What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language (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 is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Difference between Java and JavaScript:

Java	Java Script
Object Oriented	Object Based
Strongly Typed	Loosely Typed
Supports Static & Dynamic binding	Supports only Dynamic binding
Availability of Classes & Packages	Non availability of Classes & Packages

Are Java and JavaScript the same?

- NO!
- Java and JavaScript are two completely different languages in both concept and design!
- Basically java is
- Object Oriented
- It is strongly typed which means it supports various data types
- Supports Static & Dynamic binding, that is, polymorphism
- Supports Classes & Packages
- Java (which is developed by Sun Microsystems) is a powerful and much more complex programming language.
- Whereas, javascript is
- Object Based (means does not support inheritance)
- Loosely Typed (means does not have data types to declare variables)
- Supports only Dynamic binding
- Classes & Packages are not available

What can a JavaScript do:

- JavaScript gives HTML designers a programming tool
- JavaScript can put dynamic text into an HTML page
- JavaScript can react to events
- JavaScript can read and write HTML elements
- JavaScript can be used to validate data
- JavaScript can be used to detect the visitor's browser
- JavaScript can be used to create cookies

So let us look at what Java Script can do.

JavaScript gives HTML designers a programming tool

HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can work on it

JavaScript can put dynamic text into an HTML page

A JavaScript statement like this:

document.write("Welcome " + name)

allows us to write a variable, name as specified in this example into an HTML page

JavaScript can react to events

A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

JavaScript can read and write HTML elements

A JavaScript can read and change the content of an HTML element

JavaScript can be used to validate data

Form data can be validated before it is submitted to a server, this will save the server from extra processing

JavaScript can be used to detect the visitor's browser

And depending on the browser - load another page specifically designed for that browser

JavaScript can be used to create cookies

You can use JavaScript to store and retrieve information on the visitor's computer in the form of cookies.

Structure of a JavaScript:

Scripts in either the head or the body section

```
<html>
   <head> </head>
   <body>
     <script language="JavaScript">
....
       </script>
   </body>
 </html>
```

Scripts in both the head and the body section

```
<html>
  <head>
     <script type="text/javascript">
....
     </script>
  </head>
  <body>
     <script type="text/javascript">
....
     </script>
  </body>
</html>
```

Let's take a look at the structure of Java Script

The HTML <script> tag is used to insert a JavaScript into an HTML page

The lines between the <script> and </script> contain the JavaScript and are executed by the browser.

JavaScripts can be put in the <body> and in the <head> sections of an HTML page

From the given example, it is clear that java script can be inserted either into the head section or the body section or both of an HTML page

Generally, JavaScript in a page will be executed immediately while the page loads into the browser. This is not always what we want.

Sometimes we want to execute a script when a page loads, other times when a user triggers an event. Event handling would be dealt with later.

AN EXAMPLE

```
<HTML>
  <BODY>
    <SCRIPT LANGUAGE="javascript">
<!-- document.writeln("<H2 ALIGN=center><B>Hi! This is Java Script and your browser supports it.</H2></B>")//--
>
    </SCRIPT>
    <NOSCRIPT>Java Script is not supported by your browser.</NOSCRIPT>
     This text is out side the script tag in the body section
  </BODY>
</HTML>
```

Looking at the example given here, we notice that some java script code is written within <!—and //--> What does this mean?

Browsers that do not support JavaScript, will display JavaScript as page content.
To prevent this, just add an HTML comment tag <!-- before the first JavaScript statement, and a --> (end of comment) after the last JavaScript statement

The two forward slashes at the end of comment line (//) is the JavaScript comment symbol. This prevents JavaScript from executing the --> tag.

You can use the <noscript> tag to display an appropriate message if your browser does not support java script.

<source>

```
<HTML>
<BODY>

<script langauage = VBscript>
On Error Resume Next
Dim a
a = 1
b 2
MsgBox a + b
</script>
</BODY>
</HTML>
```

</source>

<Execute>

Hi! This is Java Script and your browser supports it.

This text is out side the script tag in the body section

</Execute>

JavaScript Statements:

How do you write java script statements?

- JavaScript is a sequence of statements to be executed by the browser
- These statements can be grouped together to form blocks.
- A block starts with a left curly bracket {, and end with a right curly bracket }.
- The purpose of a block is to make the sequence of statements execute together.

```
<script type="text/javascript">
    document.write("<h1>This is a heading</h1>");
    document.write("<p>This is a paragraph.</p>");
    document.write("<p>This is another
paragraph.</p>");
</script>
```

```
<script type="text/javascript">
{
    document.write("<h1>This is a heading</h1>");
    document.write("<p>This is a paragraph.</p>");
    document.write("<p>This is another
paragraph.</p>");
}
</script>
```

JavaScript Comments:

A java script comment prevents the browser from executing it.
As seen here, JavaScript comments can be used to make the code more readable.
// is a comment that is used to prevent the execution of a single code line
And Multi line comments start with /* and end with */
It prevents the execution of a set of code

JavaScript Comments

```
<script type="text/javascript">
// Write a heading
      document.write("<h1>This is a heading</h1>");

/*
The code below will write one heading and two paragraphs
*/
      document.write("<h1>This is a heading</h1>");
      document.write("<p>This is a paragraph.</p>");
      document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript variables:

We know that variables are used to hold data
•Can be declared using var
If within a JavaScript function, variable becomes LOCAL and hence can be accessed only within that function. (we say that the variable has local scope).
•In such a case, it is destroyed when you exit the function
Global Variables need not be declared using var

•if a variable is declared outside a function it becomes GLOBAL
•all scripts and functions on the web page will be able access it in this case
•they are destroyed when the page is closed
•If you declare a variable, without using "var", the variable always becomes GLOBAL
As shown in the example here,variables x and name become local if declared within a function and z is automatically global as var is not used

Local Variables

- Declared within a JavaScript function and becomes LOCAL
- Can only be accessed within that function. (the variable has local scope).
- Destroyed when you exit the function

Global Variables

- Declared outside a function become GLOBAL
- All scripts and functions on the web page can access it
- Destroyed when the page is closed
- If you declare a variable, without using "var", the variable always becomes GLOBAL

```
var x=5;

var name="anu";

z = x * 50;
```

Operators:

- **Arithmetic operators** are used to perform arithmetic between variables and/or values
- **Assignment operators** are used to assign values to JavaScript variables
- **Comparison operators** are used in logical statements to determine equality or difference between variables or values.
- **Logical operators** are used to determine the logic between variables or values
- **Conditional operator** assigns a value to a variable based on some condition
- As shown in the example, x is assigned to result if x>y else y is assigned

- All the above operators are illustrated as shown in the slide

Operators

Arithmetic Operators

- + - * / % + + --
- x=y++;For y=5, x=5 and y=6

  x=y*2; For y=5, x=10 and y=5

Assignment Operators

- = + = - = * = / = % =
- x+=y;For x=15 and y=5, x=20

  x%=y;For x=17 and y=5, x=2

Comparison Operators

- == < > === != <= >=
- For x=5, x==8 is false

  x===5 is true

  x=="5" is false

  === (exactly equal value and type)

Logical Operators

- && || !
- For x=6 and y=3,

  (x<10 && y>1) is true

  !(x==y) is true

Conditional Operator

- variablename=(condition)?value1:value2
- For x=7 and y=3,

  result=(x>y)?x:y

  result is 7

JavaScript IfElse Statements:

Here we discuss about the various programming constructs supported by java script.

- **if** statement - use this statement to execute some code only if a specified condition is true

oin the example here, if name is equal to anu only then welcome message would be displayed

- **if...else** statement - use this statement to execute some code if the condition is true and another code if the condition is false

oif name is equal to anu only then welcome message would be displayed else "Sorry, Invalid name" would be displayed

- **if...else if**....else statement - use this statement to select one of many blocks of code to be executed

oSimilarly, if name is equal to scott, then Admin would be displayed, else if name is anu, user is displayed, else invalid message is displayed

Java Script if-else Statement

if	if (condition) {
	//code to be executed if condition is true
	}
		if (name == "anu") {
		document.write("Welcome");
		}

if..else	if (condition) {
	//execute code if true
	} else {
	//execute code if false
	}
		if (name == "anu") {
		document.write("Welcome");
		} else {
		document.write("Sorry, Invalid name!")
		}

if-else if-else	if (condition1) {
	//execute code if true
	} else if (condition2) {
	//execute code if true
	} else {
	//execute code if neither condition1 nor condition2 is true
	}
		if (name == "scott") {
		document.write("Admin");
		} else if (name == "anu") {
		document.write("User");
		} else {
		document.write("Invalid!");
		}

Switch Statement:

switch statement - use this statement to select one of many blocks of code to be executed.
We have a variable n whose value is compared to the values for each case in the structure.
If there is a match, the block of code associated with that case is executed.
Use **break** to prevent the code from running into the next case automatically.
If none of the cases are satisfied, the default case is executed.
For example, if a menu is shown, based on the choice, the operation is performed, like add, modify etc.

```
switch(n)
{
   case 1:
     //  execute code block 1
     break;
   case 2:
     //  execute code block 2
     break;
   default:
    // execute this if none of
above
        cases are satisfied
}
```

```
var choice = 1;
switch(choice)
{
   case 1: document.write('Add');
         break;
   case 2:document.write('Modify');
     break;
   case 3:document.write('Delete');
     break;
   case 4:document.write('View');
     break;
   default : document.write('Invalid
Choice...');
}
```

JavaScript loop Statements:

Let's look at the looping statements in javascript
Loops execute a block of code a specified number of times, or while a specified condition is true
The **for** loop -is used when you know in advance how many times the script should run. It has the initialization part, condition and increment/decrement part
The **while** loop- loops through a block of code while a specified condition is true
The **do...while** loop -is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true
Here we see an example of displayed values 1 to 10 using all the three types of loops

Java Script loop Statements

for

```
for (v=startvalue; v<=endvalue; v=v+increment) {
    //execute if true
}
```

```
for (i=1;i<=10; i++) {
    document.write("<b> Value i = " + i );
}
```

while

```
while (v <= endvalue) {
    //execute code if true
}
```

```
var i = 1;
while (i <= 10) {
    document.write("The number is " + i);
    i++;
}
```

do-while

```
do {
    //execute code until condition becomes false
} while (v <= endvalue);
```

```
var i = 1;
do {
    document.write("Value is " + i);
    i++;
} while (i <= 10);
```

Break and continue Statements

break

```
var i=0;
for (i=0; i<=5;  i++) {
    if (i==3) {
break;
        }
            document.write("<br/>"+i);
            }
```

```
Output:
0
1
2
```

continue

```
var i=0;
for (i=0;i<=5;i++)   {
    if (i==3)   {
continue;
    }
     document.write("<br/>i = " + i);
}
```

```
Output:
0
1
2
4
5
```

The break statement -will **break** the loop and continue executing the code that follows after the loop (if any).

The **continue** statement- will break the current loop and continue with the next iteration

For…In Statement:

We have a special for loop called for..in to loop through the properties of an object and display them
As shown in the example, person has three properties such as First Name, Last Name and Age

For...In Statement

```
for (variable in object)
{
     code to be
executed
  }
```

- variablename=(condition)?value1:value2

```
var person={"First Name":"John","Last Name":"Doe",Age:25};
for ( x  in person )  {
document.writeln(x + " -> "+person[x] + "<br/>");
 }
```

- **Output :**
- First Name -> John
- Last Name -> Doe
- Age -> 25

JavaScript Functions:

To keep the browser from executing a script automatically when the page loads, you can put your script into a function and then based on an event that occurs you may call the function.
A function contains code that will be executed by an event or by a call to the function.
You may call a function from anywhere within a page
Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read or loaded by the browser before it is called, it could be wise to put functions in the <head> section

The parameters var1, var2, etc. are variables or values passed into the function. The left curly bracket{ and the right curly bracket} define the start and end of the function
The *return* statement is used to specify the value that is returned from the function.
So, functions that are going to return a value must use the return statement

JavaScript Functions

Defining a Function

```
function functionname(var1,var2,...,varX)
{
    some code
}
```

The return Statement

```
function functionname(var1,var2,...,varX)
{
    some code
    return somevalue;
}
```

An Example

```
<head>
<script type="text/javascript">
    function product(a,b)   {
        return a*b;
    }
  </script>
</head>
<body>
  <script type="text/javascript">
      document.write(product(4,3));
  </script>
</body>
```

Applications of JavaScript:

- Therefore, we summarize the applications of JavaScript, it is a client-side language
- It is often used to perform operations like form input validation.
- Java Scripts are integrated into the browsing environment
- They can also react to events

Applications of Java Script

- JavaScript is a *client-side* language
- JavaScript is often used to perform operations like form input validation.
- Java Scripts are integrated into the browsing environment
- Java Scripts can also react to events

Dialogue Boxes:

Dialog Boxes in Java Script

Alert This Dialog box is used to display an alert message.

Confirm This Dialog box is used to confirm the message from user. It will return true/false.

Prompt This Dialog box is used to take input from the user. Input value is stored in a variable.

Alert	Eg: alert("Hello world");
Confirm	Eg: x=confirm("Do you want to rename");
Prompt	Eg: a=prompt("Enter a value");

<source>

```
<html>
   <body>
         <script>
          var no=prompt("Enter a number","");
                  if(no>5)
                        document.write("You have entered a number greater than five.");
              else
                        document.write("You have entered a number lessthan or equalto five");
                  </script>
   </body>
```

</source>

<Execute>

You have entered a number less than or equalto five

</Execute>

Event Handlers:

Events are actions that can be detected by JavaScript
Few event handlers are specified below
- onBlur
- onChange
- onClick
- onFocus

A function can be executed either by an event or by a call to the function. For example, onClick event of a button performs some action.
Examples of events:
- A mouse click
- A web page or an image loading
- Moving the mouse over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke
- onMouseDown
- onMouseOver
- onMouseUp

onblur

onchange

onclick

onfocus

onmousedown

onmouseover

onmouseup

Let's discuss the use of few event handlers
- **onLoad** and **onUnload** used to trigger an event when the user enters and leaves the page respectively
- **onBlur** - In the given example, when the email text field loses focus, we invoke the checkEmail function to validate the content
- The **onSubmit** event is used to validate ALL form fields before submitting it
- The other events specified here are self explanatory
- A sample on using events is demonstrated later

onLoad and onUnload

onLoad and onUnload

- triggered when the user enters or leaves the page

onFocus, onBlur and onChange

- often used in combination with validation of form fields
- <input type="text" size="30" id="email" onchange="checkEmail()">

onSubmit

- The onSubmit event is used to validate ALL form fields before submitting it
- <form method="post" action="xxx.htm" onsubmit="return checkForm()">

onMouseOver

- The onmouseover event can be used to trigger a function when the user mouses over an HTML element

Here's an example that shows how the message() method is invoked on clicking the hyperlink, result of which displays an alert message saying "Hey, it's working".

<source>

```
<HTML>

<BODY>
<script>
 function message()
 {
    alert("Hey It's working!");
 }
</script>

        <A HREF="example5.3.html" onclick="message()">Hello! Click me</A>
        </FORM>
</BODY>
</HTML>
```

</source>

<Execute>

Hello!Click me

</Execute>

External JavaScript:

When a javascript is included in a html page, it is useful to the current web page only. Therefore, in order to reuse it for other web pages also, we can include the script into a separate text file with an extension as ".js". This gives us more security.
The src attribute of the script tag is used to specify the location of the java script file.

```
<HTML>
    <HEAD>
<script language="javascript" src="external.js">
</script>
    </HEAD>
    <BODY>
        <A HREF="example.html"  onclick="sayHello()"> Click me </A>
     </BODY>
</HTML>

external.js
   function sayHello()   {
alert ("Hello!");
    }
```

What is an Object:

•Properties are the values associated with an object. For example, consider
• variable txt="Hello World!"; Therefore, length is a property that displays the size of the text, that is, 12 for Hello World!
//document.write(txt.length);
•Methods are the actions that can be performed on objects. For example,
•variable str="Hello world!"; Therefore, toUpperCase() is a method that displays the string in upper case
//document.write(str.toUpperCase());

The document itself is an object, made up of other objects like forms, images and tables
Form objects are further made up of objects like text boxes and submit buttons

What is an Object?

An object is a special kind of data that has properties and methods

Properties **are the values associated with an object**

```
var txt="Hello World!";
document.write(txt.length);
```

Methods **are the actions that can be performed on objects**

```
var str="Hello world!";
document.write(str.toUpperCase());
```

JavaScript can read these properties and modify them, or react to events that happen to the objects, thus changing the object in the browser window.

Few Java Script Objects are specified below:

- Array
- String
- Date
- Math

Array Object in JavaScript:

Here we discuss about the **Array** object in java script, its properties and methods
The Array object is used to store multiple values in a single variable
Creation of this object is as shown here
It has a property called length that gives the number of elements it contains

Array Object in Java Script

Used to store multiple values in a single variable

```
var names=new Array(); // regular array (add an optional integer
names[0]="anu";       // argument to control array's size)
names[1]="raj";
names[2]="karan";     //OR
var  names=new Array("anu","raj","karan"); // condensed array    OR
var names=["anu","raj","karan"]; // literal array
```

- length - Returns the length of a string
- concat() - Joins two or more arrays, and returns a copy of the joined arrays
- join() - Joins all elements of an array into a string

- pop() - Removes the last element of an array, and returns that element
- push() - Adds new elements to the end of an array, and returns the new length
- reverse() - Reverses the order of the elements in an array
- shift() - Removes the first element of an array, and returns that element
- slice() - Selects a part of an array, and returns the new array
- sort() - Sorts the elements of an array
- splice() - Adds/Removes elements from an array
- toString() - Converts an array to a string, and returns the result
- unshift() - Adds new elements to the beginning of an array, and returns the new length

String Object

used to manipulate a stored piece of text

- var str="hello world"; //OR
- var str = new String("hello world");
- length - Returns the length of a string
- charAt() - Returns the character at the specified index
- charCodeAt() - Returns the Unicode of the character at the specified index
- concat() - Joins two or more strings, and returns a copy of the joined strings
- indexOf() - returns position of the first occurrence of a specified value in a string
- lastIndexOf() – returns position of last occurrence of specified value in a string
- replace() - replace a specified value with another value in a string
- search() - returns position of a match between a regular expression and a string
- slice() - Extracts a part of a string and returns a new string
- split() - Splits a string into an array of substrings
- substr() - Extracts characters from a string, from start position specified, and through the specified number of character
- substring() - Extracts the characters from a string, between two specified indices
- toLowerCase() - Converts a string to lowercase letters
- toUpperCase() - Converts a string to uppercase letters

Date Object

used to work with dates and times

- var d=new Date(); document.write(d); //get's today's date
- new Date(milliseconds) //milliseconds since 1970/01/01
- new Date(dateString)
- new Date(year,month,day,hours,minutes,seconds,milliseconds)
- Few commonly used methods are:
- getDate() - day of the month (from 1-31)
- getDay() - day of the week (from 0-6)
- getFullYear() - year (four digits)
- getHours() - hour (from 0-23)
- getMilliseconds() - milliseconds (from 0-999)
- getMinutes() - minutes (from 0-59)
- getMonth() - month (from 0-11)
- getSeconds() - seconds (from 0-59)
- getTime() - number of milliseconds since midnight Jan 1, 1970
- parse() - Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
- toString() - Converts a Date object to a string
- Similarly we have setters for each of the above

EXAMPLE //Compares two dates

```
var x=new Date();
x.setFullYear(2011,0,14);
var today = new Date();
 if (x>today)  {
    alert("Today is before 14th Jan
2011");
  }  else  {
    alert("Today is after 14th Jan
2011");
  }
```

Source Exec

Click each button to know n

<source>

```
<HTML>
<BODY>
<script language="javascript">
function display_date()
{
var d=new Date();
var day=d.getDate();
var month=d.getMonth()+1;
var year=d.getYear();
        alert("today's date is \n"+day+"-"+month+"-"+year);
}
</script>
   <input type="button" value="date" onclick="display_date()">
</BODY>
</HTML>
```

</source>

<Execute>

date

</Execute>

Math Object:

It is used to perform common mathematical tasks

Math Object

allows you to perform mathematical tasks

```
var x = Math.PI; // Returns PI
var y = Math.sqrt(16); // Returns the square root of 16
```

Properties

- PI - approx. 3.14159
- E - Euler's number (approx. 2.718)
- LN2 - natural logarithm of 2 (approx. 0.693)
- LN10 - natural logarithm of 10 (approx. 2.302)
- SQRT1_2 - square root of 1/2 (approx. 0.707)
- SQRT2 - square root of 2 (approx. 1.414)

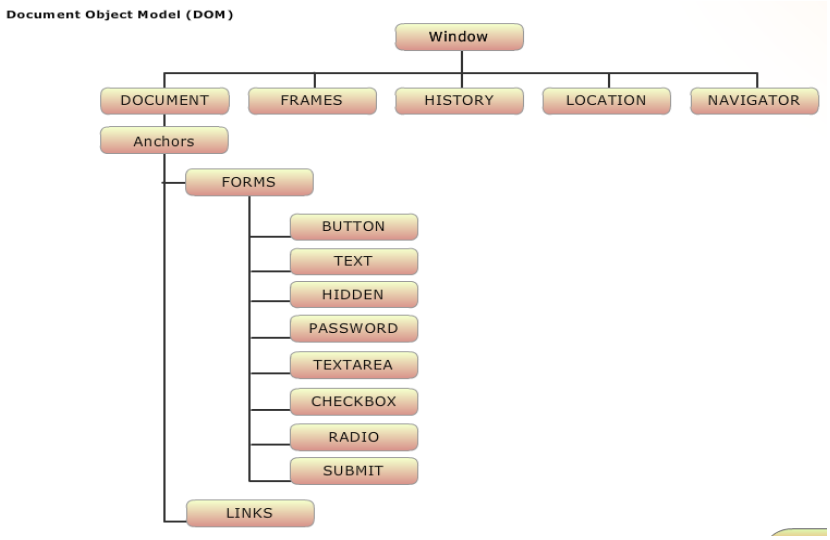
Its common methods are as follows:

onmouseover

- abs(x) - absolute value of x
- cos(x), sin(x), tan(x) – cosine, sine, tangent of x, in radians
- acos(x), asin(x), atan(x) – arccosine, arcsine, arctangent of x, in radians
- ceil(x) – rounds x upwards to the nearest integer
- exp(x) - value of Ex
- floor(x) – rounds x downwards to the nearest integer
- log(x) - natural logarithm (base E) of x
- max(x,y,z,...,n) - number with the highest value
- min(x,y,z,...,n) - number with the lowest value
- pow(x,y) - value of x to the power of y
- random() - random number between 0 and 1
- round(x) - Rounds x to the nearest integer
- sqrt(x) - square root of x

Document Object Model:

- The root in the DOM hierarchy is the **Window** object
- Each HTML document when loaded into a browser window becomes a **Document** object.
- The Document object provides access to all HTML elements in a page, from within a script, and can be accessed through the *window.document* property.



The DOM defines a standard way for manipulating and accessing HTML documents.
Some of the objects in the DOM model are as follows:

- The **Window** object relates to the current browser window
- The Document object represents a web page that contains the various HTML Elements. It relates to the <body> tag and can be accessed through the window.document property.
- **Frames** is n array of frames if the window object contains any. Each frame in turn refers to another Window object.
- **History** contains the current windows history list, namely, the collection of the various URL visited by the user recently. It can be accessed through the window.history property.
- **Location** contains information about the current URL, that is, the current location of the document being viewed in the form of an URL. It can be accessed through the window.location property.
- **Navigator** is an object that describes the basic characteristics of the browser and can be accessed through the window.navigator property

DOM

The DOM defines a standard way for manipulating and accessing HTML documents.

Some of the objects of java script are as follows:

window	The object relates to the current browser window
document	An object that contains the various HTML Elements. It relates to the body tag.
frames	An array of frames if the window object contains any. Each frame in turn refer to another Window object.
history	Contains the current windows history list, namely, the collection of the various URL visited by the user recently.
location	Contains the current location of the document being viewed in the form of an URL and its constituent pieces.
navigator	An object that describes the basic characteristics of the browser

---

Here is an example of opening the e-support Window through Java Script using the window.open() method and closing the window using window.close() method.

<source>

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<script language="javascript">
function gotoesupport(){
winRef=window.open("http://esupport.satyam.com/","esupport","resizable='no'","yes");
}
function exitesupport(){
winRef.close();
}
</script>
<input type="button" onclick="gotoesupport()" value="e-support">
<input type="button" value="close" onclick="exitesupport()">

</BODY>
</HTML>
```

</source>

<Execute>

e-support close

Click on the 'e-support' button to open e-support and Close button to close e-support

</Execute>

Document Object:

Let's understand the collections available in the Document object

- anchors[] - Returns an array of all the anchors in the document
- forms[]- Returns an array of all the forms in the document
- images[]- Returns an array of all the images in the document
- links[]- Returns an array of all the links in the document

The **Anchor** object in java script corresponds to the <a> (anchor) tag in html

The **Form** Object corresponds to the <form> tag in html. Forms are used to collect user input, and contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select menus, textarea, fieldset, legend, and label elements. Forms are used to pass data to a server

The **Link** object represents an HTML link element that must be placed inside the head section of an HTML document, and it specifies a link to an external resource

The **Image** Object corresponds to the <img>(image) tag in an HTML document and represents an embedded image.

The **Frame** object corresponds to the <frame> tag that defines one particular window (frame) within a frameset

Document Object

- All
- Anchor
- Anchors
- Applet
- Applets
- Embeds
- Forms
- Frames
- Image
- Images
- Link
- Links
- Scripts
- Selection

Form:

The **Form** object further contains other objects such as Buttons, Checkboxes, Radio buttons etc. Let's discuss few of these objects here (Please refer to the powerpoint slides for the properties and methods of these objects in detail).
The Button object represents a push button and corresponds to the <button> tag in an HTML document
The **Checkbox** object corresponds to the <input type="checkbox"> tag in an HTML form and allows a user to select one or more options of a limited number of choices.
The **FileUpload** object corresponds to the <input type="file"> tag in an HTML form and allows file uploading to a server

The **Hidden** object corresponds to the <input type="hidden"> tag in an HTML form and is used to send hidden form data to a server
The **Password** object corresponds to the <input type="password"> tag in an HTML form. The content of a password field will be masked in a browser
The **Radio** object corresponds to the <input type="radio"> tag in an HTML form. It allows the user to select only ONE of a predefined set of options
The **Text** object corresponds to the <input type="text"> tag in an HTML form. It represents a single-line text input field

The **Textarea** object corresponds to the <textarea> tag in an HTML form
The **Select** object corresponds to the <select> tag in an HTML form and represents a dropdown list that lets a user select one or more options of a limited number of choices.
The **Submit** object corresponds to the <input type="submit"> tag in an HTML form. Clicking on a submit button sends the named contents of the form to the server
The **Option** object corresponds to the <option> tag in an HTML form and represents an option in a dropdown list.

Button
Checkbox
Elements
Fileupload
Hidden
Option
Password
Radio
Select
Submit
Text
Textarea

Client Side Validation using JavaScript:

Validations can be done in one of the two ways

- validating every field individually.
- validating all fields while submitting the form.

Validating the form fields is one of the most important features of javascript. These validations help us in checking whether users are entering correct data or not.

The example specified here validates the username and password before submitting the data to ensure that the values are entered by the user as they are mandatory fields. If not entered, then an error message is alerted else the form data is submitted to the server.

Client-Side validation using Java Script

```
<html>
   <head>
      <script language="javascript"><!--
         function validate()  {
var name=document.loginfrm.user.value;
var pwd= document.loginfrm.password.value;
if(name==""  || pwd==="")   {
     alert("Please enter all fields");
     return false;
}
return true;
         }
//-->    </script>
     </head>
     <body>
         <FORM  name="loginfrm"  onsubmit="return
validate()">
  User Name :<INPUT TYPE="text" NAME="user"><br>
Password   :<INPUT TYPE="password" NAME="password">
<br>
<INPUT TYPE="submit" value="submit" >
        </FORM>
     </body>
</html>
```

Source Execute Source Execute

<source>

```
<html>
<head>
              <script language="javascript">
                    function validate()
                    {

                                    if(document.loginfrm.user.value=="" || document.loginfrm.password.value==="")
                                    {
                                            alert("Please enter all fields");
                                            document.loginfrm.user.focus();
                                            return false;
                                    }
                                    return true;
                    }
     </script>
</head>
<body>

<FORM  NAME="loginfrm"  onsubmit="return validate()">
     User Name :     <INPUT TYPE="text" NAME="user">          <br>
     Password   :    <INPUT TYPE="password" NAME="password">  <br>
     <INPUT TYPE="submit" value="submit" >
</FORM>

</body>
</html>
```

</source>

<Execute>

User Name :
Password :
submit

</Execute>

*** For <u>Task</u> and <u>Advanced JavaScript Examples</u> refer elinc ***

☺☺☺☺☺☺☺☺☺