for(i=0; i<emp.length; i++)
{
document.write(emp[i] + "<br>");
}
</script>
```

→ JavaScript Array methods:- 1) concat():- It returns a new array object that contains two or more merged arrays.

2) reverse():- It reverses the elements of given array.

3) push(): It adds one or more elements to the end of an array.

4) pop():- It removes and returns the last element of an array.

5) sort(): It returns the elements of the given array in a sorted order.

JavaScript String:- It is an object that represents a sequence of characters. There are two ways to create strings in JavaScript

1) By String literal:- It is created using double quotes.

```
<script> var str = "This is string literal";
document.write(str); </script>.
```

→ 2) By using String object:- The new keyword is used to create instance of string.

Ex: <script> var stringname = new String("hello string");
document.write(stringname);
</script>

→ String methods:- 1) charAt():- character value present at the specified index.

2) concat():- Combination of two or more strings.

3) indexOf():- Position of a char value present in the given string.

4) lastIndexOf():- position of a char value present in the given string by searching a character from the last position.

5) search():- searches a specified regular expression in a given string and returns its position if a match occurs.

6) replace():- replaces a given string with the specified replacement

7) substr():- To fetch the part of the given string on the basis of the specified starting position and length.

8) substring(): To fetch the part of the given string on the basis of the specified index

9) toLowerCase():- converts the given string into lowercase letter

10) toUpperCase():- Converts the given string into uppercase letter

11) toString():- It provides a string representing the particular object.

12) valueOf():- It provides the primitive value of string object

Ex: var s1 = "JavaScript"; var s2 = "Language";
document.write(s1.charAt(2)); // v
var s3 = s1.concat(s2); document.write(s3);
var s4 = s1.toLowerCase(); var s5 = s2.toUpperCase();
var s6 = " Java script"; var s7 = s6.trim();
var s8 = "Javascript from javatpoint index of ";
var s9 = s8.indexOf("from"); // 11. var s10 = s8.lastIndexof("java"); // 16.

Date Object:- It can be used to get year, month and day.

You can display a times on the webpage by the help of Java script date object. There are 4 possible ways to create date object

1) Date():- Ex: var d= new Date();
2) Date(milliseconds) Ex. var d=new Date(0); // Jan 1, 1970 05:30:00
3) Date(dateString). Ex: var d=new Date("October 13, 2014, 11:13:00");
4) Date(year, month, day, hours, minutes, seconds, milliseconds)
   Ex. var d=new Date(2019, 10, 1, 10, 45, 16); // 6 numbers specify year, month, day, hour, minute, second. If 5 then first 5 only etc

Date methods:-1) getDate():- Return int value between 1 and 31.

2) getDay(): Returns int value between 0 and 6.
3) getHours():- Returns int value between 0 and 23.
4) getMilliseconds(): Returns int value between 0 and 999.
5) getMinutes(): Returns int value between 0 and 59.
6) getMonth(): Returns int value between 0 and 11.
7) getSeconds(): Returns int value between 0 and 59.
8) setDate():- It sets the day value for the specified date on the basis of local time. Same for day, hours, minutes, seconds, months.
9) toDateString(): Returns the date portion of a Date object.
10) toTimeString(): Returns the time portion of a Date object.
11) toString(): It returns the date in the form of string
12) valueOf():- It returns the primitive value of a Date object.

Ex:-
```
<script> // current date and time.
    var date = new Date();
    var day = date.getDate(); var mon= date.getMonth()+1;
    var year = date.getFullYear();
    document.write("<br> Date is:"+day+"/"+mon+"/"+year);
    var today = new Date(); var h=today.getHours();
    var m=today.getMinutes(); var s=today.getSeconds();
</script> document.getElementById('tat').innerHTML = "h+":"+m+":"+s;
```

→ Math Object:- The JavaScript math object provides several constants and methods to perform mathematical operation. Unlike date object, it does not have constructors.

→ Math methods:- 1) abs() 2) acos() ⇒ arccosine of the given number in radians. 3) cbrt(): It returns cube root of given number 4) ceil(), 5) cos() 6) exp() 7) floor() 8) log() 9) max() 10) min() 11) pow() 12) random() 13) round() 14) sign() 15) sin() 16) sinh() 17) tan() 18) trunc().

→ Ex: 1) Math.min (0, 150, 30, 20, -8, 6); // returns -6.

2) Math.random(); // Random Number between 0 and 1.

3) Math.PI // returns PI value

→ Number Object:- It enables you to represent a numeric value. It may be integer or floating point.

Ex: 1) var n = new Number (value);

Ex: 2) var n = new Number (16); // Integer value by Number object

Ex: 3) var x = 102; // Integer value. You can direct assign number to a variable also.

Ex: 4) var y = 13e4; // 130000

→ Number Constants:- 1) MIN-VALUE 2) MAX_VALUE 3) NaN (represents "Not a Number" value).

→ Number Methods:- 1) isInteger() 2) parseInt() 3) parseFloat() 4) toExponential() 5) toString() 6) toPrecision().

→ Boolean Object:- It represents value in two states: true or false. The default value of JavaScript Boolean object is false.

Syn:- Boolean b = new Boolean(value);

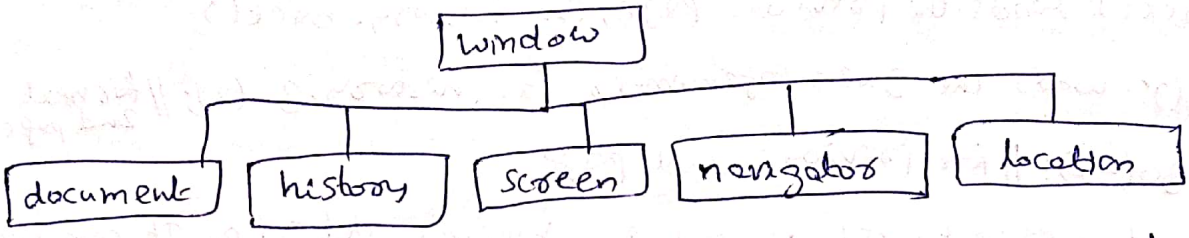
Ex: document.write (10<20); // true

Methods: 1) toSource() ⇒ Returns the source of Boolean object as a string

2) toString() ⇒ Converts Boolean into string

3) valueOf() ⇒ Converts other type into Boolean

Browser Object Model (BOM):- It is used to interact with the browser. The default object of browser is window means you can call all the functions of window by specifying window or directly. Ex. window.alert("hello"); is same as alert("hello");



Window Object:- It represents a window in browser. An object of window is created automatically by the browser. Window is the object of browser, it is not the object of javascript. The javascript objects are string, array, date, etc.

Methods:- 1) alert():- Displays the alert box containing message with ok button.

2) confirm():- Displays the confirm dialogbox containing message with ok and cancel button.

3) prompt():- Displays a dialog box to get input from the user.

4) open():- Opens the new window.

5) close():- closes the current window.

Ex:- 1) function msg() { alert("Hello AlertBox"); }

<input type="button" value="click" onclick="msg()"/>

Ex: 2) function msg() { var v=confirm("Are u sure?");

if(v==true) { alert("ok"); } else { alert("cancel"); } }

<input type="button" value="confirm" onclick="msg()"/>

Ex: 3) function msg() { var v=prompt("who are you?");

alert("I am "+v); }

<input type="button" value="click" onclick="msg()"/>

Ex: 4) function msg() { open("http://www.javatpoint.com"); }

<input type="button" value="click" onclick="msg()"/>.

→ History Object:- It represents an arrays of URLs visited by the user. By using this object, you can load previous, forward or any particular page.

Methods:- 1) forward():- Loads next page. Ex: history.forward();

2) back():- Loads the previous page, Ex: history.back();

3) go():- Loads the given pagenumber. Ex: history.go(2); // for next 2nd page.
history.go(-2); // for previous 2nd page

→ Navigator Object:- It is used for browser detection. It can be used to get browser information such as appName, appVersion, language, appCodeName etc. The navigator object is the window property, so it can be accessed by window.navigator or navigator.

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

Ex:- document.writeln("
 screen.width: " + screen.width);

→ Document Object Model (DOM):- It 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. By the help of document object, we can add dynamic content to our webpage. It is the object of window. It can be accessed by window.document or document.

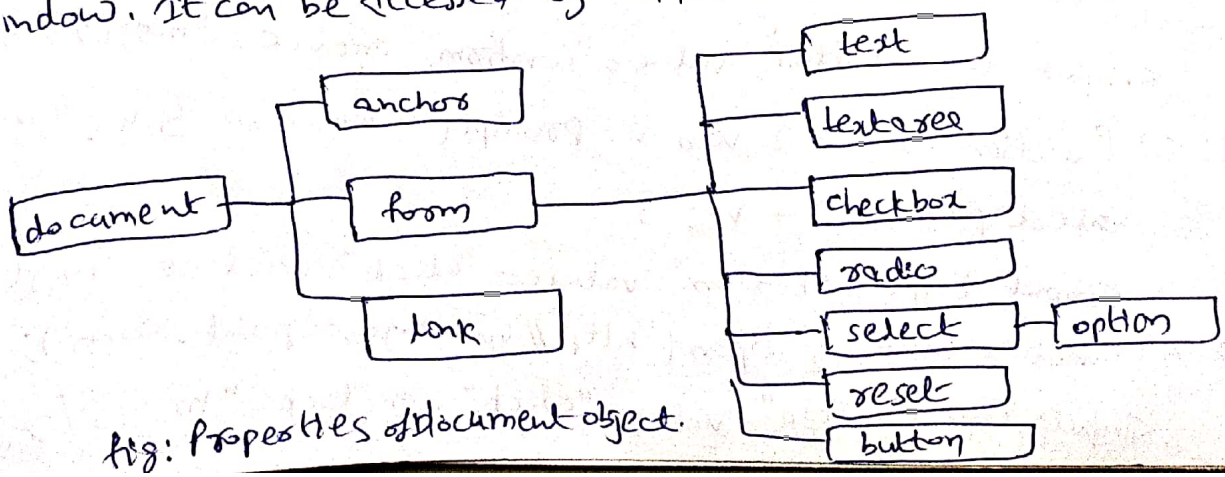


fig: Properties of document object.

Methods of document object :- We can access and change the contents of document by its methods.

1) write ("string") :- writes the given string on the document.

2) writeln ("string") :- writes the given string on the document with newline character at the end.

→ In this example, we are going to get the value of input text by user. Here, we are using document.form1.name.value to get the value of name field

→ Here, document is the root element that represents the html document. form1 is the name of the form, name is the attribute name of the input text. value is the property, that returns the value of the input text.

Ex:
```
<script type="text/javascript">
  function printvalue()
  {
    var name = document.form1.name.value;
    alert("Welcome: "+name);
  }
</script>
<form name="form1">
Enter Name: <input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

→ document.getElementById() :- The document.getElementById() method returns the element of specified id. This method is used to get value of the input text. But we need to define id for the input field.

→ // Example to print the cube of a given number
```
<script type="text/javascript">
function getcube()
{ var number = document.getElementById("number").value;
  alert(number*number*number); } </script>
```

```
<form>
Enter No: <input type="text" id="number" name="number"/>
<br/> <input type="button" value="cube" onclick="getcube()"/>
</form>
```

O/p: Enter No: 5 [Cube] → 125.

→ document.getElementsByName(): - This method returns all the elements of specified name.

Syn:- document.getElementsByName("name"); // name is required

→ In this example, we are going to count total number of genders using this method, to get all the genders.

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

O/p: Male ○ Female ⊙ [TotalGenders] Total Genders: 2

→ document.getElementsByTagName(): - This method returns all the elements of specified tagname.

Syn:- document.getElementsByTagName("name"); // name is required

→ innerHTML property:- The innerHTML property can be used to write the dynamic html on the html document. It is used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

→ In this example, we are going to create the html form when user clicks on the button. We are dynamically writing the html form inside the div name having the id mylocation. We are identifying this position by calling the document.getElementById() method.

```
<script type="text/javascript">
function showcommentform()
{
  var data="Name: <input type='text' name='name'><br>
            comment: <br><textarea rows='5' cols='80'>
            </textarea><br><input type='submit'
            value='post comment'>";
  document.getElementById('mylocation').innerHTML=data;
}
</script>
<form name="myform">
<input type="button" value="comment"
       onclick="showcommentform()">
<div id="mylocation"></div>
</form>
```

→ innerText property:- The innerText property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html but a normal text. It is used mostly in the webpages to generate the dynamic content such as writing the validation message, password strength etc

Ex: `<script type="text/javascript">` // To display password strength
`function validate() { var msg;` //when release the key after press
`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>`

→ JavaScript Validation:- It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

→ Javascript provides facility to validate the form on client side so data processing will be faster than server side validation, most of the web developers prefer Javascript form validation.

→ Through Javascript, we can validate name, password, date, email, mobile numbers and more fields.

→ Form Validation:- In this example, we are going to validate the name and password. The name can't be empty and password can't be less than 6 characters long. Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

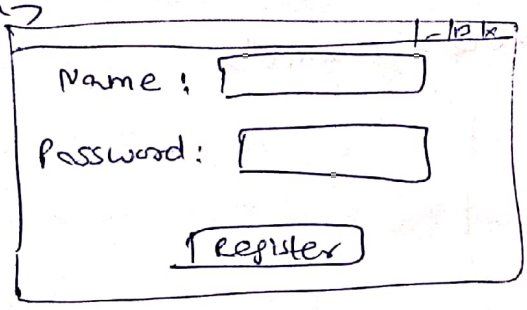
```
<script>
function validateform()
{
    var name=document.myform.name.value;
    var password=document.myform.password.value;

    if(name==null || name=="")
    {
        alert("Name can't be blank"); return false;
    }
    else if(password.length<6)
    {
        alert("Password must be atleast 6 characters long");
        return false;
    }
}
</script> <body>
<form name="myform" method="post" action="abc.jsp"
    onsubmit="return validateform()">
Name: <input type="text" name="name"><br/>
```

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

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

</form>

o/p:



→ JavaScript Retype Password Validation:-

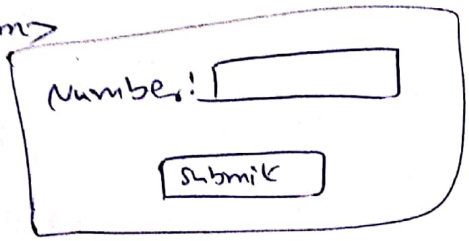
```
<script type="text/javascript">
 function matchpass()
 {
    var firstpassword = document.f1.password.value;
    var secondpassword = document.f1.password2.value;
    if(firstpassword==secondpassword) { return true; }
    else { alert("Password must be same!"); return false; }
 }</script>
 <form name="f1" action="register.jsp" onsubmit="return matchpass()">
Password: <input type="password" name="password"/><br/>
Re-Enter Password: <input type="password" name="password2"/><br/>
 <input type="submit">
 </form>
```

→ Java Script Number Validation:- Let's validate the textfield for numeric value only. Here, we use isNaN() function.

```
<script>
function validate()
{
    var num=document.myform.num.value;
    if(isNaN(num)){
document.getElementById("numloc").innerHTML="Enter Numeric value only";
    return false; }
  else{ return true; }
}</script>
```

```
<form name = "myform" onsubmit = "return validate()">
Number: <input type = "text" name = "num">
<span id = "numloc"></span> <br/>
<input type = "submit" value = "submit">
</form>
```

o/p:



→ Java Script Email Validation:- We can validate the email by the help of Javascript. There are many criteria that need to be follow to validate the email id such as: a) Email id must contain the @ and . character. b) There must be atleast one character before and after the @ c) There must be atleast 2 characters after dot.

→ Let's see the simple example to validate the email field.

```
<script>
function validateemail()
{
    var x=document.myform.email.value;
    var atposition=x.indexOf("@");
    var dotposition=x.lastIndexOf(".");
    if(atposition<1 || dotposition<atposition+2 ||
                (dotposition+2)>=x.length)
    {
alert("Please Enter a valid email address \n atposition:"+
        atposition+"\n dotposition:"+dotposition);
  return false;
  }
}</script>
<form name="myform" method="post" action="#"
                onsubmit="return validateemail();">
Email: <input type="text" name="email"> <br>
 <input type = "submit" value = "register">
</form>
```

o/p: Email: [ ]

JavaScript Events:- HTML events are "things" that happen to HTML elements. When JavaScript is used in HTML pages, JavaScript can "react" on these events.

→ An HTML event can be something the browser does, or something a user does. Examples are: 1) An HTML webpage has finished loading 2) An HTML input field was changed 3) An HTML button was clicked.

→ Often, when event happens, you may want to do something. Java script lets you execute code when events are detected. HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.

Syntax: 1) <element event = 'some JavaScript' >

(or) 2) <element event = "some JavaScript" >

Ex:1) <button onclick = "this.innerHTML = Date()">The time is ?</button>,

2) <button onclick = "displayDate()">The time is?</button>.

→ Common HTML Events:- 1) onchange:-(An HTML element has been changed), 2) onclick (The user click an HTML element) 3) onmouseover (user moves the mouse over an HTML element) 4) onmouseout (user moves the mouse away from an HTML element) 5) onkeydown (user pushes a keyboard key), 6) onload (browser has finished loading the page).

→ Event handlers can be used to handle, verify, user inputs, user actions, and browser actions: 1) Things that should be done every time a page loads. 2) Things that should be done when the page is closed. 3) Action that should be performed when a user clicks a button. 4) Content that should be verified when a user inputs data.

→ Many different methods can be used to let JavaScript work with events. 1) HTML event attributes can execute JavaScript code directly, can call JavaScript functions. 2) You can assign your own event handler functions to HTML elements.

→ JavaScript Errors Handling:- 1) The try statement lets you test a block of code for errors. 2) The catch statement lets you handle the error. 3) The throw statement lets you create custom errors. 4) The finally statement lets you execute code, after try and catch, regardless of the result.

→ When executing Javascript code, different errors can occur. Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.

```
<p id="demo"></p>
<script>
try
{
  adddlert("Welcome Guest");
}
catch(err)
{
  document.getElementById("demo").innerHTML=err.message;
}
</script>
```

→ Javascript catches adddlert as an error, and executes the catch code to handle it.

→ try and catch:- The try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code tobe executed, if an error occurs in the try block.

Syn:-
```
try
{
  Block of code to try
}
catch(err)
{
  Block of code to handle errors
}
```

→ throw:- The throw statement allows you to create a custom error. Technically you can throw an exception (throw an error). The exception can be a Javascript String, a Number, a Boolean, or an Object.

Ex: throw "too big"; // throw a text
throw 500; // throw a number

→ If you use throw together with try and catch, you can control program flow and generate custom error messages.

→ Input Validation:- This example examines input. If the value is wrong, an exception (err) is thrown. The exception (err) is caught by the catch statement and a custom error message is displayed.

```
<!DOCTYPE html>
<html>
<body>
<p> Please Input a number between 5 and 10: </p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">
        Test Input </button>
<p id="p01"></p>

<script>
function myFunction()
{
    var message, x;
    message = document.getElementById("p01");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try
    { if(x=="") throw "empty";
      if(isNaN(x)) throw "not a number";
      x = Number(x);
      if (x<5) throw "too low";
      if(x>10 throw "too high";
    }
    catch(err) { message.innerHTML = "Input is" + err;
    finally { document.getElementById("demo").value = "";
}} </script>
</body>
</html>
```

→ Finally statement:- The finally statement lets you execute code, after try and catch, regardless of the result

Syntax:
```
try
{ Block of code to try
}
catch(err)
{
   Block of code to handle errors
}
finally
{
   Block of code to be executed regardless of the try/catch result
}
```

→ Error Object:- Java script has a built in error object that provides error information when an error occurs. The error object provides two properties: name (sets or returns an error name) and message (sets or returns an error message or string).

→ Error Name Values:- There are 6 different types of values can be returned by the error name property.

1) EvalError:- An error has occurred in the eval() function.
2) RangeError:- A number "out of range" has occurred.
3) ReferenceError:- An illegal reference has occurred.
4) SyntaxError:- A syntax error has occured.
5) TypeError:- A type error has occurred
6) URIError:- An error in encodeURI() has occurred. (Uniform Resource Identifier)

→ Java Script Hoisting:- It moves the declaration of variables and functions at the top. So, in Javascript we can use variables and functions before declaring them. Hoisting is applicable only for declaration not initialization. It is required to initialize the variables and functions before using their values.

```
<script> x=10; // variable Hoisting,
  document.write(x); var x;
</script>
```

```
<script> // function Hoisting
document.write(sum(10,20));
function sum(a,b)
{ return (a+b); }
</script>
```

JavaScript Regular Expressions:- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

→ A regular expression can be a single character, or a more complicated pattern. Regular expressions can be used to perform all types of text search and text replace operations.

Syntax: /pattern/modifiers;

Ex:- var patterns = /w3schools/i; where /w3schools/i is a regular expression. w3schools is a pattern (to be used in a search) i is a modifier (modifies the search to be case-insensitive).

→ In JavaScript, regular expressions are often used with the two string methods: search() and replace().

→ The search() method uses an expression to search for a match, and returns the position of the match. The replace() method returns a modified string where the pattern is replaced.

Ex: 1) var str = "Visit w3schools";

2) var n1 = str.search("w3schools"); // 6, search for substring

3) var n2 = str.search(/w3schools/i); // 6 (Use Regular expression to do a case-insensitive search for "w3schools" in a string)

4) var str = "Visit Microsoft";

var res1 = str.replace("Microsoft", "w3schools"); // Visit w3 schools.

4) Var res2 = str.replace(/microSoft/i, "w3schools"); // Visit w3 schools

(Use a case-insensitive regular expression to replace Microsoft with w3schools in a string)

→ Regular Expression Modifiers:- Modifiers can be used to perform case-insensitive more global searches:

a) "i":- Perform case-insensitive matching

b) "g":- Perform a global match (find all matches rather than stopping after the first match).

c) "m" :- Perform multiline matching

→ Regular Expression Patterns:- Brackets are used to find a range of characters.

a) Expression [abc] ⇒ Find any of the characters between the brackets.

b) [0-9] :- Find any of the digits between the brackets.

c) (x|y) :- Find any of the alternatives separated with |

→ Meta characters:- characters with a special meaning.

① \d ⇒ find a digit ② \s ⇒ Find a white space character

→ Quantifiers:- It defines the quantities.

① "n+" ⇒ Matches any string that contains atleast one n.

② "n*" ⇒ Matches any string that contains 0 or more occurrences of n.

③ "n?" ⇒ Matches any string that contains zero or one occurrences of n.

→ test():- The test method is a "RegExp" expression method. It searches a string for a pattern, and return true or false, depending on the result.

Ex: ① Var patt = /e/; // Search a string for the character "e".

patt.test("The best things in life are free"); // o/p is true

(or) /e/.test("The best things in life are free"); // true

→ exec():- The exec() method is a "RegExp" expression method. It searches a string for a specified pattern, and returns the found text as an object. If no match is found, it returns an empty (null) object.

Ex:- /e/.exec("The best things in life are free"); // 2