Unit-3

Introduction to JavaScript

Content

- ✓JavaScript Basics
- ✓Strings, Arrays, Functions
- ✓Objects in JavaScript
- ✓building simple applications using JavaScript and HTML

Introduction

→ JavaScript is:

→ An object-oriented scripting language that is designed primarily for people who are building web pages using HTML

→ JavaScript programs are embedded within HTML documents in source code form.

→ The script is interpreted by the browser.

→ Platform independence.

Contd…

→Javascript is object-oriented.

→It allows interaction with the properties of object that it recognizes.

→Internal built-in objects(e.g. document object)

→Browser objects (e.g. window object).

→Javascript program can run both on the client and the server sides

→Client side:

-- By the browser after downloading the HTML document

→Server side:

--requires Netscape Livewire environment

Javascript Objects and Methods

→In the context of Javascript:

→An object is a collection of properties and methods which can be viewed, modified and interacted.

→ A simple example of a property is color, which is rather easy to visualize.

→They can be directly manipulated by referring to the object and the property by name and then setting its value.

→Example: The background color of a page can be changed as:

document.bgcolor="blue"

Contd.

→In the object-oriented paradigm, methods refer to functions that can be used to manipulate objects and their properties.

→Example: The method write(), which when invoked in the document object, causes a specific string of characters to be outputted.

```
document.write("Hello , Welcome to Javascript world!");
```

Contd.

→ Methods which are defined on an object give the range of choices available for interacting with the object.

→ Examples:

- → A *window* object can be opened or closed using the *open()* and *close()* methods respectively.
- → A *form* object has a *submit()* method which transmits the contents of the form to the web server.
- → The sequential list of a user's path through a member of URLs is represented by the *history* object, which has *forward()* and *backward()* methods to move through the list.

Contd.

→A part from the pre-defined methods, it is also possible to create user-defined methods.

→Control the rate with which a line of text scrolls across the screen.

→Determine the path of an animated object across the display.

→Event handlers can be coded in Javascript.

→They are triggered in response to certain conditions in some objects.

→Can be used to control the sequence of activities in response to some object state.

Running Scripts/Embedding script

→Scripts written in Javascript must either be embedded within an HTML document or be referenced as an external file that is loaded with the HTML document.

→ All recent browsers can detect and interpret inline Javascript code directly.

→ The <SCRIPT> tag is used.

→ Syntax:

```
<script language="javascript">
        //JavaScript code goes here
</script>
```

Including external JavaScript files

How to include external JS filel

→Can be done using the src attribute in the <script> tag.

Example:

```
<script src="../../codes/test.js">
</script>
```

→ This behaves exactly as if the contents of the specified javascript file appeared directly between the tags.

Example: Referring external JavaScript file

HTML file: MyWebpage.htm

```
<html>
<script src="MyJavascript.js"
  language="javascript"></script>
<body>
</body>
  <form>
        <input type="button"
  name="btMsg" value="ClickHere"
  onClick="DisplayMsg();">
   </form>
</html>
```

JavaScript file: MyJavascript.js

```
//Add your script
function DisplayMsg()
{
   alert("Text messge displayed on the Pop-
   up window is taken from external
   javascript file");
}
```

Advantages:

→Less code has to be written and stored.

→Commonly used javascript code can be shared among several pages.

→Only a single copy needs to be stored on the web server.

→Such javascript files can be cached by the browser, allowing faster page loading.

→The URL specified in the src attribute can also refer to another web server.

Contd.

→Language="JavaScript" is optional.

→For browsers that do not understand Javascript, it is possible to use the HTML comment tag "<!--" & "-->" to bracket out the Javascript code.

→ The marked block is treated as hidden by the browsers that do not understand Javascript.

→These tags are ignored by browsers that can interpret Javascript.

Contd.

language: This attribute specifies what scripting language you are using. Typically, its value will be *javascript*

type: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to *"text/javascript"*.

Example:

```
<script language="javascript" type="text/javascript">
        JavaScript code
</script>
```

General HTML page with JS

→ Javascript code snippet can be inserted anywhere in the HTML document.

```
<HTML>
    <HEAD>
        <TITLE>……..</TITLE>
        <SCRIPT LANGUAGE = "Javascript">
            <!-- java script code-->
        </SCRIPT>
    </HEAD>
     <BODY>……..</BODY>
</HTML>
```

HTML Code

JavaScript code

JavaScript Data Types

→JavaScript variables can hold many data types: numbers, strings, arrays, objects and more

Example:

```
→var length = 16;                                  // Number
→var lastName = "Johnson“;                         // String
→var cars = ["Saab", "Volvo", "BMW"];              // Array
→var x = {firstName:"John", lastName:"Doe"};    // Object
```

→JavaScript has dynamic types. This means that the same variable can be used as different types:

Example:

```
var x;               // Now x is undefined
var x = 5;           // Now x is a Number
var x = "John";      // Now x is a String
```

Javascript program:Example1

```
<html>
   <head>
         <title>javascript program demo</title>
   </head>
   <body>
         <script>
               document.writeln("<H1>Hello Good Day </H1>");
               document.writeln("<H1>Welcome to Javascript code</H1>");
         </script>
   </body>
</html>
```

Example2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <pre>
        <script language="javascript">
            var num1 = 10;
            var num2 = 20;
            var result = num1 + num2;
            document.writeln("Sum of" + "  " + num1 + "  " + num2 + " "
                    + "  " + "is" + "  " + result);
        </script>
    </pre>
</body>
</html>
```

JavaScript function

→A function is a group of reusable code which can be called anywhere in your program.

→ function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax:

```
<html>
<head>
    <title>Function Demo</title>
</head>
<body>
    <script type="text/javascript">
        function functionname(parameter-list)
        {
            statements
        }
    </script>
</body>
</html>
```

Calling function

```
<html>
<head>
<script type="text/javascript">
         function sayHello()
         {
            document.write ("Hello there!");
         }
      </script>

</head>
<body>
    <p>Click the following button to call the function</p>
    <form>
        <input type="button" onclick="sayHello()" value="Say Hello">
    </form>
    <p>Use different text in write method and then try...</p>
</body>
</html>
```

Exercise: Javascript to Add two numbers using the function

```
<!DOCTYPE html>
<html>
<head>
<title>Addition of two numbers</title>
<script type="text/javascript">
    function Addnumbers() {
        var num1 = parseInt(document.getElementById('txtnumber').value);
        var num2 = parseInt(document.getElementById('txtsecnumber').value);
        var result = num1 + num2;
        document.getElementById('result').innerHTML = result;
    }
</script>
</head>
<body>

    <form>
        <label>Enter first Number:</label> <input type="number" id="txtnumber">
        <label>Enter second Number:</label> <input type="number"
            id="txtsecnumber">
        <button type="button" id="btncal" onclick="Addnumbers()">Calculate</button>
        <p id="result"></p>
    </form>
</body>
</html>
```

Example2

```
<html>
    <body>
        <h1>My First JavaScript</h1>
        <button type="button“ onclick="document.getElementById('demo').innerHTML
          = Date()">
        Click me to display Date and Time.</button>
        <p id="demo"></p>
    </body>
</html>
```

Property	Content Type	HTML Tags	Use Case
innerText	Plain text	Ignored	Get or display raw text content
innerHTML	HTML content	Included	Modify element content with formatting

Example3:Use of functions & events

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Insert title here</title>
<script language="javascript">
    function HelloMessage()
    {
        var txt_name = document.getElementById('txtname').value;
        document.write("<h1>Hello!"+" "+ txt_name + " "+ "welcome to Javascript programming");
    }
</script>
</head>
<body>
<form>
    <label>Name:</label>
    <input type="text" id="txtname">
    <input type="button" id="btnsubmit" value="Submit" onclick="HelloMessage();">
</form>
</body>
</html>
```

Types of operators in JavaScript.

→JavaScript reserved words

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	finally	instanceof	throw	while
default	for	new	try	with

Types of operators in JavaScript.

→ Arithmetic Operators

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

Comparison Operators

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Bitwise & Logical Operators

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2
Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

Assignment Operators

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a+=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

Special Operators

Operator	Description
(?:)	**Conditional Operator returns value based on the condition. It is like if-else.**
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

JavaScript: Decision & looping statements

→JavaScript supports conditional statements which are used to perform different actions based on different conditions.

1. If Statement, If.. else statement, if.. else.. if statement
2. switch statement
3. Four types of loops in JavaScript.
    1. for loop
    2. While loop
    3. do-while loop
    4. for-in loop

Examples: if, if..else, if..else if

```
<html>
<head>
<title>IF Statement demo</title>
</head>

<body>
    Enter Value:
    <input type="number" id="txtnumber">
    <input type="button" id="btnSubmit" value="Submit"
        onclick="CompareValue();">
</body>
<script type="text/javascript">
    function CompareValue() {
        var txtinput = parseInt(document.getElementById("txtnumber").value);
        if (txtinput >= 35)
            alert("Entered value is:"
                    + parseInt(document.getElementById("txtnumber").value));
        else
            alert("Input value is less than passing score!!!!!!!!!!");
    }
</script>
</html>
```

JavaScript Global Variable

```
<!DOCTYPE html>
<html>
<head>
<title>Global Variable demo</title>
<script type="text/javascript">
    var PhoneNum = 123456789;
    function myfunction(num) {
        alert("My contact number is:" + " " + num);
        Appendnum(PhoneNum);
    }
    function Appendnum(PhoneNum) {
        alert("+91" + PhoneNum);
    }
</script>

</head>
<body>
    <form>
        <button type="button" id="btnsubmit" onclick="myfunction(PhoneNum)">Submit</button>
    </form>
</body>
</html>
```

Contd…

```
<script  type="text/javascript">
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

```
<script  type="text/javascript">
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
else{
document.write("a is not equal to 10, 15 or 20
");
}
</script>
```

Contd…

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

JavaScript: Loops

```
<script  type="text/javascript">
for (i=1; i<=5; i++)
{
            document.write(i + "<br/>")
}
</script>
```

```
<script  type="text/javascript">
var i=11;
while (i<=15)
 {
      document.write(i + "<br/>");
       i ++;
  }
</script>
```

```
<script  type="text/javascript">
var i=21;
do
   {
            document.write(i + "<br/>");
            i++;
   }while (i<=25);
</script>
```

The purpose of loop is to execute piece of code repeatedly until specified condition is satisfied.

for-in loop

Syntax:

```
for (variableName in Object) {
    statement(s);
}
```

for-in loop

Example:

```
<script type="text/javascript">
    var x;
    var items = new Array();
    items[0] = "512GB SSD";
    items[1] = "1TB USB card";
    items[2] = "Wireless Mouse";
    for (x in items)
     {
          document.write(items[x] + "<br />");
    }
</script>
```

Event Handling

→ An *event* is a notification that something specific has occurred, either in the browser, such as the completion of the loading of a document, or a browser user action, such as a mouse click on a form button.

→ An event handler is a script that is implicitly executed in response to the appearance of an event.

→ Event handlers enable a Web document to be responsive to browser and user activities.

→ Example: Check for simple errors and omissions in user input to the elements of a form

Event

→Event handlers are *JavaScript code that are not added inside the <script> tags, but rather, inside the html tags*, that execute JavaScript code when something happens, such as pressing a button, moving your mouse over a link, submitting a form etc.

→ Example:

```
<a href="http://sdmcet.ac.in" onclick="alert('hello!')">SDMCET</a>
```

→Event occurs when user or browser manipulates a page.

→Event handling normally occurs:

- → When page is loaded.
- → User clicks button.
- → Closing window.
- → resizing a window.

→ HTML elements contains set of events which trigger JavaScript code.

Contd…

→Most commonly used event handlers supported by JavaScript-

Attributes	Tag	Description
onblur	<a>	The link loses focus.
	<button>	The button loses focus.
	<input>	The input element loses focus.
	<textarea>	The text area loses focus.
	<select>	The selection element loses focus.
onchange	<input>	The input element is changed and loses focus.
	<textarea>	The text area is changed and loses focus.
	<select>	The selection element is changed and loses focus.
onclick	<a>	The user clicks the link.
	<input>	The input element is clicked.
ondblclick	Most elements	The user double-clicks the left mouse button.
onfocus	<a>	The link acquires focus.
	<input>	The input element acquires focus.

Contd…

→Most commonly used event handlers supported by JavaScript-

	<textarea>	A text area acquires focus.
	<select>	A selection element acquires focus.
onkeydown	<body>, form elements	A key is pressed.
onkeypress	<body>, form elements	A key is pressed and released.
onkeyup	<body>, form elements	A key is released.
onload	<body>	The document is finished loading.
onmousedown	Most elements	The user clicks the left mouse button.
onmousemove	Most elements	The user moves the mouse cursor within the element.
onmouseout	Most elements	The mouse cursor is moved away from being over the element.
onmouseover	Most elements	The mouse cursor is moved over the element.
onmouseup	Most elements	The left mouse button is unclicked.
onreset	<form>	The reset button is clicked.
onselect	<input>	Any text in the content of the element is selected.
	<textarea>	Any text in the content of the element is selected.
onsubmit	<form>	The *Submit* button is pressed.
onunload	<body>	The user exits the document.

Example: Onclick event

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<script type="text/javascript">
    function MyJSfuntion() {
        alert("Hello! This function is called when user clicks button");
    }
</script>
</head>
<body>
    <button type="button" onclick="MyJSfuntion();">TriggerJSFun</button>
</body>
</html>
```

onmouseover and onmouseout

```
<html>
<head>

<script type="text/javascript">
    function over() {
        document.write("Mouse Over");
    }

    function out() {
        document.write("Mouse Out");
    }
</script>

</head>
<body>
    <p>Place your mouse pointer on below mentioned text message</p>

    <div onmouseover="over()" onmouseout="out()">
        <h2>This Text message is replaced once mouse pointer is placed
            over here</h2>
    </div>

</body>
</html>
```

Sample events in JavaScript's

Attribute	Meaning
onblur	Triggers when the window loses focus
onchange	Triggers when an element changes
ondblclick	Triggers on a mouse double-click
ondrop	Triggers when dragged element is being dropped
onfocus	Triggers when the window gets focus
onkeydown	Triggers when a key is pressed
onkeypress	Triggers when a key is pressed and released
onkeyup	Triggers when a key is released
onkeyup	Triggers when a key is released

Use of 'this' in JavaScript

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Retrieval of TextBox values using object</title>
</head>
<body>
    <form>
        <label>Name:</label> <input type="text" id="txtname"> <label>USN:</label>
        <input type="text" id="txtusn"> <label>Semester:</label> <input
            type="text" id="txtsem"> <input type="button" id="btnsubmit"
            value="Submit" onclick="ReadValues(this.form)">
        <div id="mydiv"></div>
    </form>
</body>
<script type="text/javascript">
    function ReadValues(obj) {
        document.getElementById('mydiv').innerHTML = 'Name:'
                + obj.txtname.value + '</br>' + 'USN:' + obj.txtusn.value
                + '</br>' + 'Semester:' + obj.txtsem.value;
    }
</script>
</html>
```

document.write(myform. txtname.value);

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

Mode of form submission

1. Submission by generating an event using an “onsubmit” attribute in form tag.
2. Direct submission
    i. No script execution
    ii. No validation

Onsubmit event

```
<html>
<head>

<script type="text/javascript">
        <!--
           function validation() {
              all validation goes here
              .........
              return either true or false
           }
        //-->
      </script>

</head>
<body>

    <form method="POST" action="t.cgi" onsubmit="return validate()">
        ....... <input type="submit" value="Submit" />
    </form>

</body>
</html>
```

Confirm Box

```
<body>
        <script>
            function respond()
            {
                    confirm("Are you sure want to continue!");
            }
         </script>
           <H2>Click on the following button</h2>
         <form>
              <table>
                    <tr>
                          <td >
                     <input style="width:140; font-size:18pt;" type="button"  value="Click here" onClick="respond()">
                           </td>
                    </tr>
                 </table>
           </form>
      </body>
```

Confirm Box: properties

→The confirm() method returns a boolean value.

1. “true” if “OK” is pressed
2. “false” if “Cancel” is pressed

→The return value can be used in decision logic.

Example

```
<script>
      function GotoPage()
       {
           var v = confirm("Want to visit SDMCET site?");
           if (v==true)
          window.location="http://www.sdmcet.ac.in";
          else
              window.location="http://www.google.com";
           }
</script>
```

```
<form>
     <table>
          <tr>
               <td >
<input style="width:140; font-
size:18pt;" type="button"
value="Click to Go"
onClick="GotoPage()">
               </td>
          </tr>
     </table>
</form>
```

Javascript Alerts

→A Javascript alert box, when clicked, displays a text, and waits until the visitor presses the “OK” button.

→Syntax:

alert(“Any Message to be displayed.”);

Example: Alert

```
<script>
        function GotoPage()
        {
                alert("Hello!");
                alert("Good Day!");
                 window.location = "http://www.sdmcet.ac.in";
        }
</script>
<form>
      <table>
            <tr>
                <td >
            <input style="width:140; font-size:18pt;" type="button"  value="Click Here" onClick="GotoPage()">
                 </td>
              </tr>
          </table>
</form>
```

Background color

→Modify the bgColor attribute of the document class

→ An example:

→ On mouse click, the background color toggles between blue, green, and red.

Example4: Manipulating property in JS

```
<script>
        function Change_color()
        {
        if(document.bgColor=="#0000ff")
            document.bgColor = "#00ff00";
        else if(document.bgColor == "#00ff00")
             document.bgColor = "#ff0000";
        else
           document.bgColor = "#0000ff";
        }
</script>
```

```
<form>
    <table>
        <tr>
          <td >
<input style="width:140; font-size:18pt;" type="button"
value="Click Here" onClick="Change_color();">
          </td>
          </tr>
         </table>
</form>
```

Example 5:Alert

```
<script>
   function respond()
   {
            alert("Hello, Welcome to scripting language");
            alert("Your are watching pop up window");
            window.location="http://www.sdmcet.ac.in";
    }
</script>
 <form>
            <input type="button" value="Click Here" onClick="respond();">
 </form>
```

Example 5

```
<script>
        function Change_color(f)
        {
      a=confirm("Are you coming to college
        today?");
   if(a)
        f.option.value="Yes coming";
    else
        f.option.value="Not coming";
        }
</script>
```

```
<form>
    <table>
        <tr>
            <td >
        <input style="width:140; font-size:18pt;"
type="button"  value="Click Here"
onClick="Change_color(form);">
        <input style="width:140; font-size:18pt;"
type="text"  name="option">
            </td>
        </tr>
    </table>
</form>
```

Example6: Mathematical Calculation

→This example will show how to:

→Extract data from a form field into a variable.

→Perform mathematical calculations on that variable.

→Store the result back into a form field.

Example: temperature conversion

```
<script>
function Change_Temp(myform)
{
var
cent=parseFloat(myform.option.
value);
fahr = (cent * 1.8) + 32;
myform.Fahr.value = fahr;
}
</script>
```

```
<form>
   <table>
         <tr>
          <td>
          Centigrade:<input style="width:140; font-size:18pt;"
type="text"  name="option">
          </td>
<td >
<input style="width:140; font-size:18pt;" type="button"
value="Convert" onClick="Change_Temp(this.form);">
</td>
<td>
Fahrenheit:<input style="width:140; font-size:18pt;"
type="text"  name="Fahr">
                                        </td>
                                  </tr>
                            </table>
                      </form>
```

How to access form data

→This example will show how to:

→Extract the individual selected fields from a drop-down menu.

→Use the selected value for further processing.

-- Here the value is simply echoed back through an alert box.

Example

```
<script>
function Check_Color(f) {
if (f.favcolor[0].selected)
alert("favourite color is:" +" " +
    f.favcolor[0].value);
if (f.favcolor[1].selected)
alert("favourite color is" +" " +
    f.favcolor[1].value);
if (f.favcolor[2].selected)
alert("favourite color is" +" " +
    f.favcolor[2].value);
if (f.favcolor[3].selected)
alert("favourite color is" +" " +
    f.favcolor[3].value);
}
</script>
```

```
<form>
<label>My favourite color:</label>
<select name="favcolor">
      <option>Blue</option>
      <option>Red</option>
      <option>Black</option>
      <option>Green</option>
      </select>
      <input type="button" value="done"
      onClick="Check_Color(this.form)">
</form>
```

Form Validation Using JavaScript

Form validation is done to check the accuracy of the user's entered information before they could submit the form.

Form/Control validation

→Javascript can be used for client-side validation

→Data entered can be extracted and accessed.

→Checks can be made on the data entered.

-- Non-alphabetic characters in a name.

-- Non-numeric characters in roll number, age, etc.

--Example:

-- To check the age limit for particular positions.

Primitive type values-'null' & 'undefined'

→JavaScript includes two additional primitive type values

1. 'null'
2. 'undefined'

→**null** - A null means the absence of a value.

→A null value evaluates to false in the conditional expression.

→Does not apply to.

→Unknown.

→The value is known but absent

→ **undefined** – undefined means lack of value or unknown value.

→An undefined evaluates to false when used in the conditional expression.

→**NaN** is raised when a variable is assigned other than the number.

Example 1:Control validation

```
<html>
<body>
   <script>
   function checkvalue() {
   var mystring = document.getElementById('myString').value;
   if(mystring == null || mystring == "") {
      alert ('Empty value is not allowed');
      return false;
   }
```

Contd.

```
else {
        alert("correct input");
        return true;
    }
}
</script>
</body>
<form onsubmit="checkvalue()">
    <input name="myString" type="text" value='' id="myString">
    <input type="submit" value="ClickTosubmit" />
</form>
</html>
```

Data Validation: JavaScript RegExp

→ RegExp Object

I. A regular expression is an object that describes a pattern of characters.

II. Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

In JavaScript, a Regular Expression (RegEx) is an object describing a sequence of characters used to define a search pattern.

Syntax: /*pattern*/*modifiers*;

/sdm/i (i-modifies the search to be case-insensitive)

Data Validation: JavaScript RegExp

→ regexp. search() – looks for the position of the match,

→ regexp. match() – if there's no g flag, returns the first match with parentheses and all details,

→ regexp. match() – if there's a g flag – returns all matches, without details parentheses,

→ regexp. matchAll() – returns all matches with details.

→regexp. exec() –

-If there's no g, then exec(str) returns the first match

-exec(str) returns the first match and *remembers* the position after it in regexp.lastIndex property.

→regexp.test(str) - The method regexp.test(str) looks for a match and returns true/false whether it finds it.

→str.split(regexp|substr, limit) - Splits the string using the regexp (or a substring) as a delimiter.

→str.replace(str|reg, str|func) - searching and replacing a substring

Regular expression pattern

→ "^" (caret): When "^" is used at the beginning of a regular expression pattern, it signifies the start of the string.

→ It indicates that the pattern must match at the beginning of the string.

→ "$" (dollar sign): When "$" is used at the end of a regular expression pattern, it signifies the end of the string.

→ It indicates that the pattern must match at the end of the string.

Expression	Description
/^abc/	match any string that starts with "abc"
/cse$/	match any string that ends with "**cse**".
/^sdmcet$/	match only the string "**sdmcet**" exactly, with nothing before or after it.

Contd…

Brackets are used to find a range of characters

Expression	Description
[abc]	Find any character between the brackets
/^abc/	Find any character NOT between the brackets
/[0-9]/	Find any digit between the brackets
[^0-9]	Find any digit NOT between the brackets
(x|y)	Find any of the alternatives specified
/^[a-zA-Z\s]*$/	Allows only alphabets and spaces

^[2-9]\d{2}-\d{3}-\d{4}$ 800-555-5555 | 333-444-5555 | 212-666-1234

000-000-0000 | 123-456-7890 | 2126661234

(\w[-._\w]*\w@\w[-._\w]*\w\.\w{2,3}) foo@bar.com | foobar@foobar.com.au

foo@bar | $$$@bar.com

Contd...

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character

Contd…

\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

Example: Name field validation

```
<!DOCTYPE html>
<html>
<head>
    <title>control validation</title>
</head>
<body>
    <form>
        <input type="text" id="txtinput">
        <input type="button" value="submit" onclick="validatemyform();">
    </form>
</body>
<script type="text/javascript">
    function validatemyform(){
        var patrn = '\w';
        var val = document.getElementById('txtinput').value;
        if(val.match(patrn)){
            alert('matched successfully')
        }
        else{
            alert('not matched....');
        }
    }
</script>
</html>
```

E-mail validation

```
<!DOCTYPE html>
<html>
<head>
    <title>control validation</title>
</head>
<body>
    <form>
        <input type="text" id="txtinput">
        <input type="button" value="submit" onclick="validatemyform();">
    </form>
</body>
<script type="text/javascript">
    function validatemyform(){
        var patrn = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/
        var val = document.getElementById('txtinput').value;
        if(val.match(patrn)){
            alert('matched successfully')
        }
        else{
            alert('not matched....');
        }
    }
</script>
</html>
```

Example: Data validation with 'RegExp' constructor & 'test' function

```
<!DOCTYPE html>
<html>
<head>
    <title>control validation</title>
</head>
<body>
    <form>
        <input type="text" id="txtinput">
        <input type="button" value="submit" onclick="validatemyform();">
    </form>
</body>
<script type="text/javascript">
    function validatemyform(){
        var patrn = new RegExp("sdm");
        var val = document.getElementById('txtinput').value;
        if(patrn.test(val)){
            alert('matched successfully')
        }
        else{
            alert('not matched....');
        }
    }
</script>
</html>
```

Example

Sl.No	Input type	RegEx/pattern
1	Phone Number	/^\d{10}$/
2	+XX-XXXX-XXXX +XX.XXXX.XXXX +XX XXXX XXXX	/^\+?([0-9]{2})\)?[-.]?([0-9]{4})[-.]?([0-9]{4})$/
3	Name	/^[A-Za-z]+$/
4	ZipCode	/^[0-9]+$/
5	IP Address	/^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/
6	Email	/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/

Javascript - Arrays

Introduction

→JavaScript arrays are used to store multiple values in a single variable.

Syntax:

Creating a Array object:

var fruits = new Array("apple", "orange", "mango");

→You can create array by simply assigning values as follows:

var fruits = ["apple", "orange", "mango"];

→ The ordinal numbers are used to access and to set values inside an array as follows:

fruits[0] is the first element

fruits[1] is the second element

fruits[2] is the third element

Avoid new Array()

→There is no need to use the JavaScript's built-in array constructor new Array(). Use [] instead.

→These two different statements both create a new empty array named points:

Example:- Syntax of new array creation

```
var points = new Array();        // Bad
var points = [];                 // Good
```

→These two different statements both create a new array containing 6 numbers:

Example:-

```
var points = new Array(40, 100, 1, 5, 25, 10)  // Bad
var points = [40, 100, 1, 5, 25, 10];          // Good
```

Example:Index

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
var person = [];
person[0] = "John";
person[1] = "Doe";
person[2] = 46;
document.getElementById("demo").inne
rHTML =person[0] + " " +
person.length;
</script>

</body>
</html>
```

```
<html>
<body>
<p>
If you use a named index, when accessing an array,
JavaScript will redefined the array to a standard object,
and all array methods and properties will produce
undefined or incorrect results.
</p>

<p id="demo"></p>

<script>
var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
document.getElementById("demo").innerHTML =
person[0] + " " + person.length;
</script>

</body>
</html>
```

Contd.

→The new keyword complicates your code and produces nasty side effects:

Example:-

```
// Creates an array with two elements (40 and 100)
        var points = new Array(40, 100);
```

What if I remove one of the elements?

```
// Creates an array with 40 undefined elements !!!!!
        var points = new Array(40);
```

Contd…

→ Reason: when the user passes a single numeric parameter to the Array constructor, It will treat it as the initial length of the array, instead of the array element of the first item.

Example:Array

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>JS Array type Demo</title>
</head>
<body>
    <pre>
    <script type="text/javascript">
        var a = [];
        var b = new Array();
        var c = [ "sdmcet", "darwad" ];
        var e = [ 3 ];
        var f = new Array(3);
        document.writeln(a.length + " " + b.length);
        document.writeln(c.length + " " + e.length + " " + c[0] + " " + c[1]
                + "  " + e[0]);
        document.writeln(f.length + " " + f[0]);
    </script>
    </pre>
</body>
</html>
```

Array Methods

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
map()	Creates a new array with the results of calling a provided function on every element in this array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
reduceRight()	Apply a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value.
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
some()	Returns true if at least one element in this array satisfies the provided testing function.
toSource()	Represents the source code of an object
sort()	Sorts the elements of an array.
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

Example:concat()

```
<html>
    <head>
        <title>JavaScript Array concat Method</title>
    </head>
    <body>
        <script type="text/javascript">
          var alpha = ["a", "b", "c"];
          var numeric = [1, 2, 3];

          var alphaNumeric = alpha.concat(numeric);
          document.write("alphaNumeric : " + alphaNumeric );
        </script>
     </body>
</html>
```

Example: Array constructor

```
<html>
  <head>
        <title> List most popular programming languages</title>
  <script>
     var lang=new Array(3);
      lang[0] = "C programming language";
         lang[1] = "OO programming language";
      lang[2] = "Java programming language";
  </script>
   </head>
<body>
        <script>
        var num = lang.length
    document.write("<h2> Hierarchy of programming language</h2>" +"<p>");
    for(i=0;i<num; i++)
        document.writeln("hierarchy"+ (i+1) +":"+ lang[i] +"<p>");
    </script>
    </body>
</html>
```

Example: Array with push & pop

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<script type="text/javascript">
        var numbers =[ 11, 44, 99 ];

        var length = numbers.push(10);
        document.write("new numbers is added: " + numbers );

        length = numbers.push(20);
        document.write("<br />new numbers is added : " + numbers );
    </script>
</body>
</html>
```

Example: Array with push & pop

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <script type="text/javascript">
        var numbers = [ 11, 44, 99 ];

        var element = numbers.pop();
        document.write("Poped element is : " + element);

        var element = numbers.pop();
        document.write("<br />Poped element is : " + element);
    </script>
</body>
</html>
```

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <fieldset>
        <legend>Student Record</legend>
        <h3>
            How to Add values and loop through array in JavaScript
        </h3>
        <!-- form post method -->
        <form action="" method="post">
            <label>Enter Value</label> <input type="text" id="txtname" />
            <input type="button" value="Add to Array" onclick="addArray()" /> <br />
            <br /> <input type="button" value="Show Array" onclick="showArray()" />
        </form>
        Name<div id="showResult"></div>
    </fieldset>
</body>
```

Contd…

```
<script type="text/javascript">
    //define an array variable
    var arr = [];

    //this function helps to push values into array
    function addArray() {
        //get the value of the textbox
        var inputVal = document.getElementById('txtname').value;

        //check if input value is not empty, then allow to push value
        if (inputVal != '') {
            //using default JavaScript push() method to add the value into an array
            arr.push(inputVal);
            alert('Record Inserted Successfully');
            document.getElementById('txtname').value = "";
        }
    }
```

Contd…

```
//this function helps to print the array values in a div
function showArray() {
    //find array length
    var len = arr.length;
    //clear the div 'showResult'
    document.getElementById('showResult').innerHTML = '';
    //check if the length of the array is greater than 0 else alert user
    if (len > 0) {
        //for loop
        for (i = 0; i < arr.length; i++) {
            document.getElementById('showResult').innerHTML += arr[i]
                    + "<br>";
        }
    } else {
        alert("No Array Values, Please add using the above text box");
    }

}
```

JavaScript Strings

Introduction

→The **String** object let's you work with a series of characters and wraps JavaScript's string primitive data type with a number of helper methods

Syntax:

Creating a **String** object:

```
var val = new String(string);
```

String Methods:

Method	Description
charAt()	Returns the character at the specified index.
charCodeAt()	Returns a number indicating the Unicode value of the character at the given index.
concat()	Combines the text of two strings and returns a new string.
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
localeCompare()	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
match()	Used to match a regular expression against a string.
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
search()	Executes the search for a match between a regular expression and a specified string.
slice()	Extracts a section of a string and returns a new string.
split()	Splits a String object into an array of strings by separating the string into substrings.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
substring()	Returns the characters in a string between two indexes into the string.
toLocaleLowerCase()	The characters within a string are converted to lower case while respecting the current locale.
toLocaleUpperCase()	The characters within a string are converted to upper case while respecting the current locale.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.

String:length

→This property returns the number of characters in the string.

Syntax:→

string.length

```
<html>
<head>
    <title>JavaScript String length Property</title>
</head>
<body>
<script type="text/javascript">
  var str = new String( "This is string" );
  document.write("str.length is:" + str.length);
</script>
</body>
</html>
```

Strings: charAt()

→ This method returns the character from the specified index.

Characters in a string are indexed from left to right.

The index of the first character is 0, and the index of the last character in a string called stringName is stringName.length - 1.

Return Value:

Returns the character from the specified index

Example: charAt()

```
<html>
        <head>
            <title>JavaScript String charAt() Method</title>
        </head>
        <body>
            <script type="text/javascript">
              var str = new String( "This is string" );
              document.writeln("str.charAt(0) is:" + str.charAt(0));
              document.writeln("<br />str.charAt(1) is:" + str.charAt(1));
              document.writeln("<br />str.charAt(2) is:" + str.charAt(2));
              document.writeln("<br />str.charAt(3) is:" + str.charAt(3));
              document.writeln("<br />str.charAt(4) is:" + str.charAt(4));
              document.writeln("<br />str.charAt(5) is:" + str.charAt(5));
            </script>
        </body>
</html>
```

Browser Detection

→This feature allows one to find out what type of browser is being used.

→The navigator object contains information about the browser.

Browser Detection

Property	Description
appCodeName	Returns the code name of the browser
appName	Returns the name of the browser
appVersion	Returns the version information of the browser
cookieEnabled	Determines whether cookies are enabled in the browser
geolocation	Returns a Geolocation object that can be used to locate the user's position
language	Returns the language of the browser
onLine	Determines whether the browser is online
platform	Returns for which platform the browser is compiled
product	Returns the engine name of the browser
userAgent	Returns the user-agent header sent by the browser to the server

Example2

```
<script>
    var browserName = navigator.appName;
    var browserVersion = navigator.appVersion;
if(browserName == “Netscape”)
            alert(“Hello, Netscape User! ” + browserVersion );
else
    if (browserName == “Microsoft Internet Explorer”)
        alert(“Hello, IE User!” + browserVersion );
    else
            alert(“Unrecognized browser!”);
</script>
```

Or

```
alert(window.navigator.appName );
```

Example: Page redirection

```
<html>
 <head>
  <title>Page Redirection</title>
 </head>
 <body>
  <script>
        function RedirectTo()
        {
          window.location = "http://www.sdmcet.ac.in";
        }
        alert("You will be redirected in five seconds");
        setTimeout('RedirectTo()', 5000);
  </script>
  </body>
</html>
```

Creating new browser window

→To open a new window, we need to call the window.open() method

→ window.open('URL to open', 'window name', attribute1, attribute2,……);

→ Window attributes:

→ width = 300

→ height = 200

→ resizeable = yes or no

→ scrollbars = yes or no

→ toolbars = yes or no

→Status = yes or no

→ menubar = yes or no

→ copyhistory = yes or no

Example: new window

```
<html>
 <head>
  <title>Page Redirection</title>
 </head>
 <body>
 </body>
  <form>
  <input type="button" value="new window" onClick
  ="window.open('http://www.sdmcet.ac.in','mywindow','width=400','height=200','toolbar=yes'
  , 'status=yes')">
</form>
</html>
```