CONTENTS

What is Javascript?

JavaScript was designed to add interactivity to HTML pages. Its an interpreted language.

What can a JavaScript Do?

1. **JavaScript gives HTML designers a programming tool** - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
2. **JavaScript can put dynamic text into an HTML page** - A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
3. **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
4. **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element
5. **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server, this will save the server from extra processing
6. **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
7. **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

How to Put a JavaScript Into an HTML Page

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

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Semicolons at the end of statement is optional. If more than one statement is put on the same line then we have to use semicolon.

To handle old browsers which donot support Javascript and show the script code as page content:

```
<script type="text/javascript">
<!--
document.write("Hello World!")
//-->
</script>
```

The two forward slashes at the end of comment line (//) are a JavaScript comment symbol. This prevents the JavaScript compiler from compiling the line.

Where to Put the JavaScript

Scripts in the head section: Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

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

```
....
</script>
</head>
```

Scripts in the body section: Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

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

Using an External JavaScript: Sometimes you might want to run the same JavaScript on several pages, without having to write the same script on every page.

To simplify this, you can write a JavaScript in an external file. Save the external JavaScript file with a .js file extension.

Note: The external script cannot contain the <script> tag!

To use the external script, point to the .js file in the "src" attribute of the <script> tag:

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

Note: Remember to place the script exactly where you normally would write the script!

JavaScript Variables

A variable is a "container" for information you want to store.

A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.

Rules for variable names:

- Variable names are case sensitive
- They must begin with a letter or the underscore character

You can create a variable with the *var* statement:

```
var strname = some value
```

You can also create a variable without the var statement:

```
strname = some value
```

Local Variables: When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called **local variables**.

Global Variables: If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

JavaScript If...Else Statements

In JavaScript we have the following conditional statements:

- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed

```
<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>
```

The JavaScript Switch Statement example:

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.

var d=new Date()
theDay=d.getDay()
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>
```

JavaScript Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true

||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

String Operator – Concatenation (+)

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

The variable txt3 now contains "What a very nice day!".

Conditional Operator

```
greeting=(visitor=="PRES")?"Dear President ":"Dear "
```

JavaScript Popup Boxes

In JavaScript we can create three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert

When an alert box pops up, the user will have to click "OK" to proceed.

```
alert("sometext")
```

```
<html>
<head>
<script type="text/javascript">
function disp_alert()
{
alert("Hello again! This is how we" + "\n" + "add line breaks to an alert box!")
}
</script>
</head>

<body>
<form>
<input type="button" onclick="disp_alert()" value="Display alert box">
</form>
</body>

</html>
```

Confirm

A confirm box is often used if you want the user to verify or accept something.
When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
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 disp_confirm()
{
var name=confirm("Press a button")
if (name==true)
{
document.write("You pressed the OK button!")
}
else
{
document.write("You pressed the Cancel button!")
}
}
</script>
</head>

<body>
<form>
<input type="button" onclick="disp_confirm()" value="Display a confirm
box">
</form>
</body>

</html>
```

Prompt
A prompt box is often used if you want the user to input a value before entering a page.

```
prompt("sometext","defaultvalue")
```

```
<html>
<head>
<script type="text/javascript">
function disp_prompt()
{
var name=prompt("Please enter your name","")
if (name!=null && name!="")
{
document.write("Hello " + name + "! How are you today?")
}
}
</script>
</head>

<body>
<form>
<input type="button" onclick="disp_prompt()" value="Display a prompt
box">
</form>
</body>

</html>
```

JavaScript Functions

To keep the browser from executing a script as soon as the page is loaded, you can write your script as a function.
A function contains some code that will be executed only by an event or by a call to that function.

You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
Functions are defined at the beginning of a page, in the <head> section.

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!")
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

If the line: alert("Hello world!!"), in the example above had not been written within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before the user hits the button. We have added an onClick event to the button that will execute the function displaymessage() when the button is clicked.

The return statement is used to specify the value that is returned from the function.

```
function prod(a,b)
{
x=a*b
return x
}

product=prod(2,3)
```

The returned value from the prod() function is 6, and it will be stored in the variable called product.

JavaScript Loops

For Loop

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

While Loop

```
<html>
<body>
<script type="text/javascript">
var i=0
while (i<=10)
{
document.write("The number is " + i)
document.write("<br />")
i=i+1
}
</script>
</body>
</html>
```

Do..While Loop

```
<html>
<body>
<script type="text/javascript">
var i=0
do
{
document.write("The number is " + i)
document.write("<br />")
i=i+1
}
while (i<0)
</script>
</body>
</html>
```

Break and Continue

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
{
if (i==3){break}
if (i==4){continue}
document.write("The number is " + i)
document.write("<br />")
}
</script>
</body>
</html>
```

JavaScript For...In Statement

```
<html>
<body>
<script type="text/javascript">
var x
var mycars = new Array()
```

```
mycars[0] = "Saab"
mycars[1] = "Volvo"
mycars[2] = "BMW"

for (x in mycars)
{
document.write(mycars[x] + "<br />")
}
</script>
</body>
</html>
```

JavaScript Events

Events are actions that can be detected by JavaScript. Every element on a web page has certain events which can trigger JavaScript functions. 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.

New to HTML 4.0 was the ability to let HTML events trigger actions in the browser, like starting a JavaScript when a user clicks on an HTML element. Below is a list of the attributes that can be inserted into HTML tags to define event actions.

Attribute	The event occurs when...
onabort	Loading of an image is interrupted
onblur	An element loses focus
onchange	The content of a field changes
onclick	Mouse clicks an object
ondblclick	Mouse double-clicks an object
onerror	An error occurs when loading a document or an image
onfocus	An element gets focus
onkeydown	A keyboard key is pressed
onkeypress	A keyboard key is pressed or held down
onkeyup	A keyboard key is released
onload	A page or an image is finished loading
onmousedown	A mouse button is pressed
onmousemove	The mouse is moved
onmouseout	The mouse is moved off an element
onmouseover	The mouse is moved over an element
onmouseup	A mouse button is released
onreset	The reset button is clicked
onresize	A window or frame is resized
onselect	Text is selected
onsubmit	The submit button is clicked
onunload	The user exits the page

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.
Below is an example of how to use the onChange event. 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.
Below is an example of how to use the onSubmit event. 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 and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return false">
<img src="w3schools.gif" width="100" height="30">
</a>
```

JavaScript Try...Catch Statement

The example below contains the "Welcome guest!" example rewritten to use the try...catch statement. Since alert() is misspelled, a JavaScript error occurs. However, this time, the catch block catches the error and executes a custom code to handle it. The code displays a custom error message informing the user what happened:

```
<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.description + "\n\n"
  txt+="Click OK to continue.\n\n"
  alert(txt)
  }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
```

```
</body>

</html>
```

The Throw Statement

The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

```
throw(exception)
```

The exception can be a string, integer, Boolean or an object.

The example below determines the value of a variable called x. If the value of x is higher than 10 or lower than 0 we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed:

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","")
try
{
if(x>10)
throw "Err1"
else if(x<0)
throw "Err2"
}
catch(er)
{
if(er=="Err1")
alert("Error! The value is to high")
if(er == "Err2")
alert("Error! The value is to low")
}
</script>
</body>
</html>
```

The onerror Event

A way to write a global error handling function.

```
<html>
<head>
<script type="text/javascript">
onerror=handleErr
var txt=""
function handleErr(msg,url,l)
{
txt="There was an error on this page.\n\n"
txt+="Error: " + msg + "\n"
txt+="URL: " + url + "\n"
txt+="Line: " + l + "\n\n"
txt+="Click OK to continue.\n\n"
alert(txt)
return true
}
function message()
```

```
{
adddlert("Welcome guest!")
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

JavaScript Special Characters

In JavaScript you can add special characters to a text string by using the backslash sign. The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

```
var txt="We are the so-called \"Vikings\" from the north."
```

JavaScript Guidelines

1. Javascript is case sensitive. A function named "myfunction" is not the same as "myFunction" and a variable named "myVar" is not the same as "myvar".
2. You can break up a code line **within a text string** with a backslash.
   ```
   document.write("Hello \
   World!")
   ```
3. You can add comments to your script by using two slashes // or or by using /* and */ (this creates a multi-line comment block):
   ```
   //this is a comment

   /* This is a comment
   block. It contains
   several lines */
   ```

JavaScript Objects

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types. An object is composed of properties and methods. In the subsections below we will see the built-in objects:

String Object

Method	Description

anchor()	Creates an HTML anchor
big()	Displays a string in a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
charAt()	Returns the character at a specified position
charCodeAt()	Returns the Unicode of the character at a specified position
concat()	Joins two or more strings
fixed()	Displays a string as teletype text
fontcolor()	Displays a string in a specified color
fontsize()	Displays a string in a specified size
fromCharCode()	Takes the specified Unicode values and returns a string
indexOf()	Returns the position of the first occurrence of a specified string value in a string
italics()	Displays a string in italic
lastIndexOf()	Returns the position of the last occurrence of a specified string value, searching backwards from the specified position in a string
link()	Displays a string as a hyperlink
match()	Searches for a specified string value in a string
replace()	Replaces some characters with some other characters in a string
search()	Searches a string for a specified value
slice()	Extracts a part of a string and returns the extracted part in a new string
small()	Displays a string in a small font
split()	Splits a string into an array of strings
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript
substr()	Extracts a specified number of characters in a string, from a start index
substring()	Extracts the characters in a string between two specified indices
sup()	Displays a string as superscript
toLowerCase()	Displays a string in lowercase letters
toUpperCase()	Displays a string in uppercase letters
toSource()	Represents the source code of an object
valueOf()	Returns the primitive value of a String object

Property	**Description**
constructor	A reference to the function that created the object
length	Returns the number of characters in a string
prototype	Allows you to add properties and methods to the object

Example:

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

Date Object

Method	**Description**
Date()	Returns today's date and time
getDate()	Returns the day of the month from a Date object (from 1-31)
getDay()	Returns the day of the week from a Date object (from 0-6)

getMonth()	Returns the month from a Date object (from 0-11)
getFullYear()	Returns the year, as a four-digit number, from a Date object
getYear()	Returns the year, as a two-digit or a four-digit number, from a Date object. Use getFullYear() instead !!
getHours()	Returns the hour of a Date object (from 0-23)
getMinutes()	Returns the minutes of a Date object (from 0-59)
getSeconds()	Returns the seconds of a Date object (from 0-59)
getMilliseconds()	Returns the milliseconds of a Date object (from 0-999)
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
getTimezoneOffset()	Returns the difference in minutes between local time and Greenwich Mean Time (GMT)
getUTCDate()	Returns the day of the month from a Date object according to universal time (from 1-31)
getUTCDay()	Returns the day of the week from a Date object according to universal time (from 0-6)
getUTCMonth()	Returns the month from a Date object according to universal time (from 0-11)
getUTCFullYear()	Returns the four-digit year from a Date object according to universal time
getUTCHours()	Returns the hour of a Date object according to universal time (from 0-23)
getUTCMinutes()	Returns the minutes of a Date object according to universal time (from 0-59)
getUTCSeconds()	Returns the seconds of a Date object according to universal time (from 0-59)
getUTCMilliseconds()	Returns the milliseconds of a Date object according to universal time (from 0-999)
parse()	Takes a date string and returns the number of milliseconds since midnight of January 1, 1970
setDate()	Sets the day of the month in a Date object (from 1-31)
setMonth()	Sets the month in a Date object (from 0-11)
setFullYear()	Sets the year in a Date object (four digits)
setYear()	Sets the year in the Date object (two or four digits). Use setFullYear() instead !!
setHours()	Sets the hour in a Date object (from 0-23)
setMinutes()	Set the minutes in a Date object (from 0-59)
setSeconds()	Sets the seconds in a Date object (from 0-59)
setMilliseconds()	Sets the milliseconds in a Date object (from 0-999)
setTime()	Calculates a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month in a Date object according to universal time (from 1-31)
setUTCMonth()	Sets the month in a Date object according to universal time (from 0-11)
setUTCFullYear()	Sets the year in a Date object according to universal time (four digits)
setUTCHours()	Sets the hour in a Date object according to universal time (from 0-23)
setUTCMinutes()	Set the minutes in a Date object according to universal time (from 0-59)
setUTCSeconds()	Set the seconds in a Date object according to universal time (from 0-59)
setUTCMilliseconds()	Sets the milliseconds in a Date object according to universal time (from 0-999)
toSource()	Represents the source code of an object
toString()	Converts a Date object to a string
toGMTString()	Converts a Date object, according to Greenwich time, to a string. Use toUTCString() instead !!
toUTCString()	Converts a Date object, according to universal time, to a string
toLocaleString()	Converts a Date object, according to local time, to a string
UTC()	Takes a date and returns the number of milliseconds since midnight of January 1, 1970 according to universal time
valueOf()	Returns the primitive value of a Date object

Property	**Description**
constructor	A reference to the function that created the object

prototype	Allows you to add properties and methods to the object

Example: Comparing today's date with 14th Jan 2010

```
var myDate=new Date()
myDate.setFullYear(2010,0,14)
var today = new Date()
if (myDate>today)
  alert("Today is before 14th January 2010")
else
  alert("Today is after 14th January 2010")
```

Array Object

```
var mycars=new Array("Saab","Volvo","BMW")

document.write(mycars[0])
mycars[0]="Opel"
```

Method	Description
concat()	Joins two or more arrays and returns the result
join()	Puts all the elements of an array into a string. The elements are separated by a specified delimiter
pop()	Removes and returns the last element of an array
push()	Adds one or more elements to the end of an array and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes and returns the first element of an array
slice()	Returns selected elements from an existing array
sort()	Sorts the elements of an array
splice()	Removes and adds new elements to an array
toSource()	Represents the source code of an object
toString()	Converts an array to a string and returns the result
unshift()	Adds one or more elements to the beginning of an array and returns the new length
valueOf()	Returns the primitive value of an Array object

Property	Description
constructor	A reference to the function that created the object
index	
input	
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to the object

Boolean Object

The Boolean object is an object wrapper for a Boolean value.

All the following lines of code create Boolean objects with an initial value of false:

```
var myBoolean=new Boolean()
var myBoolean=new Boolean(0)
var myBoolean=new Boolean(null)
var myBoolean=new Boolean("")
var myBoolean=new Boolean(false)
var myBoolean=new Boolean(NaN)
```

And all the following lines of code create Boolean objects with an initial value of true:

```
var myBoolean=new Boolean(true)
var myBoolean=new Boolean("true")
```

```
var myBoolean=new Boolean("false")
var myBoolean=new Boolean("Richard")
```

Method	Description
toSource()	Represents the source code of an object
toString()	Converts a Boolean value to a string and returns the result
valueOf()	Returns the primitive value of a Boolean object

Property	Description
constructor	A reference to the function that created the object
prototype	Allows you to add properties and methods to the object

Math Object

The Math object includes several mathematical values and functions. You do not need to define the Math object before using it.

```
document.write(Math.floor(Math.random()*11))
Outputs: 0
```

Method	Description
abs(x)	Returns the absolute value of a number
acos(x)	Returns the arccosine of a number
asin(x)	Returns the arcsine of a number
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the angle theta of an (x,y) point as a numeric value between -PI and PI radians
ceil(x)	Returns the value of a number rounded upwards to the nearest integer
cos(x)	Returns the cosine of a number
exp(x)	Returns the value of E^x
floor(x)	Returns the value of a number rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of a number
max(x,y)	Returns the number with the highest value of x and y
min(x,y)	Returns the number with the lowest value of x and y
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 a number to the nearest integer
sin(x)	Returns the sine of a number
sqrt(x)	Returns the square root of a number
tan(x)	Returns the tangent of an angle
toSource()	Represents the source code of an object
valueOf()	Returns the primitive value of a Math object

Property	Description
constructor	A reference to the function that created the object
E	Returns Euler's constant (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14159)
prototype	Allows you to add properties and methods to the object
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

HTML DOM Object

The HTML DOM is a W3C standard and it is an abbreviation for the Document Object Model for HTML.
The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML documents.
All HTML elements, along with their containing text and attributes, can be accessed through the DOM. The contents can be modified or deleted, and new elements can be created.
The HTML DOM is platform and language independent. It can be used by any programming language like Java, JavaScript, and VBScript.

JavaScript Browser Detection

JavaScript includes an object called the Navigator object, which contains information about the visitor's browser name, browser version, and more.

```
<html>
<body>

<script type="text/javascript">
var x = navigator
document.write("CodeName=" + x.appCodeName)
document.write("<br />")
document.write("MinorVersion=" + x.appMinorVersion)
document.write("<br />")
document.write("Name=" + x.appName)
document.write("<br />")
document.write("Version=" + x.appVersion)
document.write("<br />")
document.write("CookieEnabled=" + x.cookieEnabled)
document.write("<br />")
document.write("CPUClass=" + x.cpuClass)
document.write("<br />")
document.write("OnLine=" + x.onLine)
document.write("<br />")
document.write("Platform=" + x.platform)
document.write("<br />")
document.write("UA=" + x.userAgent)
document.write("<br />")
document.write("BrowserLanguage=" + x.browserLanguage)
document.write("<br />")
document.write("SystemLanguage=" + x.systemLanguage)
document.write("<br />")
document.write("UserLanguage=" + x.userLanguage)
</script>

</body>
</html>
```

Output :

CodeName=Mozilla
MinorVersion=;SP2;
Name=Microsoft Internet Explorer
Version=4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
CookieEnabled=true
CPUClass=x86

OnLine=true
Platform=Win32
UA=Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
BrowserLanguage=en-us
SystemLanguage=en-us
UserLanguage=en-us

JavaScript Cookies

What is a Cookie?

A cookie is a variable that is stored on the visitor's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With JavaScript, you can both create and retrieve cookie values. Examples of cookies:

- **Name cookie** - The first time a visitor arrives to your web page, he or she must fill in her/his name. The name is then stored in a cookie. Next time the visitor arrives at your page, he or she could get a welcome message like "Welcome John Doe!" The name is retrieved from the stored cookie
- **Password cookie** - The first time a visitor arrives to your web page, he or she must fill in a password. The password is then stored in a cookie. Next time the visitor arrives at your page, the password is retrieved from the cookie
- **Date cookie** - The first time a visitor arrives to your web page, the current date is stored in a cookie. Next time the visitor arrives at your page, he or she could get a message like "Your last visit was on Tuesday August 11, 2005!" The date is retrieved from the stored cookie

In this example we will create a cookie that stores the name of a visitor. The first time a visitor arrives to the web page, he or she will be asked to fill in her/his name. The name is then stored in a cookie. The next time the visitor arrives at the same page, he or she will get welcome message.

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
	if (document.cookie.length>0)
	{
	  c_start=document.cookie.indexOf(c_name + "=")
	  if (c_start!=-1)
	  {
	       c_start=c_start + c_name.length+1
	       c_end=document.cookie.indexOf(";",c_start)
	       if (c_end==-1) c_end=document.cookie.length
	       return unescape(document.cookie.substring(c_start,c_end))
	  }
	}
	return null
}

function setCookie(c_name,value,expiredays)
{
	var exdate=new Date()
	exdate.setDate(expiredays)
	document.cookie=c_name+ "=" +escape(value)+
	((expiredays==null) ? "" : "; expires="+exdate)
}
```

```
function checkCookie()
{
       username=getCookie('username')
       if (username!=null)
         {alert('Welcome again '+username+'!')}
       else
         {
         username=prompt('Please enter your name:',"")
         if (username!=null||username!="")
           {
           setCookie('username',username,365)
           }
         }
}
</script>
</head>
<body onLoad="checkCookie()">
</body>
</html>
```

JavaScript Form Validation

JavaScript can be used to validate input data in HTML forms before sending off the content to a server. Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Required Fields:

```
<html>
<head>
<script type="text/javascript">
function validate_required(field,alerttxt)
{
       with (field)
       {
              if (value==null||value=="")
                {alert(alerttxt);return false}
              else {return true}
       }
}

function validate_form(thisform)
{
       with (thisform)
       {
              if (validate_required(email,"Email must be filled out!")==false)
                {email.focus();return false}
       }
}
</script>
</head>
<body>
<form action="submitpage.htm" onsubmit="return validate_form(this)" method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Email Validation:

```
<html>
<head>
```

```
<script type="text/javascript">
function validate_email(field,alerttxt)
{
        with (field)
        {
                apos=value.indexOf("@")
                dotpos=value.lastIndexOf(".")
                if (apos<1||dotpos-apos<2)
                  {alert(alerttxt);return false}
                else {return true}
        }
}

function validate_form(thisform)
{
        with (thisform)
        {
        if (validate_email(email,"Not a valid e-mail address!")==false)
          {email.focus();return false}
        }
}
</script>
</head>
<body>
<form action="submitpage.htm" onsubmit="return validate_form(this)" method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

JavaScript Animation

It is possible to use JavaScript to create animated images. The trick is to let a JavaScript change between different images on different events.

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.b1.src ="b_blue.gif"
}
function mouseOut()
{
document.b1.src ="b_pink.gif"
}
</script>
</head>

<body>
<a href="http://www.w3schools.com" target="_blank"
onmouseover="mouseOver()"
onmouseout="mouseOut()">
<img border="0" alt="Visit W3Schools!" src="b_pink.gif" name="b1" width="26" height="26"
/></a>
</body>
</html>
```

JavaScript Image Maps

An image-map is an image with clickable regions. We can add events (that can call a JavaScript) to the <area> tags inside the image map. The <area> tag supports the onClick, onDblClick, onMouseDown, onMouseUp, onMouseOver, onMouseMove, onMouseOut, onKeyPress, onKeyDown, onKeyUp, onFocus, and onBlur events.

```
<html>
```

```
<head>
<script type="text/javascript">
function writeText(txt)
{
document.getElementById("desc").innerHTML=txt
}
</script>
</head>
<body>
<img src="planets.gif" width="145" height="126"
alt="Planets" usemap="#planetmap" />

<map id ="planetmap" name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
onMouseOver="writeText('The Sun and the gas giant
planets like Jupiter are by far the largest objects
in our Solar System.')"
href ="sun.htm" target ="_blank" alt="Sun" />

<area shape ="circle" coords ="90,58,3"
onMouseOver="writeText('The planet Mercury is very
difficult to study from the Earth because it is
always so close to the Sun.')"
href ="mercur.htm" target ="_blank" alt="Mercury" />

<area shape ="circle" coords ="124,58,8"
onMouseOver="writeText('Until the 1960s, Venus was
often considered a twin sister to the Earth because
Venus is the nearest planet to us, and because the
two planets seem to share many characteristics.')"
href ="venus.htm" target ="_blank" alt="Venus" />
</map>

<p id="desc"></p>

</body>
</html>
```

JavaScript Timing Events

With JavaScript, it is possible to execute some code NOT immediately after a function is called, but after a specified time interval. This is called **timing events**.

- setTimeout() - executes a code some time in the future
- clearTimeout() - cancels the setTimeout()

```
var t=setTimeout("javascript statement",milliseconds)
```
var t=setTimeout("alert('5 seconds!')",5000)

JavaScript Create Your Own Objects

Create a direct instance of an Object:

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

```
personObj.eat=eat
```

where, eat() is a method.

Create a template of an object:

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname
this.lastname=lastname
this.age=age
this.eyecolor=eyecolor

this.newlastname=newlastname

}

function newlastname(new_lastname)
{
this.lastname=new_lastname
}


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

myMother.newlastname("Doe").
```

DHTML

DHTML is a combination of HTML, CSS, and JavaScript. DHTML is used to create dynamic and interactive Web sites.