UNIT 4: JAVASCRIPT

Features of JavaScript, extension of JavaScript, Syntax of JavaScript: data types, operators, variables, tag, Document Object Model (DOM) with JavaScript, Selection Statement using if and Switch, Iterative statement: for, for/in, while, do while, break and continue

What is JavaScript?

- JavaScript was once upon a time used only in *client* side (browser).
- But *node js* (execution engine/run time/web server) have made possible to run JavaScript on server side.
- JavaScript is everywhere – on Desktop/Server/Mobile.

Introduction

- It is a client side scripting language.
- It is basically used for validations.
- It was developed by Netscape Communications.
- JavaScript is object based language not object oriented language.
- JavaScript is embedded in the web pages and interpreted by the web browser.

Advantages and Disadvantages of client side scripting

- Advantages:
  - The Web browser uses its own resources so less load on the server.

- Disadvantages
  - Code is usually visible
  - Local files and databases cannot be accessed as they are located on to the server..

JavaScript Features

- Light Weight Scripting language
- Dynamic Typing
- Object-oriented programming support
- Functional Style
- Platform Independent
- Prototype-based
- Interpreted Language
- Async Processing
- Client-Side Validation
- More control in the browser

Where to place JavaScript

- Embedded Script
- JavaScript can be embedded in the HTML document.
- It begins with

  <script type="text/javascript>

  - and ends with

  </script>
- You can also write

  <script language="Javascript">

  - And end with

  </script>

JavaScript example

```
<html>
  <head>
     <title>A small piece of JavaScript</title>
  </head>
  <body>
     <script type="text/javascript">
          document.write("hello, JavaScript user!");
     </script>
  </body>
</html>
```

- **Javascript can also be placed in the head section of HTML document.**

External Scripts

- If you want to use same script on several pages, it is better to create a script in the separate file.

- For eg:

```
<html>
  <head>
     <script src="myscript.js">
     </script>
  </head>
<html>
```

myscript.js

```
document.write("hello, JavaScript user!");
```

The external js file can not use <script>

JavaScript Display Possibilities

- **JavaScript can "display" data in different ways:**

- Writing into an HTML element, using **innerHTML**.
- Writing into the HTML output using **document.write()**.
- Writing into an alert box, using **window.alert()**.
- Writing into the browser console, using **console.log()**.

Using innerHTML

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

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

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

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Using window.alert()

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

  <h1>My First Web Page</h1>
  <p>My first paragraph.</p>

  <script>
  window.alert(5 + 6);
  </script>

  </body>
  </html>
  ```
- **You can skip the window keyword**

Using console.log()

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

  <script>
  console.log(5 + 6);
  </script>

  </body>
  </html>
  ```

Check LOG:

1. F12 (or inspect) on your keyboard to activate debugging.
2. Then select "Console" in the debugger menu.
3. Click Run again.

Variables

- The primitive data types in JS are as follows:
  1. number
  2. string
  3. boolean
  4. null
  5. function
  6. object
- Some conditions for variables:
  1. It is case sensitive.
  2. It cannot contain punctuations, spaces etc.
  3. It should not be reserved word.

Scope of the variable

- A JS variable can either have local or global scope.
- A variable is declared with the var keyword.

```
var  a,b;
```

- The above variables if defined outside function, have a global scope.
- If a variable is defined within the function, it has the local scope.

An Example

```
<html>
<body>
<script type="text/javascript">
        var a= "abc";
        test();
        document.write(a, "<br>");
        test();
        document.write(c, "<br>");
        document.write(b, "<br>");

                function test()
                {
                        var a="XYZ";
                        var b="def";
                        var c="pqr";
                        document.write(a, "<br>");
                }
</script>
</body>
</html>
```

Here variable **a** is defined globally so it is accessible.

Again three more variables **a,b** and **c** defined in function with local scope

The output is

```
XYZ
abc
XYZ
```

Variable: Assignments

- Assignment is used with the help of “=“ operator.
  - X=123;
  - X=“abc”;
  - X=X+1;
  - X++;
  - X--;
  - X+=y;
  - X-=y;

Strings

- To **define** strings we can use

  var a="abc";

  a="abc";

- To **concat** strings we can use + operator

  a ="abc"+"def";

  b= "def"+a;

- To find the **length**

  length=a.length;

- To find a **particular** character

  nchar=a.charAt(5);

Arrays

- There are 3 ways to construct array in JavaScript
  - By array literal
  - By creating instance of Array directly (using new keyword)
  - By using an Array constructor (using new keyword)

- EX (using literal):

```
<script>
var emp=[“One",“Two",“Three"];
for (i=0;i<emp.length;i++) {
  document.write(emp[i] + "<br/>");
  }
</script>
```

- EX (using constructor):

```
var myarray=new Array();
var myarray= new Array(1,2,”three”,false);
var myarray=new Array(15);
```

Arrays: Example

Using 'new' Keyword

```
<script type="text/javascript">
var i;
var myArray = new Array(3);
myArray[0] = “NMIMS";
myArray[1] = “University";
myArray[2] = “Navi Mumbari";

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

Comment Style in HTML

```
<! --
      insert your comment here….
-->
```

Using Constructor

```
<script>
var emp=new Array(“ABC",“PQR",“XYZ");

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

Comment Style in JS:

Same as C Language

JavaScript: Array Methods
For Examples, visit https://www.javatpoint.com/javascript-array

concat()	It returns a new array object that contains two or more merged arrays.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

Methods used with character arrays

1. charAt
2. Indexof
3. lastIndexOf
4. substring
5. valueOf
6. toLowerCase
7. toUpperCase

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

//charAt  finds the position in a string
var str = "HELLO NMIMS";
var res = str.charAt(0);
document.write("char at 0: ",res);

//indexOf variable in string..
var str = "Hello, welcome to NMIMS ";
var n1 = str.indexOf("welcome");
document.write("<br>index of: ",n1);

//lastIndexOf a charater in string
var str = "Hello NMIMS Friends";
var n2 = str.lastIndexOf("e");
document.write("<br>last index of: ",n2);

//substring from a string
var str = "Hello world!";
var res = str.substr(1, 4);
document.write("<br>substring: ",res);
```

Output

```
1.   char at 0: H
2.   index of: 7
3.   last index of: 15
4.   substring: ello
5.   valueof: Hello World!
6.   lower case: hello world!
7.   Upper case: HELLO WORLD!
```

Functions

- A function is a section of code that is separated from the main program.

```
function abc
{
        Document.write(“hello”);
}
abc();
```

Functions: Example

```
<html>
<body>
  <script type="text/javascript">
  var z = multXbyY(10, 15);
  document.write("<br>The result is "+z);
  function multXbyY(x,y) {
      document.write("x is "+x);
      document.write("<br>y is "+y);
      return x*y;
  }
</script>
</body>
</html>
```

Conditions

if –else

```
if(a==0)
{
  …
}
else
{
  …
}
```

Switch Case

```
<html> <body>
<script type="text/javascript">
   var m=new Date();
   theMonth=m.getMonth();
   document.write(theMonth);
   switch (theMonth)
   {
       case 4:  document.write("The Merry Month of May");
              break;
       case 0:  document.write("Cold January!");
              break;
       case 6:  document.write("Summer Time!");
              break;
       default:  document.write("When is it holiday time?");
   }
</script> </body> </html>
```

Loops

```
for(i=0;i<10;i++)
{
        ….
}
```

```
while(i>0)
```

The “for…in” loop

- JavaScript supports different kinds of loops:
  - for - loops through a block of code a number of times
  - for/in - loops through the properties of an object

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
  text += person[x] + " ";
}
```

  - More on objects in next session…

OBJECTS IN JS

Objects are Variables

- Objects are variables too. But objects can contain many values.
- Object values are written as name : value pairs

  let person = {firstName:“A”, lastName:“B”, age:50, eyeColor:"blue"};

```
<script>
let person = {
  firstName      : “A”,
  lastName       : “B”,
  age            :  50,
  eyeColor       : "blue”
};

document.getElementById("demo").innerHTML = person.firstName + " " + person.lastName;
</script>
```

A JavaScript object is a collection of **named values**

The named values, in JavaScript objects, are called **properties (e.g firstname)** the value of **firstname** property is **“A”** in above example

let keyword

- The **let** keyword was introduced in ES6 (2015).
- Variables defined with **let** cannot be Re-declared.
- Variables defined with **let** must be Declared before use.
- Variables defined with **let** have Block Scope.

- It is a common practice to declare objects with the const keyword.

const person = {firstName:“A”, lastName:“B”, age:50, eyeColor:"blue"};

JavaScript Objects are Mutable

- Objects are **mutable**: They are addressed by **reference**, not by **value**.

  const x = person; // Will not create a copy of person.

- The object x is **not a copy** of person. It **is** person. Both x and person are the same object.
- Any changes to x will also change person, because x and person are the **same** object.

```
const person = {
  firstName:“A",
  lastName:“B",
  age:50, eyeColor:"blue"
}

const x = person;
x.age = 10;      // Will change both x.age and person.age
```

Objects

- A javaScript object is an entity having state and behavior
  - properties and method
- JavaScript is an object-based language.
  - Everything is an object in JavaScript.
- JavaScript is template based not class based.
  - Here, we don't create class to get the object. But, we directly create objects.

- **Math Object (Inbuilt Object)**

```
Math.PI  is property
document.write(Math.PI);
value=Math.round(10.2);
```

Nested Objects

```
myObj = {
 name:"John",
 age:30,
 cars: {
   car1:"Ford",
   car2:"BMW",
   car3:"Fiat"
 }
}
```

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

<script>
const myObj = {
  name: "John",
  age: 30,
  cars: {
     car1: "Ford",
     car2: "BMW",
     car3: "Fiat"
     }
}
document.getElementById("demo").innerHTML =
myObj.cars.car2;
</script>
```

Using new keyword to create objects

```
<script type="text/JavaScript">
    var  person = new Object();
    person.firstName = "Jane";
    person.lastName = "Smith";
    person.age = 32;

    person.hair = new Object();
    person.hair.length = "long";
    person.hair.colour = "red";

    document.write(person.firstName+" "+person.lastName+" "+person.age);
    document.write("</br>");
    document.write(person.hair.length+" "+person.hair.colour);
</script>
```

DOM (Document Object Model)

- The base class in JavaScript is the window object.
- Whenever we write

```
document.write(“hello”);
```

- We are actually writing

```
window.document.write(“hello”);
```

- The window object is there by default.

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:

  "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

- The W3C DOM standard is separated into 3 different parts:
  - Core DOM - standard model for all document types
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents

What is the HTML DOM?

- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements
- In other words:

The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

The following example changes the content (the innerHTML) of the <p> element with id="demo"

```
<html>
<body>

<p id="demo">This is SVMIT College</p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

JavaScript HTML DOM - Changing CSS

```
<html> <body>
<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

<button type="button"
onclick="document.getElementById('id1').innerHTML = 'Now, Not Allowed
!!!'">
Ohh No!</button>

<button type="button"
onclick="document.getElementById('id1').style.color = 'black'">
Black?</button>

<button type="button"
onclick="document.getElementById('id1').innerHTML = 'My Heading 1'">
Origional!</button>
</body> </html>
```

Finding HTML Elements by HTML Object Collections

```
<html> <body>
<form id="frm1" action="/action_page.php">
  First name: <input type="text" name="fname" value="NMIMS"><br>
  Last name: <input type="text" name="lname" value="NM"><br><br>
  <input type="submit" value="Submit">
</form>
<p>Click "Try it" to display the value of each element in the form.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var x = document.forms["frm1"];
  var text = "";      var i;
  for (i = 0; i < x.length ;i++) {
    text += x.elements[i].value + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
}
</script> </body> </html>
```

JavaScript Errors - Throw and Try to Catch

```
<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;
  }
}              </script>  </body>  </html>
```

JavaScript Debugging

```
<html> <body>

<h2>My First Web Page</h2>

<p>Activate debugging in your browser (Chrome, IE, Firefox) with
F12, and select "Console" in the debugger menu.</p>

<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>    </body>    </html>
```

JAVASCRIPT VALIDATIONS

Validate Numeric Input

```
<html>
<body>

<h2>JavaScript Can Validate Input</h2>

<p>Please input a number between 1 and 10:</p>

<input id="numb">

<button type="button" onclick="myFunction()">Submit</button>

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

<script>
function myFunction() {
  var x, text;



 // Get the value of the input field with id="numb"
 x = document.getElementById("numb").value;
```

Automatic HTML Form Validation

```
<html>
<body>

<form action="/action_page.php" method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>

<p>If you click submit, without filling out the text field,
your browser will display an error message.</p>

</body>
</html>
```

HTML Constraint Validation

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

```
<html><body>
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
</body></html>
```

Constraint Validation DOM Methods

```
<html> <body>
<p>Enter a number and click OK:</p>
<input id="id1" type="number" min="100" max="300" required>
<button onclick="myFunction()">OK</button>

<p>If the number is less than 100 or greater than 300, an error message will be
displayed.</p>
<p id="demo"></p>

<script>
function myFunction() {
  var inpObj = document.getElementById("id1");
  if (!inpObj.checkValidity()) {
    document.getElementById("demo").innerHTML = inpObj.validationMessage;
  } else {
    document.getElementById("demo").innerHTML = "Input OK";
  }
}
</script> </body> </html>
```

Constraint Validation DOM Properties

Property	Description
validity	Contains boolean properties related to the validity of an input element.
validationMessage	Contains the message a browser will display when the validity is false.
willValidate	Indicates if an input element will be validated.

Validity Properties

Property	Description
customError	Set to true, if a custom validity message is set.
patternMismatch	Set to true, if an element's value does not match its pattern attribute.
rangeOverflow	Set to true, if an element's value is greater than its max attribute.
rangeUnderflow	Set to true, if an element's value is less than its min attribute.
stepMismatch	Set to true, if an element's value is invalid per its step attribute.
tooLong	Set to true, if an element's value exceeds its maxLength attribute.
typeMismatch	Set to true, if an element's value is invalid per its type attribute.
valueMissing	Set to true, if an element (with a required attribute) has no value.
valid	Set to true, if an element's value is valid.

The rangeOverflow Property

```
<html> <body>
<p>Enter a number and click OK:</p>
<input id="id1" type="number" max="100">
<button onclick="myFunction()">OK</button>
<p>If the number is greater than 100 (the input's max attribute), an error message
will be displayed.</p>
<p id="demo"></p>

<script>
function myFunction() {
  var txt = "";
  if (document.getElementById("id1").validity.rangeOverflow) {
    txt = "Value too large";
  } else {
    txt = "Input OK";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script> </body> </html>
```

JavaScript Form Validation Example

```
<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 at least 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"><br/>
<input type="submit" value="register">
</form>
```

Validation with IMAGE (HTML form on next slide)

```
<script>
function validate(){
var name=document.f1.name.value;
var password=document.f1.password.value;
var status=false;

if(name.length<1){
document.getElementById("nameloc").innerHTML=  " <img src='uncheck.png'/> Please enter 
your name";
status=false;
}else{
document.getElementById("nameloc").innerHTML=" <img src='check.png'/>";  status=true;
}
if(password.length<6){
document.getElementById("passwordloc").innerHTML= " <img src='uncheck.png'/> 
Password must be at least 6 char long";   status=false;
}else{
document.getElementById("passwordloc").innerHTML=" <img src='check.png'/>";
}
return status;
}
</script>
```

HTML form for previous slide

```
<form name="f1" action="#" onsubmit="return validate()">
<table>
<tr><td>Enter Name:</td><td><input type="text" name="name"/>
<span id="nameloc"></span></td></tr>
<tr><td>Enter Password:</td><td><input type="password"
name="password"/>
<span id="passwordloc"></span></td></tr>
<tr><td colspan="2"><input type="submit"
value="register"/></td></tr>
</table>
</form>
```

Regular Expressions: Modifiers (g, i, m)

```
<html> <body>
<p>Click the button to do a global (g), multiline(m), case insensitive (i) search for
"is" at the beginning of each line in a string.</p>
<button onclick="myFunction()">Try it</button>

<p id="strdisp">input string is:</p>

<p id="demo"> Regular Expression result: </p>

<script>
function myFunction() {
  var str = "\nIs th\nis h\nis?";
  document.getElementById("strdisp").innerHTML += str;
  var patt1 = /is/gim;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML += result;
}
</script>  </body> </html>
```

Brackets

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

*** Example of [abc]**

```
<html> <body>
<p>Click the button to do a global search for the character "h" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "Is this all there is?";
  var patt1 = /[h]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script> </body> </html>
```

[^abc]

```
<html> <body>

<p>Click the button to do a global search for characters that are NOT "i" and "s" in a
string.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var str = "Do you know if this is all there is?";
  var patt1 = /[^is]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>  </body>  </html>
```

[^abc] another example

```
<html> <body>

<p>Click the button to do a global search for the character-span NOT from uppercase A to
uppercase E.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var str = "I SCREAM FOR ICE CREAM!";
  var patt1 = /[^A-E]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body> </html>
```

Meta Characters for Regular Expression

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, beginning like this: \bHI, end like this: HI\b
\B	Find a match, but not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character

\d – Find a digit (\D for non-digit)

```
<html> <body>

<p>Click the button to do a global search for digits in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

<script>
function myFunction() {
  var str = "Give 100%!";
  var patt1 = /\d/g;   //use d+ for complete number e.g. 100
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>  </body> </html>
```

JAVASCRIPT EVENTS

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Mouse event: onclick, onmouseover, onmouseout (HTML)

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

<button onclick="document.getElementById('d1').innerHTML=Date()"> The time is?
</button>
<p id="d1"></p>
<button onmouseover= "display()" onmouseout= "displayout()"> Over ME </button>

<p id="d2"></p>
<script>
function display() {
  document.getElementById('d2').innerHTML="mouse over";
}

function displayout() {
  document.getElementById('d2').innerHTML="mouset out";
}
</script>  </body>   </html>
```

Mouse event: onclick (JavaScript)

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

<p>This example uses the HTML DOM to assign an "onclick" event to
a p element.</p>
<p id="demo">Click me.</p>

<script>
document.getElementById("demo").onclick = function()
{myFunction()};

function myFunction() {
  document.getElementById("demo").innerHTML = "CLICKED !!!";
}
</script>  </body>  </html>
```

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

HTML EXAMPLE

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

<p>A function is triggered when the user is pressing a key in the input field.</p>

<input type="text" onkeydown="myFunction()">

<script>
function myFunction() {
  alert("You pressed a key inside the input field");
}
</script>

</body>  </html>
```

Keyboard Event: JavaScript Example

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

<p>This example uses the HTML DOM to assign an "onkeypress" event to
an input element.</p>

<p>Press a key inside the text field to set a red background color.</p>

<input type="text" id="demo">

<script>
document.getElementById("demo").onkeypress = function()
{myFunction()};

function myFunction() {
  document.getElementById("demo").style.backgroundColor = "red";
}
</script>  </body>  </html>
```

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

form Event (onsubmit): JavaScript Example

```
<!DOCTYPEhtml>
<html> <body>
<p>Welcome To My Page.</p>
<p>Have a Nice Day.</p>

<form id="first" action="yourpage.html">
Enter your name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>

<script>
document.getElementById("first").onsubmit = function() {demo()};
function demo() {
alert("The form was submitted successfully");
}
</script>

</body> </html>
```

Window/Document events:

Event Performed	**Event Handler**	**Description**
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

onload

```
<!DOCTYPE html>
<html>
<body onload="myFunction()">

<h1>Hello World!</h1>

<script>
function myFunction() {
  alert("Page is loaded");
}
</script>

</body>
</html>
```

addEventListener

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

<p>This example uses the addEventListener() method to attach a "load" event to an
iframe element.</p>

<iframe id="myFrame" src="/default.asp"></iframe>

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

<script>
document.getElementById("myFrame").addEventListener("load", myFunction);

function myFunction() {
  document.getElementById("demo").innerHTML = "Iframe is loaded.";
}
</script>

</body> </html>
```