Java Script

Java Script

Contact: 9598948810,9532878816, Website: infomaxacademy.com

Notes By: Dinesh Maurya

Introduction to JavaScript

JavaScript is the scripting language of the Web that **i**nserted in HTML documents, and make pages more dynamic and interactive.

JavaScript is used in millions of Web pages to improve the design, validate forms, and much more. JavaScript was developed by Netscape and is the most popular scripting language on the internet.

Application of JavaScript

- **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
- **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
- **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 will save the server from extra processing

Common HTML Events

Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed

onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Writing JavaScript

Scripts in the body section will be executed WHILE the page loads.

Scripts in the head section will be executed when CALLED.

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">
    some statements
</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">
    some statements
</script>
</body>
```

Scripts in both the body and the head section: You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

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

How to Run an External JavaScript

Sometimes you might want to run the same script on several pages, without writing the script on each and every page.

To simplify this you can write a script in an external file, and save it with a .js file extension, like this:

```
document.write("This script is external")
```

Save the external file as myjsfile.js.

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

Now you can call this script, using the "src" attribute, from any of your pages:

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

Remember to place the script exactly where you normally would write the script.

JavaScript Panel(model)/Window

Examples

Alert box
How to display an alert box.

```
<script type="text/javascript">
alert("Hello World!")
</script>
```

Confirm box
How to display a confirm box.

```
<script type="text/javascript">
var name = confirm("Press a button");
if (name == true)
{
document.write("You pressed OK");
}
```

```
else
{
document.write("You pressed Cancel");

}
</script>
```

Prompt box

```
<script type="text/javascript">
var name = prompt("Please enter your name","");
if (name != null && name != "")
{
document.write("Hello " + name);
}
</script>
```

Open a new window when clicking on a button

How you can display a new window.

```
<script language=javascript>
function openwindow()
{
window.open("http://www.schools.com");
}
</script>
```

Call a function in specific event:

```
<input type=button value="Open Window" onclick="openwindow()">
```

Location

How to send the client to a new location (URL/page).

```
<script type="text/javascript">
function locate()
{
location=”http://www.schools.com/”;
}
</script>
```

Call a function in specific event:

```
<input type="button" onclick="locate()" value="New location">
```

Refresh

```
<script type="text/javascript">
function refresh()
{
location.reload();
}
</script>
```

Call a function in specific Event:

```
<input type="button" value="Refresh" onclick="refresh()">
```

Status bar

How to write some text in the windows status bar.

```
<script type="text/javascript">
function load()
```

```
{
window.status = "put your message here";
}
</script>
```

Call function in page load event:

```
<body onload="load()">
```

Print page

```
<script type="text/javascript">
function printpage()
{
window.print()
}
</script>
```

Call a function in specific Event

```
<input type="button" value="Print this page" onclick="printpage()">
```

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

Declare a Variable

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
```

Assign a Value to a Variable

You assign a value to a variable like this:

```
var strname = "Hege"
```

Or like this:

```
strname = "Hege"
```

JavaScript Operators

Operators are used to operate on values.

Arithmetic Operators

+ , - , *, / ,%

Assignment Operators

= ,+= , -= , *= , / =, %=

Comparison Operators

= = , > , < , <= , >= , !=

Logical Operators

&& , || , !

String Operator

A string is most often text, for example "Hello World!". To stick two or more string variables together, use the + operator.

```
txt1="What a very"

txt2="nice day!"

txt3=txt1+txt2
```

The variable txt3 now contains "What a verynice day!".

To add a space between two string variables, insert a space into the expression, OR in one of the strings.

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

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

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename=(condition)?value1:value2
```

Example

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

If the variable visitor is equal to PRES, then put the string "Dear President " in the variable named greeting. If the variable visitor is not equal to PRES, then put the string "Dear " into the variable named greeting.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have three conditional statements:

- **if statement** - use this statement if you want to execute a set of code when a condition is true
- **if...else statement** - use this statement if you want to select one of two sets of lines to execute
- **switch statement** - use this statement if you want to select one of many sets of lines to execute

If Statement

You should use the if statement if you want to execute some code if a condition is true.

Syntax

```
if (condition)
{
code to be executed if condition is true
}
```

Example

```
<script type="text/javascript">
//If the time on your browser is less than 10,
//you will get a "Good morning" greeting.
var d=new Date()
var time=d.getHours()
```

```
if (time<10)
{
document.write("<b>Good morning</b>")
}
</script>
```

Notice that there is no ..else.. in this syntax. You just tell the code to execute some code **if the condition is true**.

If Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

```
if (condition)
{
code to be executed if condition is true
}
else
{
code to be executed if condition is false
}
```

Example

```
<script type="text/javascript">
//If the time on your browser is less than 10,
//you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.
var d = new Date()
var time = d.getHours()
if (time < 10)
{
document.write("Good morning!")
}
else
{
```

```
document.write("Good day!")
}
</script>
```

Switch Statement

You should use the Switch statement if you want to select one of many blocks of code to be executed.

Syntax

```
switch (expression)
{
case label1:
  code to be executed if expression = label1
  break
case label2:
  code to be executed if expression = label2
  break
default:
  code to be executed
  if expression is different
  from both label1 and label2
}
```

This is how it works: First we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

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 Looping

Looping statements in JavaScript are used to execute the same block of code a specified number of times.

Examples

For loop
How to write a For loop. Use a For loop to run the same block of code a specified number of times

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

While loop

How to write a While loop. Use a While loop to run the same block of code while or until a condition is true

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

Do while loop

How to write a Do While loop. Use a Do While loop to run the same block of code while or until a condition is true. This loop will always be executed once, even if the condition is false, because the statements are executed before the condition is tested

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

JavaScript Array Object

An Array object is used to store a set of values in a single variable name.

Examples

```
<html>
<body>
<script type="text/javascript">
var famname = new Array(6)
famname[0] = "Jan Egil"
famname[1] = "Tove"
famname[2] = "Hege"
famname[3] = "Stale"
famname[4] = "Kai Jim"
famname[5] = "Borge"
for (i=0; i<6; i++)
{
document.write(famname[i] + "<br>")
}
</script>
</body></html>
```

Initialization of an Array

This is another way of making an array that gives the same result as the one above. Note that the length method is used to find out how many elements the array contains.

```
<html>
<body>
<script type="text/javascript">
var famname = new Array("Jan Egil","Tove","Hege","Stale","Kai
Jim","Borge")
for (i=0; i<famname.length; i++)
{
document.write(famname[i] + "<br>")
```

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

Methods Of Array:

Methods	**Explanation**
length	Returns the number of elements in an array. This property is assigned a value when an array is created
concat()	Returns an array concatenated of two arrays
join()	Returns a string of all the elements of an array concatenated together
reverse()	Returns the array reversed
Slice()	Returns a specified part of the array
sort()	Returns a sorted array

Functions

A function contains some code that will be executed by an event or a call to that function. A function is a set of statements. You can reuse functions within the same script, or in other documents. You define functions at the beginning of a file (in the head section), and call them later in the document. It is now time to take a lesson about the alert-box:

This is JavaScript's method to alert the user.

```
alert("This is a message")
```

How to Define a Function

To create a function you define its name, any values ("arguments"), and some statements:

```
function myfunction(argument1,argument2,etc)
{
some statements
}
```

A function with no arguments must include the parentheses:

```
function myfunction()
{
some statements
}
```

Arguments are variables used in the function. The variable values are values passed on by the function call.

By placing functions in the head section of the document, you make sure that all the code in the function has been loaded before the function is called.

Some functions return a value to the calling expression

```
function result(a,b)
{
c=a+b
return c
}
```

How to Call a Function

A function is not executed before it is called.

You can call a function containing arguments:

```
myfunction(argument1,argument2,etc)
```

or without arguments:

```
myfunction()
```

The return Statement

Functions that will return a result must use the "return" statement. This statement specifies the value which will be returned to where the function was called from. Say you have a function that returns the sum of two numbers:

```
function total(a,b)
{
result=a+b
return result
}
```

When you call this function you must send two arguments with it:

```
sum=total(2,3)
```

The returned value from the function (5) will be stored in the variable called sum.

JavaScript Function List:

Math Function

Methods	**Explanation**
abs(x)	Returns the absolute value of x
ceil(x)	Returns the nearest integer greater than or equal to x
cos(x)	Returns the cosine of x
exp(x)	Returns the value of E raised to the power of x
floor(x)	Returns the nearest integer less than or equal to x
log(x)	Returns the natural log of x
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 the number x raised to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sqrt(x)	Returns the square root of x

String Function:

Properties	**Explanation**
length	Returns the number of characters in a string

Methods	Explanation
charAt()	Returns the character at a specified position
concat()	Returns two concatenated strings
indexOf()	Returns the position of the first occurrence of a specified string inside another string. Returns -1 if it never occurs
lastIndexOf()	Returns the position of the first occurrence of a specified string inside another string. Returns -1 if it never occurs. Note: This method starts from the right and moves left!
match()	Similar to indexOf and lastIndexOf, but this method returns the specified string, or "null", instead of a numeric value
replace()	Replaces some specified characters with some new specified characters
search()	Returns an integer if the string contains some specified characters, if not it returns -1
slice()	Returns a string containing a specified character index
split()	Splits a string into an array of strings
strike()	Returns a string strikethrough
sub()	Returns a string as subscript
substr()	Returns the specified characters. 14,7 returns 7 characters, from the 14th character (starts at 0)
substring()	Returns the specified characters. 7,14 returns all characters from the 7th up to but not including the 14th (starts at 0)
sup()	Returns a string as superscript
toLowerCase()	Converts a string to lower case

toUpperCase()	Converts a string to upper case