UNIT II – Java Script

Mrs.V.Mareeswari
Assistant Professsor
School of Information Technology and Engineering
VIT University

Cabin No:SJT 210-A30

Topics

- Introduction to JavaScript
- Variables
- Conditional and Loops
- Events
- Functions
- Frames
- HTML document
- Predefined Objects
- Image Object
- Layers
- Drag and Drop
- Building a Sample Form.

Introduction

- JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.
- JavaScript is used in billions of Web pages to add functionality, validate forms, communicate with the server, and much more.

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**
- 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

What Can JavaScript 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!
- **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 -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A 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** - A JavaScript can be used to store and retrieve information on the visitor's computer

History

- JavaScript was invented by **Brendan Eich at Netscape** (with Navigator 2.0), and has appeared in all browsers since 1996.
- The official standardization was adopted by the ECMA organization (an industry standardization association) in 1997.
- **ECMA-262** is the official JavaScript standard.

Writing to The HTML Document

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

```
<html> <body>
<h1>My First Web Page</h1>
<script type="text/javascript">
document.write("<p>" + Date() + "</p>");
</script>
</body>
</html>
```

Where to write the Java Script

Scripts in <head> and <body>

- You can place an unlimited number of scripts in your document, and you can have scripts in both the body and the head section at the same time.
- It is a common practice to put all functions in the head section, or at the bottom of the page. This way they are all in one place and do not interfere with page content.

Using an External JavaScript

- JavaScript can also be placed in external files.
- External JavaScript files often contain code to be used on several different web pages.
- External JavaScript files have the file extension .js.
- **Note:** External script cannot contain the <script></script> tags!
- To use an external script, point to the .js file in the "src" attribute of the <script> tag:

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


Statements

- Unlike HTML, **JavaScript is case sensitive** - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.
- A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do.

```
<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>
```

Comments

- Single line comments start with //
- Multi line comments start with /* and end with */

Variables

- Variables are "containers" for storing information.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter, the $ character, or the underscore character

Declaring (Creating) JavaScript Variables

```
var x;
var carname="Volvo";
```


Local JavaScript Variables

- A variable declared within a JavaScript function becomes **LOCAL** and can only be accessed within that function. (the variable has local scope).
- Local variables are destroyed when you exit the function.

Global JavaScript Variables

- Variables declared outside a function become **GLOBAL**, and all scripts and functions on the web page can access it.
- Global variables are destroyed when you close the page.
- If you declare a variable, **without using "var",** the variable always becomes **GLOBAL**.

Operators

- **Arithmetic Operators: + - * / % ++ --**
- **Assignment operators: = += -= *= /= %=**
- To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" " +txt2;
```

- **Comparison Operators : == != > >= < <=**
  === (is exactly equal to (value and type) **)**
- **Logical Operators : && || !**
- **Conditional Operator : ? :**

Output ???

```
<html>
<body>
<script type="text/javascript">
var x;
x=5+5;
document.write(x);
document.write("<br />");
x="5"+"5";
document.write(x);
document.write("<br />");
x=5+"5";
document.write(x);
document.write("<br />");
x="5"+5;
document.write(x);
document.write("<br />");
</script>
<p>The rule is: If you add a number and a string, the result will be a string.</p>
</body>
</html>
```


If...Else Statements

```
<html><body>
<script type="text/javascript">
  var d = new Date();
  var time = d.getHours();
  if (time<10)
    {
    document.write("<b>Good morning</b>");
    }
  else if (time>=10 && time<16)
    {
    document.write("<b>Good day</b>");
    }
  else
    {
    document.write("<b>Hello World!</b>");
    }
  </script> </body></html>
```

```
<html><body>
<script type="text/javascript">
var r=Math.random();
if (r>0.5)
{
document.write("
<a href='http://www.w3schools.com'>
  Learn Web Development!</a>");
}
else
{
document.write("<a
  href='ftp://192.168.2.172/lab/'>Visit FTP Data!</a>");
}
</script></body></html>
```


Switch Statement

```
<html><body>
<script type="text/javascript">
  var d=new Date();
  var theDay=d.getDay();     //Note that Sunday=0,Monday=1, etc.
  switch (theDay)
  {
  case 5:  document.write("Finally Friday");    break;
  case 6:  document.write("Super Saturday");  break;
  case 0:  document.write("Sleepy Sunday");   break;
  default:  document.write("I'm looking forward to this weekend!");
  }
  </script></body><html>
```


For Loop

```
<html><body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br >");
}
</script></body></html>
```

```
<html><body>
<script type="text/javascript">
for (i = 1; i <= 6; i++)
{
document.write("<h" + i + ">This is heading " + i);
document.write("</h" + i + ">");
}
</script></body></html>
```

While and Do-While Loop

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
 {
 document.write("The number is "
+ i);
 document.write("<br >");
 i++;
 }
</script>
</body>
</html>
```

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
 {
 document.write("The number is "
+ i);
 document.write("<br >");
 i++;
 }
while (i<=5);
</script>
</body>
</html>
```


Break and Continue Statement

```
<html> <body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
 {
 if (i==3)
  {
  break;
  }
 document.write("The number is "
+ i);
 document.write("<br >");
 }
</script> </body> </html>
```

```
<html> <body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
 {
 if (i==3)
  {
  continue;
  }
 document.write("The number is "
+ i);
 document.write("<br >");
 }
</script> </body> </html>
```


Functions

- To keep the browser from executing a script when the page loads, you can put your script into a 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 (or even from other pages if the function is embedded in an external .js file).
- Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

Function Definition

- function *functionname*(*var1,var2,...,varX*)
  { *some code* }
- The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.
- A function with no parameters must include the parentheses () after the function name.
- Do not forget about the importance of capitals in JavaScript! The word *function* must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.
- The return statement is used to specify the value that is returned from the function.

```
<html>
<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>
</html>
```


Popup Boxes

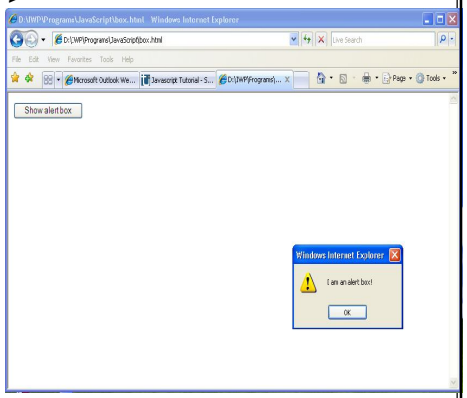
- JavaScript has three kind of popup boxes:

1. **Alert Box**
2. **Confirm Box**
3. **Prompt box**

Alert Box

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

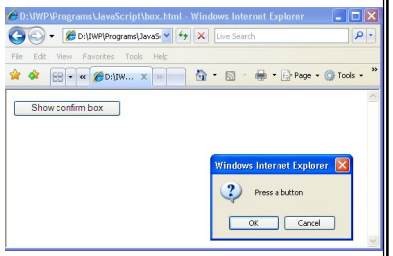
```
<html> <head>
<script type="text/javascript">
function show_alert()
{ alert("I am an alert box!"); }
</script> </head>
<body>
<input type="button"
onclick="show_alert()"
value="Show alert box" >
</body> </html>
```



Confirm Box

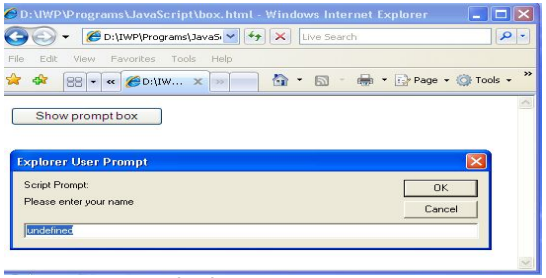
A confirm box is often used if you want the user to verify or accept something. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

```
<html> <head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
 {  alert("You pressed OK!");  }
else
 {  alert("You pressed Cancel!");  }
}
</script> </head>
<body>
<input type="button" onclick="show_confirm()" value="Show confirm box" >
</body> </html>
```

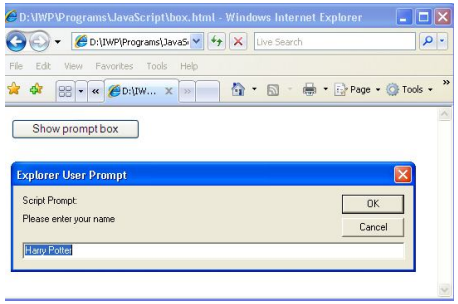


Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.




```
<html> <head> <script type="text/javascript">
function show_prompt()
{
        var name=prompt("Please enter your name","Harry Potter");
        if (name!=null && name!="")
        {  document.write("<p>Hello " + name + "! How are you
                today?</p>"); }
}</script> </head>
<body>
<input type="button"
onclick="show_prompt()"
value="Show prompt box" >
</body> </html>
```



Objects

- **JavaScript is an Object Based Programming language.**
- An Object Based Programming language allows you to define your own objects and make your own variable types.
- An object is just a special kind of data. An object has properties and methods.
- **Properties** are the values associated with an object.

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

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

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


String object

- The String object is used to manipulate a stored piece of text.

```
var txt="Hello world!";
document.write(txt.length);  //12
document.write(txt.toUpperCase()); //HELLOWORLD!
document.write(txt.match("world") ); //world
document.write(txt.match("World") ) ;//null
document.write(txt.indexOf("world")); //6
var str="Visit Microsoft!";
document.write(str.replace("Microsoft","CTS")); //Visit CTS!
```

Example:

```
var txt = "Hello World!";
document.write("<p>Big: " + txt.big() + "</p>");
document.write("<p>Small: " + txt.small() + "</p>");
document.write("<p>Bold: " + txt.bold() + "</p>");
document.write("<p>Italic: " + txt.italics() + "</p>");
document.write("<p>Strike: " + txt.strike() + "</p>");
document.write("<p>Fontcolor: " + txt.fontcolor("green") +
  "</p>");
document.write("<p>Fontsize: " + txt.fontsize(6) + "</p>");
document.write("<p>Subscript: " + txt.sub() + "</p>");
document.write("<p>Superscript: " + txt.sup() + "</p>");
document.write("<p>Link: " +
  txt.link("http://www.w3schools.com") + "</p>");
document.write("<p>Blink: " + txt.blink() + " (does not work in IE,
  Chrome, or Safari)</p>");
```


Date Object

- The Date object is used to work with dates and times.
- var d = new Date();
- getDate() → Returns the day of the month (from 1-31)
- getDay() → Returns the day of the week (from 0-6)
- getYear()→Returns the year (four digits)
- getHours()→Returns the hour (from 0-23)
- getMinutes()→Returns the minutes (from 0-59)
- getMonth()→Returns the month (from 0-11)
- getSeconds()→Returns the seconds (from 0-59)
- getTime() → Returns the number of milliseconds since midnight Jan 1, 1970

Boolean Object

- The Boolean object represents two values: "true" or "false".
- var myBoolean=new Boolean();
- 0 →False
- 1 →True

Array Object

- The Array object is used to store multiple values in a single variable.
- var myCars=new Array(); // regular array (add an optional integer myCars[0]="Saab"; // argument to control array's size)
  myCars[1]="Volvo";
  myCars[2]="BMW";
- var myCars=["Saab","Volvo","BMW"]; //literal array
- var myCars=new Array("Saab","Volvo","BMW"); // condensed array

```
var a=new Array(10);
for(var i=0;i<10;i++)
{
        a[i]=i+1;
        document.write(a[i]);
}
```

Example 1:

```
var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);    // Jani,Tove,Cecilie,Lone
```

Example 2:

```
var brothers = ["Stale", "Kai Jim", "Borge"];
var family = parents.concat(brothers, children);
document.write(family); //Jani,Tove,Stale,Kai Jim,Borge,Cecilie,Lone
```

Example 3:

```
var fruits = ["Banana", "Orange", "Apple" ];
document.write(fruits.join() + "<br >"); //Banana,Orange,Apple
document.write(fruits.join("+") + "<br >"); //Banana+Orange+Apple
document.write(fruits.join(" and ")); //Banana and Orange and Apple
```


Example 4:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.pop() ); //Mango //remove the last item
document.write(fruits ); //Banana,Orange,Apple
document.write(fruits.push("Lemon","Pineapple") ); //5
document.write(fruits); //Banana,Orange,Apple, Lemon,Pineapple
document.write(fruits.reverse()); //Pineapple, Lemon, Apple, Orange,
                                  Banana
document.write(fruits.shift() ); // Pineapple
document.write(fruits); // Lemon, Apple, Orange, Banana
document.write(fruits.unshift("Kiwi","Pineapple") ); //6
// added 5th and 6th item
document.write(fruits); // Kiwi,Pineapple,Lemon,Apple,Orange,Banana
```


Example 5:

```
Var fruits= ["Lemon", "Apple", "Orange", "Banana"];
//Display from index 0 to index 2 (0 and1)
document.write(fruits.slice(0,2)); // Lemon, Apple
document.write(fruits.slice(1)); //Apple, Orange, Banana //From 1st
document.write(fruits.slice(-2)); // Orange, Banana //Last 2 items
```

Example 6:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort()); // Apple,Banana,Mango,Orange
var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort()); //1,10,100,25,40,5
```

Example 7:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.toString()); //Banana,Orange,Apple,Mango
```


Numbers

- The Number() function converts the object argument to a number that represents the object's value.
- If the value cannot be converted to a legal number, NaN is returned.
- **Note:** If the parameter is a Date object, the Number() function returns the number of milliseconds since midnight January 1, 1970

```
var test4= 10 , test5= 20;
document.write(test4 +test5); //1020
document.write(Number(test4)+Number(test5)); //30
document.write(Number("99.66") +Number("01.34") ); //101
The parseInt() function parses a string and returns an integer.
document.write(parseInt("10.33") ) ;  //10
document.write(parseFloat("10.33") ); //10.33
```

Math Object

- sqrt(x)→Returns the square root of x
- ceil(x)→Returns x, rounded upwards to the nearest integer
- floor(x)→Returns x, rounded downwards to the nearest integer
- exp(x)→Returns the value of E^x
- 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)
- cos(x)→Returns the cosine of x (x is in radians)
- tan(x)→Returns the tangent of an angle

For...In Statement

- The code in the body of the for…in loop is executed once for each property.

```
for (variable in object)
  {
  code to be executed
  }
var person={fname:"John",lname:"Doe",age:25};  //object creation
var x;
for (x in person)
{
document.write(person[x] + " "); // John Doe 25
}
```


Events

- By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.
- Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

onLoad and onUnload

- The onLoad and onUnload events are triggered when the user enters or leaves the page.
- The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.
- Both the onLoad and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange

- The onFocus, onBlur and onChange events are often used in combination with validation of form fields.
- The checkEmail() function will be called whenever the user changes the content of the field:
- `<input type="text" size="30" id="email" onchange="checkEmail()" />`

onSubmit

- The onSubmit event is used to validate ALL form fields before submitting it.
- The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<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.

- onresize → The event occurs when the size of an element has changed
- onselect →The event occurs after some text has been selected in an element
- onclick→The event occurs when the user clicks on an element
- ondblclick→The event occurs when the user double-clicks on an element
- onkeypress→The event occurs when the user is pressing a key or holding down a key

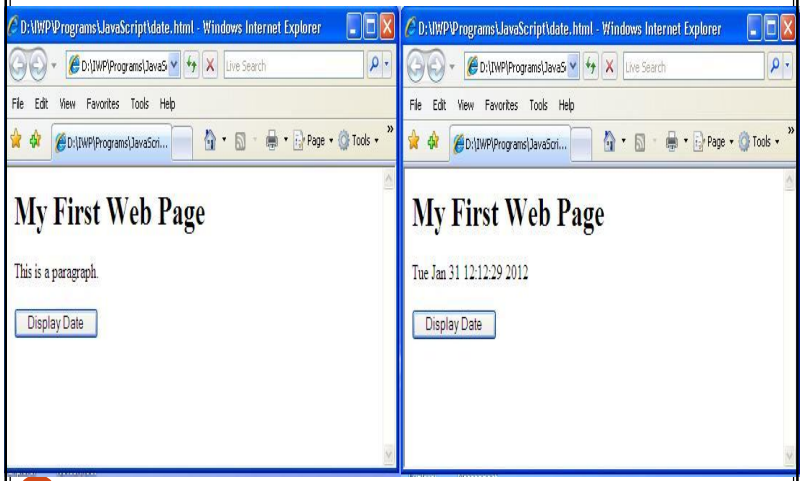
Event Properties:

- screenX → Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered
- screenY → Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered


```
<html> <head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
} </script> </head>
<body>
<h1>My First Web Page</h1>
<p id="demo">This is a paragraph.</p>
<input type= "button" onClick="displayDate()" value="Display Date">
</body>
</html>
```

innerHTML → Sets or returns the HTML contents (+text) of an element

Output



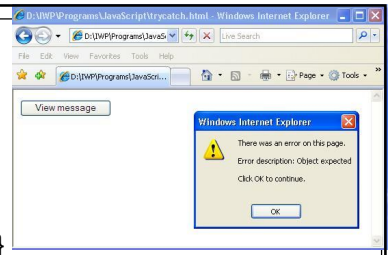
Try...Catch Statement

- When browsing Web pages on the internet, we all have seen a JavaScript alert box telling us there is a runtime error and asking "Do you wish to debug?". Error message like this may be useful for developers but not for users. When users see errors, they often leave the Web page.

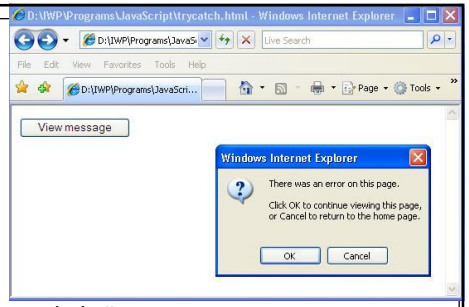
```
try
  {
  //Run some code here
  }
catch(err)
  {
  //Handle errors here
  }
```



```
<html> <head>
<script type="text/javascript">
var txt="";
function message()
{
try
   {       adddlert("Welcome guest!");   }
catch(err)
   {      txt="There was an error on this page.\n\n";
          txt+="Error description: " + err.message + "\n\n";
          txt+="Click OK to continue.\n\n";
          alert(txt);                          }        }
</script> </head>     <body>
<input type="button" value="View message" onclick="message()" >
</body> </html>
```




```
<html> <head>
<script type="text/javascript">
  var txt="";
  function message()
  {
  try
{adddlert("Welcome guest!"); }
catch(err)
{
 txt="There was an error on this page.\n\n";
 txt+="Click OK to continue viewing this page,\n";
 txt+="or Cancel to return to the home page.\n\n";
 if(!confirm(txt))
 {     document.location.href="http://www.w3schools.com/";    }  }  }
</script> </head>  <body>
<input type="button" value="View message" onclick="message()" />
</body> </html>
```




```
<html>  <body> <script type="text/javascript">
var x=prompt("Enter a number between 5 and 10:","");
try
 {        if(x>10)
            {    throw "Err1";    }
          else if(x<5)
            {    throw "Err2";    }
          else if(isNaN(x))
            {    throw "Err3";    }  }
catch(err)
 {        if(err=="Err1")
            {    document.write("Error! The value is too high.");    }
          if(err=="Err2")
            {    document.write("Error! The value is too low.");    }
          if(err=="Err3")
            {    document.write("Error! The value is not a number.");    }   }
</script> </body> </html>
```

The throw statement allows you to create an exception. The exception can be a string, integer, Boolean or an object.

throw *exception;*

Special Characters

- In JavaScript you can add special characters to a text string by using the backslash sign.
- \' → single quote
- \" → double quote
- \\ → backslash
- \n → new line
- \t → tab

Form Object

- The Form object represents an HTML form.
- For each <form> tag in an HTML document, a Form object is created.
- 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.

Form Object Collections

- elements[] →Returns an array of all elements in a form

Form Object Properties

- acceptCharset → Sets or returns the value of the accept-charset attribute in a form.

  Eg: Character encoding for Unicode (UTF-8) , Latin alphabet (ISO-8859-1)
- action → Sets or returns the value of the action attribute in a form
- enctype → Sets or returns the value of the enctype attribute in a form
- length →Returns the number of elements in a form
- method→ Sets or returns the value of the method attribute in a form (get or post)
- name →Sets or returns the value of the name attribute in a form
- target→Sets or returns the value of the target attribute in a form (_blank,_top,_parent,_self, framename)

Form Object Methods

1. reset()→Resets a form

   Eg : document.getElementById("form1").reset();
1. submit()→Submits a form

Form Object Events

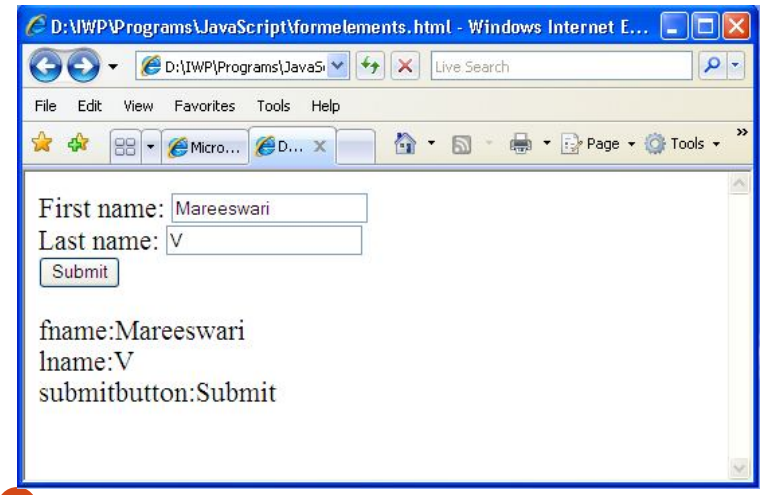
1. onreset→ The reset button is clicked
2. onsubmit→The submit button is clicked

```
<form onreset="alert('The form will be reset')">
Firstname: <input type="text" name="fname" value="Marees" ><br >
Lastname: <input type="text" name="lname" value="V" ><br ><br>
<input type="reset" value="Reset" >
</form>
```

The elements collection returns an array of all the elements in a form

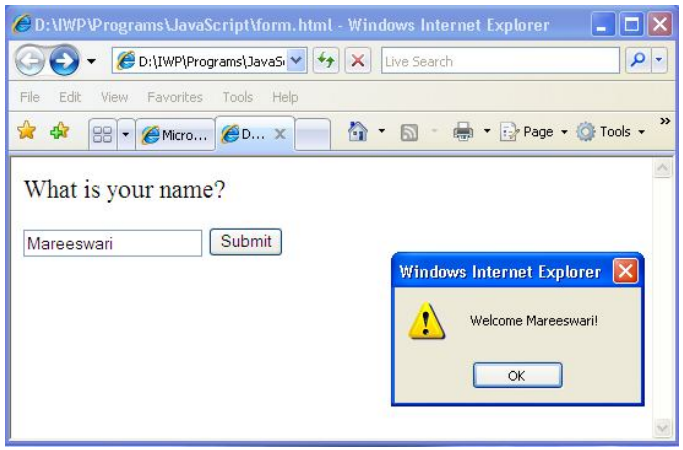
```
<html> <body>
<form id="form1" action="">
  First name: <input type="text" name="fname" value="Mareeswari">
  <br>
  Last name: <input type="text" name="lname" value="V"><br>
  <input type="button" value="Submit" name="submitbutton"> /form>
<script type="text/javascript">
  var x=document.getElementById("form1");
  for (var i=0;i<x.length;i++)
{
document.write(x.elements[i].name+":"+x.elements[i].value + "<br>");
   } </script> </body> </html>
```


Output




```
<html> <head>
<script type="text/javascript">
function greeting()
{
alert("Welcome " + document.forms["form1"]["fname"].value + "!");
}
</script> </head> <body>
What is your name?<br>
<form  name="form1"  action="image.html"
        onsubmit="greeting()">
<input type="text" name="fname">
<input type="submit" value="Submit">
</form> </body> </html>
```


Output



Form – reset () method

```
<html> <head>
<script type="text/javascript">
function formReset()
{ document.getElementById("frm1").reset(); }
</script> </head> <body>
<p>Enter some text in the fields below, then press the "Reset form"
  button to reset the form.</p>
<form id="frm1">
First name: <input type="text" name="fname" ><br >
Last name: <input type="text" name="lname" ><br ><br >
<input type="button" onclick="formReset()" value="Reset form" >
</form> </body> </html>
```


E-mail Validation

- The function below checks if the content has the general syntax of an email.
- This means that the input data must contain an @ sign and at least one dot (.). Also, the @ must not be the first character of the email address, and the last dot must be present after the @ sign, and minimum 2 characters before the end:

```
function validateForm(){
var x=document.forms["myForm"]["email"].value;
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
  {  alert("Not a valid e-mail address");  return false;  }}
```

- The function above could be called when a form is submitted:

```
<form name="myForm" action="login.html" onsubmit="return
validateForm();" method="post">
Email: <input type="text" name="email">
<input type="submit" value="Submit"></form>
```

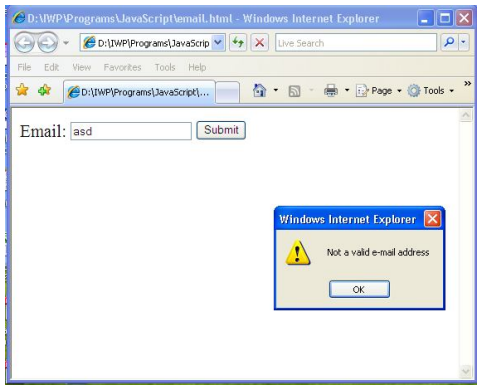



Image Object

- For each <img> tag in an HTML document, an Image object is created.

Properties:

- **align** → Sets or returns the value of the align attribute of an image
- **alt** → Sets or returns the value of the alt attribute of an image
- **border** → Sets or returns the value of the border attribute of an image
- **complete**→ Returns whether or not the browser is finished loading an image
- **height**→ Sets or returns the value of the height attribute of an image
- **hspace**→ Sets or returns the value of the hspace (left, right) attribute of an image
- **longDesc**→ Sets or returns the value of the longdesc attribute of an image
- **lowsrc**→ Sets or returns a URL to a low-resolution version of an image
- **name**→Sets or returns the name of an image
- **src**→ Sets or returns the value of the src attribute of an image
- **useMap**→ Sets or returns the value of the usemap attribute of an image
- **vspace**→Sets or returns the value of the vspace (top, bottom) attribute of an image
- **width**→ Sets or returns the value of the width attribute of an image

Image Object- Event

- onabort → Loading of an image is interrupted
- onerror → An error occurs when loading an image
- onload → An image is finished loading

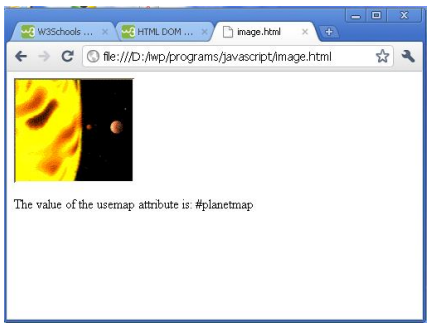
Image - useMap Property

- The useMap property sets or returns the value of the usemap attribute of an image.
- The usemap attribute specifies an image as a client-side image-map (an image-map is an image with clickable areas).
- The usemap attribute is associated with a map element's name attribute, and creates a relationship between the image and the map.

Example (image.html)

```
<html> <body>
<img id="planets" src="planets.gif" width="145" height="126"
  useMap="#planetmap">
<map name="planetmap">
<area id="venus" shape="circle" coords="124,58,8" alt="The planet Venus"
  href="venus.html">
<area id="earth" shape="square" coords="0,0,100,100" alt="The planet
  Earth" href="earth.html">
</map>
<p>The value of the usemap attribute is:
<script type="text/javascript">
document.write(document.getElementById("planets").useMap);
</script> </p> </body> </html>
```


Output



Example (earth.html)

```
<html> <head>
<script type="text/javascript">
function setSpace()
{
document.getElementById("compman").hspace="50";
document.getElementById("compman").vspace="50";
}
</script> </head> <body>
 <img id="compman" src="compman.gif" alt="Computerman" width=107
   height=98>
<p>Some text. Some text. Some text. Some text.</p>
<input type="button" onclick="setSpace()" value="Set hspace and vspace">
</body> </html>
```


Output

onLoad onClick

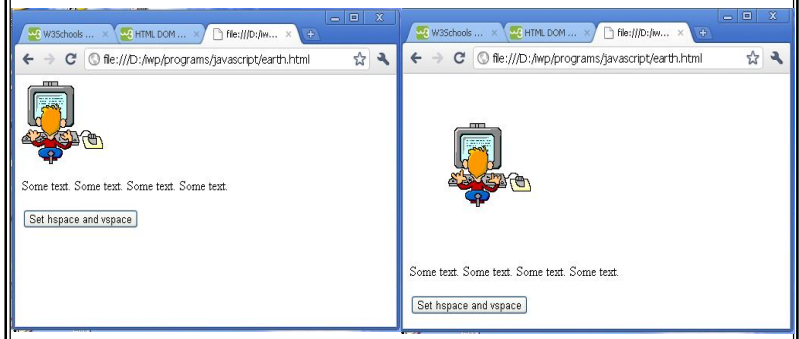


Image Object - Complete

```
<html> <head>
<script type="text/javascript">
function alertComplete()
{
alert("Image loaded: " +
document.getElementById("compman").complete); //true or false
}
</script> </head>
<body onload="alertComplete()">
<img id="compman" src="compman.gif" alt="Computerman" />
</body> </html>
```


Frame Object

- The Frame object represents an HTML frame.
- The <frame> tag defines one particular window (frame) within a frameset.
- For each <frame> tag in an HTML document, a Frame object is created.

Frame Object Events:

- **onload** → Script to be run immediately after a frame is loaded

Frame Properties

- **contentDocument**→Returns the document object generated by a frame
- **contentWindow**→Returns the window object generated by a frame
- **frameBorder**→Sets or returns the value of the frameborderattribute in a frame.
- **longDesc**→Sets or returns the value of the longdescattribute in a frame
- **marginHeight**→Sets or returns the value of the marginheightattribute in a frame
- **marginWidth**→Sets or returns the value of the marginwidthattribute in a frame
- **name**→Sets or returns the value of the name attribute in a frame
- **noResize**→Sets or returns the value of the noresizeattribute in a frame
- **scrolling**→Sets or returns the value of the scrolling attribute in a frame
- **src**→Sets or returns the value of the srcattribute in a frame

frameset.html

```
<html>
<frameset cols="50%,50%">
 <frame id="leftFrame" src="LeftFrameCode.html">
 <frame id="rightFrame" src="RightFrameCode.html">
</frameset>
</html>
```

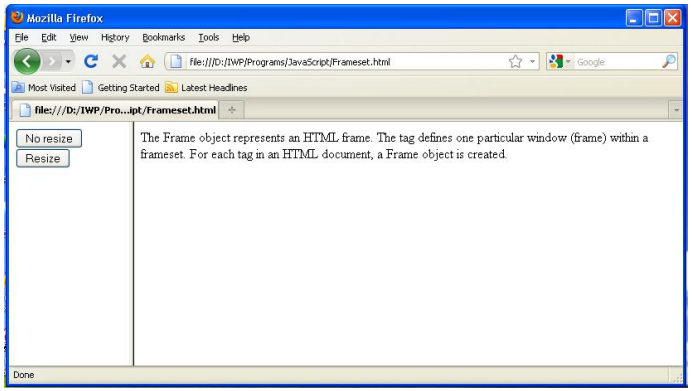

LeftFrameCode.html

```
<html><head><script type="text/javascript">
function disableResize()
 {
  parent.document.getElementById("rightFrame").noResize=true;
 }
function enableResize()
 {
  parent.document.getElementById("rightFrame").noResize=false;
 }
</script></head><body>
<input type="button" onclick="disableResize()" value="No resize“ >
<input type="button" onclick="enableResize()" value="Resize" >
</body></html>
```


RightFrameCode.html

```
<html>
<body>
The Frame object represents an HTML frame.
The <frame> tag defines one particular window (frame) within a
  frameset.
For each <frame> tag in an HTML document, a Frame object is created.
</body>
</html>
```

Output



JavaScript Timing Events

- With JavaScript, it is possible to execute some code after a specified time-interval. This is called timing events.

The setTimeout() Method - executes a code some time in the future

var t=setTimeout("*javascript statement*",*milliseconds*);

- The setTimeout() method returns a value. In the syntax defined above, the value is stored in a variable called t. If you want to cancel the setTimeout() function, you can refer to it using the variable name.
- The first parameter of setTimeout() can be a string of executable code, or a call to a function. The second parameter indicates how many milliseconds from now you want to execute the first parameter.
- **Note:** There are 1000 milliseconds in one second.

The clearTimeout() Method - cancels the setTimeout()

clearTimeout(*setTimeout_variable*)

- **Note:** The setTimeout() and clearTimeout() are both methods of the HTML DOM Window object.


```
<html> <head> <script type="text/javascript">
function Redirect()
{
  //window.location="http://www.vit.ac.in";
  document.location.href="http://www.vit.ac.in";
}
document.write("You will be redirected to main page in 10 sec.");
setTimeout("Redirect()", 10000);
</script> </head> </html>
```

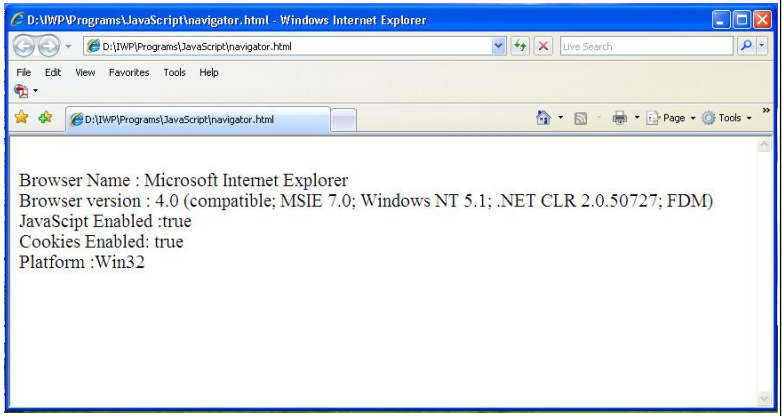

Navigator Object

- However, there are some things that just don't work on certain browsers - especially on older browsers.
- Sometimes it can be useful to detect the visitor's browser, and then serve the appropriate information.
- The Navigator object contains information about the visitor's browser name, version, and more.

Properties:

- appVersion→ This property is a string that contains the version of the browser as well as other useful information such as its language and compatibility.
- platform →This property is a string that contains the platform for which the browser was compiled. "Win32" for 32-bit Windows operating systems

```
<html>
<head>
<script type="text/javascript">
document.write("<br> Browser Name : " + navigator.appName);
document.write("<br> Browser version  : " + navigator.appVersion);
document.write("<br> JavaScipt Enabled :"+navigator.javaEnabled());
document.write("<br> Cookies Enabled: " + navigator.cookieEnabled)
document.write("<br> Platform  :"+ navigator.platform);
</script> </head>
</html>
```

Creating Your Own Objects

1. Create a direct instance of an object

```
personObj=new Object();
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=50;
personObj.eyecolor="blue";
```

Alternative syntax (using object literals):

```
personObj={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};
```

Adding a method to the personObj is also simple.

The following code adds a method called eat() to the personObj:

```
personObj.eat=eat;
```


2. Create an object constructor

- Create a function that construct objects:

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname;
this.lastname=lastname;
this.age=age;                    //"this" is: the instance of the object at hand.
this.eyecolor=eyecolor;
this.newlastname=newlastname; //method
}
```

- Once you have the object constructor, you can create new instances of the object, like this:

```
var myFather=new person("John","Doe",50,"blue");
var myMother=new person("Sally","Rally",48,"green");
```

```
<html><body><script type="text/javascript">
function mycircle(x,y,r)                    //Constructor
{
  this.xcoord = x;        // Adding Properties
  this.ycoord = y;
  this.radius = r;
  this.retArea = retArea; // Adding Methods
}
function retArea() {              // Method Definition
  return ( Math.PI * this.radius * this.radius );
}
var testcircle = new mycircle(3,4,5);            // Object Creation
alert( 'The area of the circle is ' + testcircle.retArea() ); // Method Call
</script></body></html>
```