Introduction to JavaScript

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, and Opera.

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, 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

Java and JavaScript are two completely different languages in both concept and design!

Advantages

- **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 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

The Real Name is ECMAScript

JavaScript's official name is "ECMAScript". The standard is developed and maintained by the **ECMA organisation.**

The language was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all Netscape and Microsoft browsers since 1996.

The standard was approved as an international ISO (ISO/IEC 16262) standard in 1998.

The development of the standard is still in progress.

Example Explained

To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and ends:

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

The word **document.write** is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

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

</head>
<body>
</body>
</html>
```

Note: If we had not entered the <script> tag, the browser would have treated the document.write("Hello World!") command as pure text, and just write the entire line on the page.

JavaScript Where To

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

JavaScripts in the head section will be executed when CALLED.

JavaScripts in a body will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

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>
```

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">
....
</script>
</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:

JavaScript is a sequence of statements to be executed by the browser.

JavaScript is case sensitive - therefore watch your capitalization closely when you write

JavaScript statements, create or call variables, objects and functions.

It is normal to add a semicolon at the end of each executable statement.

Using semicolons makes it possible to write multiple statements on one line

JavaScript Statement

A JavaScript statement is a command to the browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

```
document.write("Hello Dolly");
```

JavaScript Code

JavaScript code (or just JavaScript) is a sequence of JavaScript statements.

Each statement is executed by the browser in the sequence they are written.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks.

Blocks start with a left curly bracket {, and ends with a right curly bracket }.

The purpose of a block is to make the sequence of statements execute together.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">
{
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
}
</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

JavaScript Comments

Comments can be added to explain the JavaScript, or to make it more readable.

Single line comments start with //.

This example uses single line comments to explain the code:

```
<script type="text/javascript">
// This will write a header:
document.write("<h1>This is a header</h1>");
// This will write two paragraphs:
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

JavaScript Multi-Line Comments

Multi line comments start with /* and end with */.

This example uses a multi line comment to explain the code:

```
<script type="text/javascript">
/*
The code below will write
one header and two paragraphs
*/
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

Using Comments to Prevent Execution

In this example the comment is used to prevent the execution of a single code line:

```
<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
//document.write("<p>This is another paragraph</p>");
</script>
```

In this example the comments is used to prevent the execution of multiple code lines:

```
<script type="text/javascript">
/*
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
*/
</script>
```

Using Comments at the End of a Line

In this example the comment is placed at the end of a line:

```
<script type="text/javascript">
document.write("Hello"); // This will write "Hello"
document.write("Dolly"); // This will write "Dolly"
</script>
```

JavaScript Variables

Variables are "containers" for storing information.

JavaScript variables are used to hold values or expressions.

Rules for JavaScript variable names:

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

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var statement**:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared.

These statements:

```
x=5;
carname="Volvo";
```

have the same effect as:

```
var x=5;
var carname="Volvo";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;
var x;
```

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Operators

The operator = is used to assign values.

The operator + is used to add values.

The assignment operator = is used to assign values to JavaScript variables.

The arithmetic operator + is used to add values together.

```
y=5;
z=2;
x=y+z;
```

The value of x, after the execution of the statements above is 7.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10

/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

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

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

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

or insert a space into the expression:

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

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

Look at these examples:

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

Try it yourself.

The rule is:

If you add a number and a string, the result will be a string.

JavaScript Comparison and Logical Operators

Previous Next

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

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

How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) document.write("Too young");
```

You will learn more about the use of conditional statements in the next chapter of this tutorial.

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true

||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

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** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear".

JavaScript If...Else Statements

Previous Next

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Examples

If statement
How to write an if statement.

If...else statement
How to write an if...else statement.

If..else if...else statement
How to write an if..else if...else statement.

Random link
This example demonstrates a link, when you click on the link it will take you to W3Schools.com OR to RefsnesData.no. There is a 50% chance for each of them.

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 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

If Statement

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

Syntax

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

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example 1

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();

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

```
</script>
```

Example 2

```
<script type="text/javascript">
//Write "Lunch-time!" if the time is 11
var d=new Date();
var time=d.getHours();

if (time==11)
{
document.write("<b>Lunch-time!</b>");
}
</script>
```

Note: When **comparing** variables you must always use two equals signs next to each other (==)!

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

If...else Statement

If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

Syntax

```
if (condition)
{
code to be executed if condition is true
}
else
{
code to be executed if condition is not true
}
```

Example

```
<script type="text/javascript">
//If the time 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>
```

If...else if...else Statement

You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

Syntax

```
if (condition1)
{
code to be executed if condition1 is true
}
else if (condition2)
{
code to be executed if condition2 is true
}
else
{
code to be executed if condition1 and
condition2 are not true
}
```

Example

```
<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>
```

<html>

```
<body>

<script type="text/javascript">
var d = new Date();
var time = d.getHours();

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

<p>
This example demonstrates the If statement.
</p>

<p>
If the time on your browser is less than 10,
you will get a "Good morning" greeting.
</p>

</body>
</html>

<html>
<body>
```

```
<script type="text/javascript">
var d = new Date();
var time = d.getHours();

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

<p>
This example demonstrates the If...Else statement.
</p>

<p>
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.
</p>

</body>
```

```
</html>

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>

<p>
This example demonstrates the if..else if...else statement.
</p>
```

```
</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='http://www.refsnesdata.no'>Visit Refsnes Data!</a>");

}

</script>


</body>

</html>
```

JavaScript Switch Statement

Previous Next

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Examples

Switch statement
How to write a switch statement.

The JavaScript Switch Statement

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

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is
  different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (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>
```

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

This JavaScript will generate a different greeting based on what day it is. Note that Sunday=0, Monday=1, Tuesday=2, etc.

</body>

</html>

JavaScript Popup Boxes

Previous Next

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

Examples

Alert box

Alert box with line breaks

Confirm box

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.

Syntax:

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

Confirm Box

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.

Syntax:

```
confirm("sometext");
```

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.

Syntax:

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

```
<html>

<head>

<script type="text/javascript">

function disp_alert()

{

alert("I am an alert box!!");

}

</script>

</head>

<body>


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

```
</body>
</html>

<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>

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

</body>
</html>

<html>
<head>
<script type="text/javascript">
function disp_confirm()
{
var r=confirm("Press a button");
```

```
if (r==true)
  {
  document.write("You pressed OK!");
  }
else
  {
  document.write("You pressed Cancel!");
  }
}
</script>
</head>
<body>

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

</body>
</html>

<html>
<head>
<script type="text/javascript">
function disp_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
```

```
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>

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

</body>
</html>
```

JavaScript Functions

Previous Next

A function is a reusable code-block that will be executed by an event, or when the function is called.

Examples

Function
How to call a function.

Function with arguments
How to pass a variable to a function, and use the variable in the function.

Function with arguments 2
How to pass variables to a function, and use these variables in the function.

Function that returns a value
How to let the function return a value.

A function with arguments, that returns a value
How to let the function find the product of two arguments and return the result.

JavaScript 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 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 can be defined both in the <head> and in the <body> section of a document. However, to assure that the function is read/loaded by the browser before it is called, it could be wise to put it in the <head> section.

Example

```
<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 put 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.

You will learn more about JavaScript events in the JS Events chapter.

How to Define a Function

The syntax for creating a function is:

```
function functionname(var1,var2,...,varX)
{
some code
}
```

var1, var2, etc are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name:

```
function functionname()
{
some code
}
```

Note: 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

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

So, functions that are going to return a value must use the return statement.

Example

The function below should return the product of two numbers (a and b):

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

When you call the function above, you must pass along two parameters:

```
product=prod(2,3);
```

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

The Lifetime of JavaScript 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. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

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.

```
<html>
<head>

<script type="text/javascript">
function myfunction()
{
alert("HELLO");
}
</script>

</head>
<body>

<form>
<input type="button"
```

```
onclick="myfunction()"

value="Call function">

</form>


<p>By pressing the button, a function will be called. The function will alert a message.</p>


</body>

</html>



<html>

<head>


<script type="text/javascript">

function myfunction(txt)

{

alert(txt);

}

</script>


</head>

<body>


<form>

<input type="button"

onclick="myfunction('Hello')"
```

```
value="Call function">

</form>


<p>By pressing the button, a function with an argument will be called. The function
will alert

this argument.</p>


</body>

</html>


<html>

<head>

<script type="text/javascript">

function myfunction(txt)

{

alert(txt);

}

</script>

</head>


<body>

<form>

<input type="button"

onclick="myfunction('Good Morning!')"

value="In the Morning">


<input type="button"
```

```
onclick="myfunction('Good Evening!')"

value="In the Evening">

</form>


<p>

When you click on one of the buttons, a function will be called. The function will alert

the argument that is passed to it.

</p>


</body>

</html>


<html>

<head>


<script type="text/javascript">

function myFunction()

{

return ("Hello, have a nice day!");

}

</script>


</head>

<body>


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

```
document.write(myFunction())
</script>

<p>The script in the body section calls a function.</p>

<p>The function returns a text.</p>

</body>
</html>
<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>

<p>The script in the body section calls a function with two parameters (4 and 3).</p>
<p>The function will return the product of these two parameters.</p>
```

```
</body>

</html>
```

JavaScript For Loop

Previous Next

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

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.

Looping through HTML headers
How to use the for loop to loop through the different HTML headers.

JavaScript Loops

Very often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
    code to be executed
}
```

Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 10. **i** will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

```
<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>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

The while loop

The while loop will be explained in the next chapter.

<html>

<body>

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

for (i = 0; i <= 5; i++)

{

document.write("The number is " + i);

document.write("<br />");

}

</script>


<p>Explanation:</p>


<p>This for loop starts with i=0.</p>


<p>As long as <b>i</b> is less than, or equal to 5, the loop will continue to
run.</p>


<p><b>i</b> will increase by 1 each time the loop runs.</p>


</body>

</html>

<html>

<body>


<script type="text/javascript">

for (i = 1; i <= 6; i++)

{

document.write("<h" + i + ">This is header " + i);
```

```
document.write("</h" + i + ">");

}

</script>


</body>

</html>
```

JavaScript While Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

The while loop

The while loop is used when you want the loop to execute and continue executing while the specified condition is true.

```
while (var<=endvalue)
{
    code to be executed
}
```

Note: The <= could be any comparing statement.

Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 10. **i** will increase by 1 each time the loop runs.

```
<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>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true. This loop will always be executed at least once, even if the condition is false, because the code is executed before the condition is tested.

```
do
{
    code to be executed
}
while (var<=endvalue);
```

Example

```
<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>
```

Result

```
The number is 0
```

```
<html>
<body>

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

<p>Explanation:</p>

<p><b>i</b> is equal to 0.</p>

<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>

<p><b>i</b> will increase by 1 each time the loop runs.</p>
```

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

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

<p>Explanation:</p>

<p><b>i</b>  equal to 0.</p>

<p>The loop will run</p>

<p><b>i</b> will increase by 1 each time the loop runs.</p>

<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>
```

JavaScript Break and ContinueThere are two special statements that can be used inside loops: break and continue.

JavaScript break and continue Statements

There are two special statements that can be used inside loops: break and continue.

Break

The break command will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<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>
```

Result

```
The number is 0
The number is 1
The number is 2
```

Continue

The continue command will break the current loop and continue with the next value.

Example

```
<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>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

```
<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>
<p>Explanation: The loop will break when i=3.</p>
</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>
```

Explanation: The loop will break the current loop and continue with the next value when i=3.

</body>

</html>

JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

The code in the body of the for ... in loop is executed once for each element/property.

Syntax

```
for (variable in object)
{
    code to be executed
}
```

The variable argument can be a named variable, an array element, or a property of an object.

Example

Using for...in to loop through an array:

```
<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>
```

<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] + "
");

}

</script>

</body>

</html>

JavaScript Events

Events are actions that can be detected by JavaScript.

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 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.

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 box 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!

For a complete reference of the events recognized by JavaScript, go to our complete Event reference.

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.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<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 try...catch statement allows you to test a block of code for errors.

JavaScript - Catching Errors

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.

This chapter will teach you how to trap and handle JavaScript error messages, so you don't lose your audience.

There are two ways of catching errors in a Web page:

- By using the **try...catch** statement (available in IE5+, Mozilla 1.0, and Netscape 6)
- By using the **onerror** event. This is the old standard solution to catch errors (available since Netscape 3)

Try...Catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

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

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example 1

The example below contains a script that is supposed to display the message "Welcome guest!" when you click on a button. However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs:

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

```
<script type="text/javascript">
function message()
{
adddlert("Welcome guest!");
}
</script>
</head>

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

</html>
```

To take more appropriate action when an error occurs, you can add a 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>
```

Example 2

The next example uses a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method

returns false, the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing:

```
<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>
```

The onerror Event

The onerror event will be explained soon, but first you will learn how to use the throw statement to create an exception. The throw statement can be used together with the try...catch statement.

<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>

<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>
```

JavaScript Throw Statement

The throw statement allows you to create an exception.

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.

Syntax

```
throw(exception)
```

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

Note that throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example 1

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 too high");
if(er == "Err2")
alert("Error! The value is too low");
}
</script>
</body>
</html>
```

```
<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";
  }
else if(isNaN(x))
  {
  throw "Err3";
  }
}
catch(er)
{
if(er=="Err1")
  {
  alert("Error! The value is too high");
  }
if(er=="Err2")
```

```
  {
  alert("Error! The value is too low");
  }
if(er=="Err3")
  {
  alert("Error! The value is not a number");
  }
}
</script>
</body>
</html>
```

JavaScript The onerror Event

Using the onerror event is the old standard solution to catch errors in a web page.

The onerror Event

We have just explained how to use the try...catch statement to catch errors in a web page. Now we are going to explain how to use the onerror event for the same purpose.

The onerror event is fired whenever there is a script error in the page.

To use the onerror event, you must create a function to handle the errors. Then you call the function with the onerror event handler. The event handler is called with three arguments: msg (error message), url (the url of the page that caused the error) and line (the line where the error occurred).

Syntax

```
onerror=handleErr
function handleErr(msg,url,l)
{
//Handle the error here
```

```
return true or false
}
```

The value returned by onerror determines whether the browser displays a standard error message. If you return false, the browser displays the standard error message in the JavaScript console. If you return true, the browser does not display the standard error message.

Example

The following example shows how to catch the error with the onerror event:

```
<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>
```

<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.

Insert Special Characters

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Look at the following JavaScript code:

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: We are the so-called

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north.

Here is another example:

```
document.write ("You \& I are singing!");
```

The example above will produce the following output:

```
You & I are singing!
```

The table below lists other special characters that can be added to a text string with the backslash sign:

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return

\t	tab
\b	backspace
\f	form feed

JavaScript Guidelines

Some other important things to know when scripting with JavaScript.

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".

JavaScript is case sensitive - therefore watch your capitalization closely when you create or call variables, objects and functions.

White Space

JavaScript ignores extra spaces. You can add white space to your script to make it more readable. The following lines are equivalent:

```
name="Hege";
name = "Hege";
```

Break up a Code Line

You can break up a code line **within a text string** with a backslash. The example below will be displayed properly:

```
document.write("Hello \
World!");
```

However, you cannot break up a code line like this:

```
document.write \
("Hello World!");
```

JavaScript Objects Introduction

JavaScript is an Object Oriented Programming (OOP) language.

An OOP language allows you to define your own objects and make your own variable types.

Object Oriented Programming

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.

Note that an object is just a special kind of data. An object has properties and methods.

Properties

Properties are the values associated with an object.

In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.length);
</script>
```

The output of the code above will be:

```
12
```

Methods

Methods are the actions that can be performed on objects.

In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

The output of the code above will be:

```
HELLO WORLD!
```

JavaScript String Object

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

Complete String Object Reference

For a complete reference of all the properties and methods that can be used with the String object, go to our complete String object reference.

The reference contains a brief description and examples of use for each property and method!

String object

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

Examples of use:

The following example uses the length property of the String object to find the length of a string:

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

The code above will result in the following output:

```
12
```

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

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

The code above will result in the following output:

```
HELLO WORLD!
```

```
<html>

<body>


<script type="text/javascript">


var txt="Hello World!";

document.write(txt.length);


</script>


</body>
```

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

<script type="text/javascript">

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>Blink: " + txt.blink() + " (does not work in IE)</p>");
document.write("<p>Fixed: " + txt.fixed() + "</p>");
document.write("<p>Strike: " + txt.strike() + "</p>");

document.write("<p>Fontcolor: " + txt.fontcolor("Red") + "</p>");
document.write("<p>Fontsize: " + txt.fontsize(16) + "</p>");

document.write("<p>Lowercase: " + txt.toLowerCase() + "</p>");
document.write("<p>Uppercase: " + txt.toUpperCase() + "</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>");
</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">

var str="Hello world!";
document.write(str.indexOf("Hello") + "<br />");
document.write(str.indexOf("World") + "<br />");
document.write(str.indexOf("world"));

</script>

</body>
</html>

<html>
<body>

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

```
var str="Hello world!";
document.write(str.match("world") + "<br />");
document.write(str.match("World") + "<br />");
document.write(str.match("worlld") + "<br />");
document.write(str.match("world!"));

</script>

</body>
</html>
<html>
<body>

<script type="text/javascript">

var str="Visit Microsoft!";
document.write(str.replace(/Microsoft/,"W3Schools"));

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

JavaScript Date Object

The Date object is used to work with dates and times.

Complete Date Object Reference

For a complete reference of all the properties and methods that can be used with the Date object, go to our complete Date object reference.

The reference contains a brief description and examples of use for each property and method!

Create a Date Object

The Date object is used to work with dates and times.

The following code create a Date object called myDate:

```
var myDate=new Date()
```

Note: The Date object will automatically hold the current date and time as its initial value!

Set Dates

We can easily manipulate the date by using the methods available for the Date object.

In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

Compare Two Dates

The Date object is also used to compare two dates.

The following example compares today's date with the 14th January 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");
}
```

<html>

<body>

<script type="text/javascript">

document.write(Date());

</script>

</body>

</html>

<html>

<body>

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

var minutes = 1000*60;
var hours = minutes*60;
var days = hours*24;
var years = days*365;
var d = new Date();
var t = d.getTime();
var y = t/years;

document.write("It's been: " + y + " years since 1970/01/01!");

</script>

</body>
</html>
<html>
<body>

<script type="text/javascript">

var d = new Date();
d.setFullYear(1992,10,3);
document.write(d);

</script>
```

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

<script type="text/javascript">

var d = new Date();
document.write (d.toUTCString());

</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">

var d=new Date();
var weekday=new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
```

```
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";

document.write("Today it is " + weekday[d.getDay()]);

</script>

</body>
</html>

<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
```

```
}

function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

JavaScript Array Object

The Array object is used to store multiple values in a single variable.

Complete Array Object Reference

For a complete reference of all the properties and methods that can be used with the Array object, go to our complete Array object reference.

The reference contains a brief description and examples of use for each property and method!

Create an Array

The following code creates an Array object called myCars:

```
var myCars=new Array()
```

There are two ways of adding values to an array (you can add as many values as you need to define as many variables you require).

1:

```
var myCars=new Array();
mycars[0]="Saab";
mycars[1]="Volvo";
mycars[2]="BMW";
```

You could also pass an integer argument to control the array's size:

```
var myCars=new Array(3);
mycars[0]="Saab";
mycars[1]="Volvo";
mycars[2]="BMW";
```

2:

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

Note: If you specify numbers or true/false values inside the array then the type of variables will be numeric or Boolean instead of string.

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Saab
```

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Opel
```

```
<html>
<body>

<script type="text/javascript">
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";

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

```
}
</script>

</body>
</html>
<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>
<html>
<body>

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

```
var arr = new Array(3);
arr[0] = "Jani";
arr[1] = "Tove";
arr[2] = "Hege";

var arr2 = new Array(3);
arr2[0] = "John";
arr2[1] = "Andy";
arr2[2] = "Wendy";

document.write(arr.concat(arr2));

</script>

</body>
</html>
<html>
<body>

<script type="text/javascript">

var arr = new Array(3);
arr[0] = "Jani";
arr[1] = "Hege";
arr[2] = "Stale";
```

```
document.write(arr.join() + "<br />");
document.write(arr.join("."));

</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">

var arr = new Array(6);
arr[0] = "Jani";
arr[1] = "Hege";
arr[2] = "Stale";
arr[3] = "Kai Jim";
arr[4] = "Borge";
arr[5] = "Tove";

document.write(arr + "<br />");
document.write(arr.sort());
</script>

</body>
```

```
</html>

<html>
<body>

<script type="text/javascript">

function sortNumber(a, b)
{
return a - b;
}

var arr = new Array(6);
arr[0] = "10";
arr[1] = "5";
arr[2] = "40";
arr[3] = "25";
arr[4] = "1000";
arr[5] = "1";

document.write(arr + "<br />");
document.write(arr.sort(sortNumber));

</script>

</body>
```

</html>

JavaScript Boolean Object

The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).

Complete Boolean Object Reference

For a complete reference of all the properties and methods that can be used with the Boolean object, go to our complete Boolean object reference.

The reference contains a brief description and examples of use for each property and method!

Create a Boolean Object

The Boolean object represents two values: "true" or "false".

The following code creates a Boolean object called myBoolean:

```
var myBoolean=new Boolean();
```

Note: If the Boolean object has no initial value or if it is 0, -0, null, "", false, undefined, or NaN, the object is set to false. Otherwise it is true (even with the string "false")!

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");
```

```
<html>
<body>

<script type="text/javascript">
var b1=new Boolean( 0);
var b2=new Boolean(1);
var b3=new Boolean("");
var b4=new Boolean(null);
var b5=new Boolean(NaN);
var b6=new Boolean("false");

document.write("0 is boolean "+ b1 +"<br />");
document.write("1 is boolean "+ b2 +"<br />");
document.write("An empty string is boolean "+ b3 + "<br />");
document.write("null is boolean "+ b4+ "<br />");
document.write("NaN is boolean "+ b5 +"<br />");
document.write("The string 'false' is boolean "+ b6 +"<br />");
</script>

</body>
```

</html>

JavaScript Math Object

The Math object allows you to perform mathematical tasks.

Complete Math Object Reference

For a complete reference of all the properties and methods that can be used with the Math object, go to our complete Math object reference.

The reference contains a brief description and examples of use for each property and method!

Math Object

The Math object allows you to perform mathematical tasks.

The Math object includes several mathematical constants and methods.

Syntax for using properties/methods of Math:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Note: Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

Mathematical Constants

JavaScript provides eight mathematical constants that can be accessed from the Math object. These are: E, PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, and base-10 log of E.

You may reference these constants from your JavaScript like this:

```
Math.E
Math.PI
Math.SQRT2
Math.SQRT1_2
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
```

Mathematical Methods

In addition to the mathematical constants that can be accessed from the Math object there are also several methods available.

The following example uses the round() method of the Math object to round a number to the nearest integer:

```
document.write(Math.round(4.7));
```

The code above will result in the following output:

```
5
```

The following example uses the random() method of the Math object to return a random number between 0 and 1:

```
document.write(Math.random());
```

The code above can result in the following output:

```
0.06312843761329368
```

The following example uses the floor() and random() methods of the Math object to return a random number between 0 and 10:

```
document.write(Math.floor(Math.random()*11));
```

The code above can result in the following output:

```
3
```

```
<html>
<body>

<script type="text/javascript">

document.write(Math.round(0.60) + "<br />");
document.write(Math.round(0.50) + "<br />");
document.write(Math.round(0.49) + "<br />");
document.write(Math.round(-4.40) + "<br />");
document.write(Math.round(-4.60));

</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">

document.write(Math.random());

</script>

</body>
```

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

<script type="text/javascript">

document.write(Math.max(5,7) + "<br />");
document.write(Math.max(-3,5) + "<br />");
document.write(Math.max(-3,-5) + "<br />");
document.write(Math.max(7.25,7.30));

</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">

document.write(Math.min(5,7) + "<br />");
document.write(Math.min(-3,5) + "<br />");
document.write(Math.min(-3,-5) + "<br />");
document.write(Math.min(7.25,7.30));
```

</script>

</body>

</html>

JavaScript RegExp Object

The RegExp object is used to specify what to search for in a text

What is RegEp

RegExp, is short for regular expression.

When you search in a text, you can use a pattern to describe what you are searching for. **RegExp IS this pattern**.

A simple pattern can be a single character.

A more complicated pattern consists of more characters, and can be used for parsing, format checking, substitution and more.

You can specify where in the string to search, what type of characters to search for, and more.

Defining RegExp

The RegExp object is used to store the search pattern.

We define a RegExp object with the *new* keyword. The following code line defines a RegExp object called patt1 with the pattern "e":

```
var patt1=new RegExp("e");
```

When you use this RegExp object to search in a string, you will find the letter "e".

Methods of the RegExp Object

The RegExp Object has 3 methods: test(), exec(), and compile().

test()

The test() method searches a string for a specified value. Returns true or false

Example:

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
true
```

Try it yourself

exec()

The exec() method searches a string for a specified value. Returns the text of the found value. If no match is found, it returns *null*

Example 1:

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
e
```

Example 2:

You can add a second parameter to the RegExp object, to specify your search. For example; if you want to find all occurrences of a character, you can use the "g" parameter ("global").

For a complete list of how to modify your search, visit our complete RegExp object reference.

When using the "g" parameter, the exec() method works like this:

- Finds the first occurence of "e", and stores its position
- If you run exec() again, it starts at the stored position, and finds the next occurence of "e", and stores its position

```
var patt1=new RegExp("e","g");
do
{
result=patt1.exec("The best things in life are free");
document.write(result);
}
while (result!=null)
```

Since there is six "e" letters in the string, the output of the code above will be:

```
eeeeeenull
```

compile()

The compile() method is used to change the RegExp.

compile() can change both the search pattern, and add or remove the second parameter.

Example:

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
patt1.compile("d");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, but not a "d", the output of the code above will be:

```
truefalse
```

Complete RegExp Object Reference

For a complete reference of all the properties and methods that can be used with the RegExp object, go to our complete RegExp object reference.

The reference contains a brief description and examples of use for each property and method including the string object

```
<html>
<body>

<script type="text/javascript">
var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));
</script>

</body>
</html>
<html>
<body>

<script type="text/javascript">
var patt1=new RegExp("e");

document.write(patt1.exec("The best things in life are free"));
</script>
```

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

<script type="text/javascript">
var patt1=new RegExp("e","g");

do
{
result=patt1.exec("The best things in life are free");
document.write(result);
}
while (result!=null)
</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

```
patt1.compile("d");


document.write(patt1.test("The best things in life are free"));
</script>


</body>
</html>
```

JavaScript HTML DOM Objects

In addition to the built-in JavaScript objects, you can also access and manipulate all of the HTML DOM objects with JavaScript.

More JavaScript Objects

Follow the links to learn more about the objects and their collections, properties, methods and events.

Object	Description
Window	The top level object in the JavaScript hierarchy. The Window object represents a browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag
Navigator	Contains information about the client's browser
Screen	Contains information about the client's display screen
History	Contains the visited URLs in the browser window
Location	Contains information about the current URL

The HTML DOM

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.

Follow the links below to learn more about how to access and manipulate each DOM object with JavaScript:

Object	**Description**
Document	Represents the entire HTML document and can be used to access all elements in a page
Anchor	Represents an <a> element
Area	Represents an <area> element inside an image-map
Base	Represents a <base> element
Body	Represents the <body> element
Button	Represents a <button> element
Event	Represents the state of an event
Form	Represents a <form> element
Frame	Represents a <frame> element
Frameset	Represents a <frameset> element
Iframe	Represents an <iframe> element
Image	Represents an <img> element
Input button	Represents a button in an HTML form
Input checkbox	Represents a checkbox in an HTML form
Input file	Represents a fileupload in an HTML form
Input hidden	Represents a hidden field in an HTML form
Input password	Represents a password field in an HTML form
Input radio	Represents a radio button in an HTML form
Input reset	Represents a reset button in an HTML form
Input submit	Represents a submit button in an HTML form
Input text	Represents a text-input field in an HTML form
Link	Represents a <link> element

Meta	Represents a <meta> element
Option	Represents an <option> element
Select	Represents a selection list in an HTML form
Style	Represents an individual style statement
Table	Represents a <table> element
TableData	Represents a <td> element
TableRow	Represents a <tr> element
Textarea	Represents a <textarea> element

JavaScript HTML DOM Objects

In addition to the built-in JavaScript objects, you can also access and manipulate all of the HTML DOM objects with JavaScript.

More JavaScript Objects

Follow the links to learn more about the objects and their collections, properties, methods and events.

Object	Description
Window	The top level object in the JavaScript hierarchy. The Window object represents a browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag
Navigator	Contains information about the client's browser
Screen	Contains information about the client's display screen
History	Contains the visited URLs in the browser window
Location	Contains information about the current URL

The HTML DOM

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.

Follow the links below to learn more about how to access and manipulate each DOM object with JavaScript:

Object	**Description**
Document	Represents the entire HTML document and can be used to access all elements in a page
Anchor	Represents an <a> element
Area	Represents an <area> element inside an image-map
Base	Represents a <base> element
Body	Represents the <body> element
Button	Represents a <button> element
Event	Represents the state of an event
Form	Represents a <form> element
Frame	Represents a <frame> element
Frameset	Represents a <frameset> element
Iframe	Represents an <iframe> element
Image	Represents an <img> element
Input button	Represents a button in an HTML form
Input checkbox	Represents a checkbox in an HTML form
Input file	Represents a fileupload in an HTML form
Input hidden	Represents a hidden field in an HTML form
Input password	Represents a password field in an HTML form
Input radio	Represents a radio button in an HTML form
Input reset	Represents a reset button in an HTML form
Input submit	Represents a submit button in an HTML form
Input text	Represents a text-input field in an HTML form
Link	Represents a <link> element
Meta	Represents a <meta> element
Option	Represents an <option> element

Select	Represents a selection list in an HTML form
Style	Represents an individual style statement
Table	Represents a <table> element
TableData	Represents a <td> element
TableRow	Represents a <tr> element
Textarea	Represents a <textarea> element

```
<html>
<body>
<script type="text/javascript">
var browser=navigator.appName;
var b_version=navigator.appVersion;
var version=parseFloat(b_version);
document.write("Browser name: "+ browser);
document.write("<br />");
document.write("Browser version: "+ version);
</script>
</body>
</html>
<html>
<body>
<script type="text/javascript">
document.write("<p>Browser: ");
document.write(navigator.appName + "</p>");

document.write("<p>Browserversion: ");
document.write(navigator.appVersion + "</p>");
```

```
document.write("<p>Code: ");
document.write(navigator.appCodeName + "</p>");

document.write("<p>Platform: ");
document.write(navigator.platform + "</p>");

document.write("<p>Cookies enabled: ");
document.write(navigator.cookieEnabled + "</p>");

document.write("<p>Browser's user agent header: ");
document.write(navigator.userAgent + "</p>");
</script>
</body>
</html>

<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>

<html>
<head>
<script type="text/javascript">
function detectBrowser()
```

```
{
var browser=navigator.appName;
var b_version=navigator.appVersion;
var version=parseFloat(b_version);
if ((browser=="Netscape"||browser=="Microsoft Internet Explorer") &&
(version>=4))
  {
  alert("Your browser is good enough!");
  }
else
  {
  alert("It's time to upgrade your browser!");
  }
}
</script>
</head>

<body onload="detectBrowser()">
</body>

</html>
```

JavaScript Cookies

A cookie is often used to identify a user.

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

Create and Store a 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.

First, we create a function that stores the name of the visitor in a cookie variable:

```
function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+
```

```
((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}
```

The parameters of the function above hold the name of the cookie, the value of the cookie, and the number of days until the cookie expires.

In the function above we first convert the number of days to a valid date, then we add the number of days until the cookie should expire. After that we store the cookie name, cookie value and the expiration date in the document.cookie object.

Then, we create another function that checks if the cookie has been set:

```
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 "";
}
```

The function above first checks if a cookie is stored at all in the document.cookie object. If the document.cookie object holds some cookies, then check to see if our specific cookie is stored. If our cookie is found, then return the value, if not - return an empty string.

Last, we create the function that displays a welcome message if the cookie is set, and if the cookie is not set it will display a prompt box, asking for the name of the user:

```
function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
{
alert('Welcome again '+username+'!');
}
else
{
username=prompt('Please enter your name:',"");
if (username!=null && username!="")
{
setCookie('username',username,365);
}
}
}
```

All together now:

```
<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 "";
}
function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}
function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
  {
  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>
```

The example above runs the checkCookie() function when the page loads.

```
<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 ""
}


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

```
function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
  {
  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.

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

The function below checks if a required field has been left empty. If the required field is blank, an alert box alerts a message and the function returns false. If a value is entered, the function returns true (means that data is OK):

```
function validate_required(field,alerttxt)
{
with (field)
{
  if (value==null||value=="")
  {
  alert(alerttxt);return false;
  }
  else
  {
  return true;
  }
}
}
```

The entire script, with the HTML form could look something like this:

```
<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>
```

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 at least an @ sign and a dot (.). Also, the @ must not be the first character of the email address, and the last dot must at least be one character after the @ sign:

```
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;}
}
}
```

The entire script, with the HTML form could look something like this:

```
<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

With JavaScript we can create animated images.

Examples

Button animation

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.

In the following example we will add an image that should act as a link button on a web page. We will then add an onMouseOver event and an onMouseOut event that will run two JavaScript functions that will change between the images.

The HTML Code

The HTML code looks like this:

```
<a href="http://www.w3schools.com" target="_blank">
<img border="0" alt="Visit W3Schools!"
src="b_pink.gif" name="b1"
onmouseOver="mouseOver()"
onmouseOut="mouseOut()" />
</a>
```

Note that we have given the image a name to make it possible for JavaScript to address it later.

The onMouseOver event tells the browser that once a mouse is rolled over the image, the browser should execute a function that will replace the image with another image.

The onMouseOut event tells the browser that once a mouse is rolled away from the image, another JavaScript function should be executed. This function will insert the original image again.

The JavaScript Code

The changing between the images is done with the following JavaScript:

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

```
}
</script>
```

The function mouseOver() causes the image to shift to "b_blue.gif".

The function mouseOut() causes the image to shift to "b_pink.gif".

The Entire Code

```
<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">
<img border="0" alt="Visit W3Schools!"
src="b_pink.gif" name="b1"
onmouseOver="mouseOver()"
onmouseOut="mouseOut()" />
</a>
</body>
</html>
```

<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>

</body>

</html>

JavaScript Image Maps

An image-map is an image with clickable regions.

HTML Image Maps

From our HTML tutorial we have learned that an image-map is an image with clickable regions. Normally, each region has an associated hyperlink. Clicking on one of the regions takes you to the associated link.

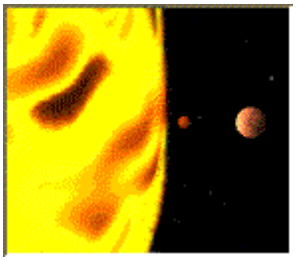
Example

The example below demonstrates how to create an HTML image map, with clickable regions. Each of the regions is a hyperlink:

```
<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"
  href ="sun.htm" target ="_blank"
  alt="Sun" />
<area shape ="circle" coords ="90,58,3"
  href ="mercur.htm" target ="_blank"
  alt="Mercury" />
<area shape ="circle" coords ="124,58,8"
  href ="venus.htm" target ="_blank"
  alt="Venus" />
</map>
```

Result



Adding some JavaScript

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.

Here's the above example, with some JavaScript added:

```
<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>
```

<html>

<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"

href ="sun.htm" target ="_blank" alt="Sun" />

<area shape ="circle" coords ="90,58,3"

href ="mercur.htm" target ="_blank" alt="Mercury" />

<area shape ="circle" coords ="124,58,8"

href ="venus.htm" target ="_blank" alt="Venus" />

```
</map>


</body>

</html>


<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>

</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.

Examples

Simple timing

Another simple timing

Timing event in an infinite loop

Timing event in an infinite loop - with a Stop button

A clock created with a timing event

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.

It's very easy to time events in JavaScript. The two key methods that are used are:

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

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

setTimeout()

Syntax

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

The setTimeout() method returns a value - In the statement above, the value is stored in a variable called t. If you want to cancel this setTimeout(), you can refer to it using the variable name.

The first parameter of setTimeout() is a string that contains a JavaScript statement. This statement could be a statement like "alert('5 seconds!')" or a call to a function, like "alertMsg()".

The second parameter indicates how many milliseconds from now you want to execute the first parameter.

Note: There are 1000 milliseconds in one second.

Example

When the button is clicked in the example below, an alert box will be displayed after 5 seconds.

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 seconds!')",5000);
}
</script>
</head>
<body>
<form>
<input type="button" value="Display timed alertbox!"
onClick="timedMsg()">
</form>
</body>
</html>
```

Example - Infinite Loop

To get a timer to work in an infinite loop, we must write a function that calls itself. In the example below, when the button is clicked, the input field will start to count (for ever), starting at 0:

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!"
onClick="timedCount()">
<input type="text" id="txt">
</form>
</body>
</html>
```

clearTimeout()

Syntax

```
clearTimeout(setTimeout_variable)
```

Example

The example below is the same as the "Infinite Loop" example above. The only difference is that we have now added a "Stop Count!" button that stops the timer:

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}
function stopCount()
{
clearTimeout(t);
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!"
onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="Stop count!"
onClick="stopCount()">
</form>
</body>
</html>
```

<html>

<head>

<script type="text/javascript">

function timedMsg()

{

var t=setTimeout("alert('5 seconds!')",5000);

```
}
</script>
</head>

<body>
<form>
<input type="button" value="Display timed alertbox!" onClick =
"timedMsg()">
</form>
<p>Click on the button above. An alert box will be displayed after 5
seconds.</p>
</body>

</html>

<html>
<head>
<script type="text/javascript">
function timedText()
{
var t1=setTimeout("document.getElementById('txt').value='2
seconds!'",2000);
var t2=setTimeout("document.getElementById('txt').value='4
seconds!'",4000);
var t3=setTimeout("document.getElementById('txt').value='6
seconds!'",6000);
}
</script>
```

```
</head>


<body>

<form>

<input type="button" value="Display timed text!" onClick="timedText()">

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

</form>

<p>Click on the button above. The input field will tell you when two, four, and six seconds have passed.</p>

</body>


</html>


<html>

<head>

<script type="text/javascript">

var c=0;

var t;

function timedCount()

{

document.getElementById('txt').value=c;

c=c+1;

t=setTimeout("timedCount()",1000);

}

</script>

</head>
```

```
<body>

<form>

<input type="button" value="Start count!" onClick="timedCount()">

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

</form>

<p>Click on the button above. The input field will count for ever, starting at 0.</p>

</body>


</html>




<html>

<head>

<script type="text/javascript">

var c=0;

var t;

function timedCount()

{

document.getElementById('txt').value=c;

c=c+1;

t=setTimeout("timedCount()",1000);

}


function stopCount()

{

clearTimeout(t);
```

```
}
</script>
</head>

<body>
<form>
<input type="button" value="Start count!" onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="Stop count!" onClick="stopCount()">
</form>
<p>
Click on the "Start count!" button above to start the timer. The input field will count forever, starting at 0. Click on the "Stop count!" button to stop the counting.
</p>
</body>

</html>

<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
```

```
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
}

function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

JavaScript Create Your Own Objects

Objects are useful to organize information.

Examples

Create a direct instance of an object

Create a template for an object

JavaScript Objects

Earlier in this tutorial we have seen that JavaScript has several built-in objects, like String, Date, Array, and more. In addition to these built-in objects, you can also create your own.

An object is just a special kind of data, with a collection of properties and methods.

Let's illustrate with an example: A person is an object. Properties are the values associated with the object. The persons' properties include name, height, weight, age, skin tone, eye color, etc. All persons have these properties, but the values of those properties will differ from person to person. Objects also have methods. Methods are the actions that can be performed on objects. The persons' methods could be eat(), sleep(), work(), play(), etc.

Properties

The syntax for accessing a property of an object is:

```
objName.propName
```

You can add properties to an object by simply giving it a value. Assume that the personObj already exists - you can give it properties named firstname, lastname, age, and eyecolor as follows:

```
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=30;
personObj.eyecolor="blue";
document.write(personObj.firstname);
```

The code above will generate the following output:

```
John
```

Methods

An object can also contain methods.

You can call a method with the following syntax:

```
objName.methodName()
```

Note: Parameters required for the method can be passed between the parentheses.

To call a method called sleep() for the personObj:

```
personObj.sleep();
```

Creating Your Own Objects

There are different ways to create a new object:

1. Create a direct instance of an object

The following code creates an instance of an object and adds four properties to it:

```
personObj=new Object();
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=50;
personObj.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 a template of an object

The template defines the structure of an object:

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

```
this.lastname=lastname;
this.age=age;
this.eyecolor=eyecolor;
}
```

Notice that the template is just a function. Inside the function you need to assign things to this.propertyName. The reason for all the "this" stuff is that you're going to have more than one person at a time (which person you're dealing with must be clear). That's what "this" is: the instance of the object at hand.

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

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

You can also add some methods to the person object. This is also done inside the template:

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

Note that methods are just functions attached to objects. Then we will have to write the newlastname() function:

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

The newlastname() function defines the person's new last name and assigns that to the person. JavaScript knows which person you're talking about by using "this.". So, now you can write: myMother.newlastname("Doe").

<html>

<body>

<script type="text/javascript">

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

document.write(personObj.firstname + " is " + personObj.age + " years old.");
</script>

</body>
</html>
<html>
<body>

<script type="text/javascript">
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname;
this.lastname=lastname;
this.age=age;
this.eyecolor=eyecolor;
}

myFather=new person("John","Doe",50,"blue");
```

```
document.write(myFather.firstname + " is " + myFather.age + " years
old.");

</script>


</body>

</html>
```

Examples

```
<html>

<body>


<script type="text/javascript">

document.write("Hello World!");

</script>


</body>

</html>


<html>

<body>


<script type="text/javascript">

document.write("<h1>This is a header</h1>");

</script>


</body>

</html>

<html>
```

```
<head>
</head>

<body>

<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>

</body>
</html>

<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>

<body onload="message()">

</body>
</html>
```

```
<html>
<head>
</head>
<body>

<script src="xxx.js">
</script>

<p>
The actual script is in an external script file called "xxx.js".
</p>

</body>
</html>
<html>
<body>

<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>

</body>
```

```
</html>

<html>
<body>

<script type="text/javascript">
{
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
}
</script>

</body>
</html>
<html>
<body>

<script type="text/javascript">
// This will write a header:
document.write("<h1>This is a header</h1>");
// This will write two paragraphs:
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

```
</body>
</html>

<html>
<body>

<script type="text/javascript">
/*
The code below will write
one header and two paragraphs
*/
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
//document.write("<p>This is another paragraph</p>");
```

```
</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">
/*
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
*/
</script>

</body>
</html>

<html>
<body>

<script type="text/javascript">
var firstname;
firstname="Hege";
document.write(firstname);
```

```
document.write("<br />");

firstname="Tove";

document.write(firstname);

</script>


<p>The script above declares a variable,

assigns a value to it, displays the value, change the value,

and displays the value again.</p>


</body>

</html>
```

HTML DOM - Events

Events are actions that can be detected by JavaScript.

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 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.

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 box 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!

For a complete reference of the events supported by the HTML DOM, see our HTML DOM Event reference.

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()">
```

onMouseOver and onMouseOut

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

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return false">
```

```
<img src="w3schools.gif" width="100" height="30">
</a>
```

HTML 4.0 Event Attributes

New to HTML 4.0 is 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 attributes that can be inserted into HTML tags to define event actions.

If you want to learn more about programming with these events, you should study our JavaScript tutorial and our DHTML tutorial.

Window Events

Only valid in body and frameset elements.

Attribute	Value	Description
onload	*script*	Script to be run when a document loads
onunload	*script*	Script to be run when a document unloads

Form Element Events

Only valid in form elements.

Attribute	Value	Description
onchange	*script*	Script to be run when the element changes
onsubmit	*script*	Script to be run when the form is submitted
onreset	*script*	Script to be run when the form is reset
onselect	*script*	Script to be run when the element is selected
onblur	*script*	Script to be run when the element loses focus
onfocus	*script*	Script to be run when the element gets focus

Keyboard Events

Not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title elements.

Attribute	Value	Description
onkeydown	*script*	What to do when key is pressed
onkeypress	*script*	What to do when key is pressed and released
onkeyup	*script*	What to do when key is released

Mouse Events

Not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, title elements.

Attribute	Value	Description
onclick	*script*	What to do on a mouse click
ondblclick	*script*	What to do on a mouse double-click
onmousedown	*script*	What to do when mouse button is pressed
onmousemove	*script*	What to do when mouse pointer moves
onmouseout	*script*	What to do when mouse pointer moves out of an element
onmouseover	*script*	What to do when mouse pointer moves over an element
onmouseup	*script*	What to do when mouse button is released

HTML DOM Window Object

Window Object

The Window object is the top level object in the JavaScript hierarchy.

The Window object represents a browser window.

A Window object is created automatically with every instance of a <body> or <frameset> tag.

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Window Object Collections

Collection	Description	IE	F	O
frames[]	Returns all named frames in the window	4	1	9

Window Object Properties

Property	Description	IE	F	O
closed	Returns whether or not a window has been closed	4	1	9
defaultStatus	Sets or returns the default text in the statusbar of the window	4	No	9
document	See Document object	4	1	9
history	See History object	4	1	9
length	Sets or returns the number of frames in the window	4	1	9
location	See Location object	4	1	9
name	Sets or returns the name of the window	4	1	9
opener	Returns a reference to the window that created the window	4	1	9
outerHeight	Sets or returns the outer height of a window	No	1	No
outerWidth	Sets or returns the outer width of a window	No	1	No
pageXOffset	Sets or returns the X position of the current page in relation to the upper left corner of a window's display area	No	No	No
pageYOffset	Sets or returns the Y position of the current page in relation to the upper left corner of a window's display area	No	No	No
parent	Returns the parent window	4	1	9
personalbar	Sets whether or not the browser's personal bar (or directories bar) should be visible			
scrollbars	Sets whether or not the scrollbars should be visible			

self	Returns a reference to the current window	4	1	9
status	Sets the text in the statusbar of a window	4	No	9
statusbar	Sets whether or not the browser's statusbar should be visible			
toolbar	Sets whether or not the browser's tool bar is visible or not (can only be set before the window is opened and you must have UniversalBrowserWrite privilege)			
top	Returns the topmost ancestor window	4	1	9

Window Object Methods

Method	Description	IE	F	O
alert()	Displays an alert box with a message and an OK button	4	1	9
blur()	Removes focus from the current window	4	1	9
clearInterval()	Cancels a timeout set with setInterval()	4	1	9
clearTimeout()	Cancels a timeout set with setTimeout()	4	1	9
close()	Closes the current window	4	1	9
confirm()	Displays a dialog box with a message and an OK and a Cancel button	4	1	9
createPopup()	Creates a pop-up window	4	No	No
focus()	Sets focus to the current window	4	1	9
moveBy()	Moves a window relative to its current position	4	1	9
moveTo()	Moves a window to the specified position	4	1	9
open()	Opens a new browser window	4	1	9
print()	Prints the contents of the current window	5	1	9
prompt()	Displays a dialog box that prompts the user for input	4	1	9
resizeBy()	Resizes a window by the specified pixels	4	1	9

resizeTo()	Resizes a window to the specified width and height	4	1.5	9
scrollBy()	Scrolls the content by the specified number of pixels	4	1	9
scrollTo()	Scrolls the content to the specified coordinates	4	1	9
setInterval()	Evaluates an expression at specified intervals	4	1	9
setTimeout()	Evaluates an expression after a specified number of milliseconds			

HTML DOM Document Object

Document Object

The Document object represents the entire HTML document and can be used to access all elements in a page.

The Document object is part of the Window object and is accessed through the window.document property.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Document Object Collections

Collection	Description	IE	F	O	W3C
anchors[]	Returns a reference to all Anchor objects in the document	4	1	9	Yes
forms[]	Returns a reference to all Form objects in the document	4	1	9	Yes
images[]	Returns a reference to all Image objects in the document	4	1	9	Yes
links[]	Returns a reference to all Area and Link objects in the document	4	1	9	Yes

Document Object Properties

Property	Description	IE	F	O	W3C
body	Gives direct access to the <body> element				
cookie	Sets or returns all cookies associated with the current document	4	1	9	Yes
domain	Returns the domain name for the current document	4	1	9	Yes
lastModified	Returns the date and time a document was last modified	4	1	No	No
referrer	Returns the URL of the document that loaded the current document	4	1	9	Yes
title	Returns the title of the current document	4	1	9	Yes
URL	Returns the URL of the current document	4	1	9	Yes

Document Object Methods

Method	Description	IE	F	O	W3C
close()	Closes an output stream opened with the document.open() method, and displays the collected data	4	1	9	Yes
getElementById()	Returns a reference to the first object with the specified id	5	1	9	Yes
getElementsByName()	Returns a collection of objects with the specified name	5	1	9	Yes
getElementsByTagName()	Returns a collection of objects with the specified tagname	5	1	9	Yes
open()	Opens a stream to collect the output from any document.write() or document.writeln() methods	4	1	9	Yes
write()	Writes HTML expressions or JavaScript code to a document	4	1	9	Yes

writeln()	Identical to the write() method, with the addition of writing a new line character after each expression	4	1	9	Yes

Your browser does not support inline frames or is currently configured not to display inline frames.

HTML DOM Style Object

Style object

The Style object represents an individual style statement. The Style object can be accessed from the document or from the elements to which that style is applied.

Syntax for using the Style object properties:

```
document.getElementById("id").style.property="value"
```

The Style object property categories:

- Background
- Border and Margin
- Layout
- List
- Misc
- Positioning
- Printing
- Scrollbar
- Table
- Text
- Standard

IE: Internet Explorer, **M:** Mac IE only, **W:** Windows IE only, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Background properties

Property	Description	IE	F	O	W3C
background	Sets all background properties in one	4	1	9	Yes
backgroundAttachment	Sets whether a background-image is fixed	4	1	9	Yes

	or scrolls with the page				
backgroundColor	Sets the background-color of an element	4	1	9	Yes
backgroundImage	Sets the background-image of an element	4	1	9	Yes
backgroundPosition	Sets the starting position of a background-image	4	No	No	Yes
backgroundPositionX	Sets the x-coordinates of the backgroundPosition property	4	No	No	No
backgroundPositionY	Sets the y-coordinates of the backgroundPosition property	4	No	No	No
backgroundRepeat	Sets if/how a background-image will be repeated	4	1	9	Yes

Border and Margin properties

Property	Description	IE	F	O	W3C
border	Sets all properties for the four borders in one	4	1	9	Yes
borderBottom	Sets all properties for the bottom border in one	4	1	9	Yes
borderBottomColor	Sets the color of the bottom border	4	1	9	Yes
borderBottomStyle	Sets the style of the bottom border	4	1	9	Yes
borderBottomWidth	Sets the width of the bottom border	4	1	9	Yes
borderColor	Sets the color of all four borders (can have up to four colors)	4	1	9	Yes
borderLeft	Sets all properties for the left border in one	4	1	9	Yes
borderLeftColor	Sets the color of the left border	4	1	9	Yes
borderLeftStyle	Sets the style of the left border	4	1	9	Yes
borderLeftWidth	Sets the width of the left border	4	1	9	Yes

borderRight	Sets all properties for the right border in one	4	1	9	Yes
borderRightColor	Sets the color of the right border	4	1	9	Yes
borderRightStyle	Sets the style of the right border	4	1	9	Yes
borderRightWidth	Sets the width of the right border	4	1	9	Yes
borderStyle	Sets the style of all four borders (can have up to four styles)	4	1	9	Yes
borderTop	Sets all properties for the top border in one	4	1	9	Yes
borderTopColor	Sets the color of the top border	4	1	9	Yes
borderTopStyle	Sets the style of the top border	4	1	9	Yes
borderTopWidth	Sets the width of the top border	4	1	9	Yes
borderWidth	Sets the width of all four borders (can have up to four widths)	4	1	9	Yes
margin	Sets the margins of an element (can have up to four values)	4	1	9	Yes
marginBottom	Sets the bottom margin of an element	4	1	9	Yes
marginLeft	Sets the left margin of an element	4	1	9	Yes
marginRight	Sets the right margin of an element	4	1	9	Yes
marginTop	Sets the top margin of an element	4	1	9	Yes
outline	Sets all outline properties in one	5M	1	9	Yes
outlineColor	Sets the color of the outline around an element	5M	1	9	Yes
outlineStyle	Sets the style of the outline around an element	5M	1	9	Yes
outlineWidth	Sets the width of the outline around an element	5M	1	9	Yes

padding	Sets the padding of an element (can have up to four values)	4	1	9	Yes
paddingBottom	Sets the bottom padding of an element	4	1	9	Yes
paddingLeft	Sets the left padding of an element	4	1	9	Yes
paddingRight	Sets the right padding of an element	4	1	9	Yes
paddingTop	Sets the top padding of an element	4	1	9	Yes

Layout properties

Property	Description	IE	F	O	W3C
clear	Sets on which sides of an element other floating elements are not allowed	4	1	9	Yes
clip	Sets the shape of an element	4	1	9	Yes
content	Sets meta-information	5M	1		Yes
counterIncrement	Sets a list of counter names, followed by an integer. The integer indicates by how much the counter is incremented for every occurrence of the element. The default is 1	5M	1		Yes
counterReset	Sets a list of counter names, followed by an integer. The integer gives the value that the counter is set to on each occurrence of the element. The default is 0	5M	1		Yes
cssFloat	Sets where an image or a text will appear (float) in another element	5M	1	9	Yes
cursor	Sets the type of cursor to be displayed	4	1	9	Yes
direction	Sets the text direction of an element	5	1	9	Yes
display	Sets how an element will be displayed	4	1	9	Yes
height	Sets the height of an element	4	1	9	Yes
markerOffset	Sets the distance between the nearest	5M	1		Yes

	border edges of a marker box and its principal box				
marks	Sets whether cross marks or crop marks should be rendered just outside the page box edge	5M	1		Yes
maxHeight	Sets the maximum height of an element	5M	1	9	Yes
maxWidth	Sets the maximum width of an element	5M	1	9	Yes
minHeight	Sets the minimum height of an element	5M	1	9	Yes
minWidth	Sets the minimum width of an element	5M	1	9	Yes
overflow	Specifies what to do with content that does not fit in an element box	4	1	9	Yes
verticalAlign	Sets the vertical alignment of content in an element	4	1	No	Yes
visibility	Sets whether or not an element should be visible	4	1	9	Yes
width	Sets the width of an element	4	1	9	Yes

List properties

Property	Description	IE	F	O	W3C
listStyle	Sets all the properties for a list in one	4	1	9	Yes
listStyleImage	Sets an image as the list-item marker	4	1	No	Yes
listStylePosition	Positions the list-item marker	4	1	9	Yes
listStyleType	Sets the list-item marker type	4	1	9	Yes

Misc properties

Property	Description	IE	F	O	W3C
cssText		4	1		

Positioning properties

Property	Description	IE	F	O	W3C

bottom	Sets how far the bottom edge of an element is above/below the bottom edge of the parent element	5	1	9	Yes
left	Sets how far the left edge of an element is to the right/left of the left edge of the parent element	4	1	9	Yes
position	Places an element in a static, relative, absolute or fixed position	4	1	9	Yes
right	Sets how far the right edge of an element is to the left/right of the right edge of the parent element	5	1	9	Yes
top	Sets how far the top edge of an element is above/below the top edge of the parent element	4	1	9	Yes
zIndex	Sets the stack order of an element	4	1	9	Yes

Printing properties

Property	Description	IE	F	O	W3C
orphans	Sets the minimum number of lines for a paragraph that must be left at the bottom of a page	5M	1	9	Yes
page	Sets a page type to use when displaying an element	5M	1	9	Yes
pageBreakAfter	Sets the page-breaking behavior after an element	4	1	9	Yes
pageBreakBefore	Sets the page-breaking behavior before an element	4	1	9	Yes
pageBreakInside	Sets the page-breaking behavior inside an element	5M	1	9	Yes
size	Sets the orientation and size of a page		1	9	Yes
widows	Sets the minimum number of lines for a	5M	1	9	Yes

	paragraph that must be left at the top of a page				

Scrollbar properties (IE-only)

Property	Description	IE	F	O	W3C
scrollbar3dLightColor	Sets the color of the left and top sides of the arrows and scroll boxes	5W	No	No	No
scrollbarArrowColor	Sets the color of the arrows on a scroll bar	5W	No	No	No
scrollbarBaseColor	Sets the base color of the scroll bar	5W	No	No	No
scrollbarDarkShadowColor	Sets the color of the right and bottom sides of the arrows and scroll boxes	5W	No	No	No
scrollbarFaceColor	Sets the front color of the scroll bar	5W	No	No	No
scrollbarHighlightColor	Sets the color of the left and top sides of the arrows and scroll boxes, and the background of a scroll bar	5W	No	No	No
scrollbarShadowColor	Sets the color of the right and bottom sides of the arrows and scroll boxes	5W	No	No	No
scrollbarTrackColor	Sets the background color of a scroll bar	5W	No	No	No

Table properties

Property	Description	IE	F	O	W3C
borderCollapse	Sets whether the table border are collapsed into a single border or detached as in standard HTML	5	1	9	Yes
borderSpacing	Sets the distance that separates cell borders	5M	1	9	Yes
captionSide	Sets the position of the table caption	5M	No	No	Yes
emptyCells	Sets whether or not to show empty cells in a table	5M	1	9	Yes
tableLayout	Sets the algorithm used to display the	5	No	No	Yes

	table cells, rows, and columns				

Text properties

Property	Description	IE	F	O	W3C
color	Sets the color of the text	4	1	9	Yes
font	Sets all font properties in one	4	1	9	Yes
fontFamily	Sets the font of an element	4	1	9	Yes
fontSize	Sets the font-size of an element	4	1	9	Yes
fontSizeAdjust	Sets/adjusts the size of a text	5M	1	No	Yes
fontStretch	Sets how to condense or stretch a font	5M	No	No	Yes
fontStyle	Sets the font-style of an element	4	1	9	Yes
fontVariant	Displays text in a small-caps font	4	1	9	Yes
fontWeight	Sets the boldness of the font	4	1	9	Yes
letterSpacing	Sets the space between characters	4	1	9	Yes
lineHeight	Sets the distance between lines	4	1	9	Yes
quotes	Sets which quotation marks to use in a text	5M	1		Yes
textAlign	Aligns the text	4	1	9	Yes
textDecoration	Sets the decoration of a text	4	1	9	Yes
textIndent	Indents the first line of text	4	1	9	Yes
textShadow	Sets the shadow effect of a text	5M	1		Yes
textTransform	Sets capitalization effect on a text	4	1	9	Yes
unicodeBidi		5	1		Yes
whiteSpace	Sets how to handle line-breaks and white-space in a text	4	1	9	Yes
wordSpacing	Sets the space between words in a text	6	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

HTML DOM Hidden Object

Hidden Object

The Hidden object represents a hidden input field in an HTML form.

For each instance of an <input type="hidden"> tag in an HTML form, a Hidden object is created.

You can access a hidden input field by searching through the elements[] array of the form, or by using document.getElementById().

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Hidden Object Properties

Property	Description	IE	F	O	W3C
alt	Sets or returns an alternate text to display if a browser does not support hidden fields	5	1	9	Yes
form	Returns a reference to the form that contains the hidden field	4	1	9	Yes
id	Sets or returns the id of a hidden field	4	1	9	Yes
name	Sets or returns the name of a hidden field	4	1	9	Yes

type	Returns the type of form element a hidden input field is	4	1	9	Yes
value	Sets or returns the value of the value attribute of the hidden field	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes
title	Sets or returns an element's advisory title	5	1	9	Yes

Your browser does not support inline frames or is currently configured not to display inline frames.

HTML DOM TableCell Object

TableCell Object

The TableCell object represents an HTML table cell.

For each instance of a <td> tag in an HTML document, a TableCell object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

TableCell Object Properties

Property	Description	IE	F	O	W3C
abbr	Sets or returns an abbreviated version of the content in a table cell	6	1	9	Yes
align	Sets or returns the horizontal alignment of data	4	1	9	Yes

	within a table cell				
axis	Sets or returns a comma-delimited list of related table cells	6	1	9	Yes
cellIndex	Returns the position of a cell in the cells collection of a row	4	1	9	Yes
ch	Sets or returns the alignment character for a table cell				Yes
chOff	Sets or returns the offset of alignment character for a table cell				Yes
colSpan	Sets or returns the number of columns a table cell should span	4	1	9	Yes
headers	Sets or returns a list of space-separated header-cell ids				Yes
id	Sets or returns the id of a table cell	4	1	9	Yes
innerHTML	Sets or returns the HTML between the start and end tags of a table cell	4	1	9	No
rowSpan	Sets or returns the number of rows a table cell should span	4	1	9	Yes
scope	Sets or returns if this cell provides header information				Yes
vAlign	Sets or returns the vertical alignment of data within a table cell	4	1	9	Yes
width	Sets or returns the width of a table cell	4	1	9	Yes

Standard Properties

Property	Description	IE	F	O	W3C
className	Sets or returns the class attribute of an element	5	1	9	Yes
dir	Sets or returns the direction of text	5	1	9	Yes
lang	Sets or returns the language code for an element	5	1	9	Yes

title	Sets or returns an element's advisory title	5	1	9	Yes

Your browser does not support inline frames or is currently configured not to display inline frames.

HTML DOM TableRow Object

TableRow Object

The TableRow object represents an HTML table row.

For each instance of a <tr> tag in an HTML document, a TableRow object is created.

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

TableRow Object Collections

Collection	Description	IE	F	O	W3C
cells[]	Returns an array containing each cell in the table row	4	1	9	Yes

TableRow Object Properties

Property	Description	IE	F	O	W3C
align	Sets or returns the horizontal alignment of data within a table row	4	1	9	Yes
ch	Sets or returns the alignment character for cells in a table row				Yes
chOff	Sets or returns the offset of alignment character for the cells in a table row				Yes
id	Sets or returns the id of a table row	4	1	9	Yes
innerHTML	Sets or returns the HTML between the start and end tags of a table row	5	1	9	No

rowIndex	Returns the position of a row in the table's rows collection	4	1	9	Yes
sectionRowIndex	Returns the position of a row in the tBody, tHead, or tFoot rows collection				Yes
vAlign	Sets or returns the vertically alignment of data within a table row	4	1	9	Yes

TableRow Object Methods

Method	Description	IE	F	O	W3C
deleteCell()	Deletes a cell in a table row	4	1	9	Yes
insertCell()	Inserts a cell in a table row	4	1	9	Yes

Your browser does not support inline frames or is currently configured not to display inline frames.

HTML DOM Event Object

The event object gives you information about an event that has occurred.

Examples

Which mouse button was clicked?

What are the coordinates of the cursor?

What is the unicode of the key pressed?

What are the coordinates of the cursor, relative to the screen?

What are the coordinates of the cursor?

Was the shift key pressed?

Which element was clicked?

Which eventype occured?

Event Object

The Event object represents the state of an event, such as the element in which the event occurred, the state of the keyboard keys, the location of the mouse, and the state of the mouse buttons.

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

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** World Wide Web Consortium (Internet Standard).

Event Handlers

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...	IE	F	O	W3C
onabort	Loading of an image is interrupted	4	1	9	Yes
onblur	An element loses focus	3	1	9	Yes
onchange	The content of a field changes	3	1	9	Yes
onclick	Mouse clicks an object	3	1	9	Yes
ondblclick	Mouse double-clicks an object	4	1	9	Yes
onerror	An error occurs when loading a document or an image	4	1	9	Yes
onfocus	An element gets focus	3	1	9	Yes
onkeydown	A keyboard key is pressed	3	1	No	Yes

onkeypress	A keyboard key is pressed or held down	3	1	9	Yes
onkeyup	A keyboard key is released	3	1	9	Yes
onload	A page or an image is finished loading	3	1	9	Yes
onmousedown	A mouse button is pressed	4	1	9	Yes
onmousemove	The mouse is moved	3	1	9	Yes
onmouseout	The mouse is moved off an element	4	1	9	Yes
onmouseover	The mouse is moved over an element	3	1	9	Yes
onmouseup	A mouse button is released	4	1	9	Yes
onreset	The reset button is clicked	4	1	9	Yes
onresize	A window or frame is resized	4	1	9	Yes
onselect	Text is selected	3	1	9	Yes
onsubmit	The submit button is clicked	3	1	9	Yes
onunload	The user exits the page	3	1	9	Yes

Mouse / Keyboard Attributes

Property	Description	IE	F	O	W3C
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	6	1	9	Yes
button	Returns which mouse button was clicked when an event was triggered	6	1	9	Yes
clientX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
clientY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
ctrlKey	Returns whether or not the "CTRL" key	6	1	9	Yes

	was pressed when an event was triggered				
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered	6	1	9	Yes
relatedTarget	Returns the element related to the element that triggered the event	No	1	9	Yes
screenX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
screenY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	6	1	9	Yes

Other Event Attributes

Property	Description	IE	F	O	W3C
bubbles	Returns a Boolean value that indicates whether or not an event is a bubbling event	No	1	9	Yes
cancelable	Returns a Boolean value that indicates whether or not an event can have its default action prevented	No	1	9	Yes
currentTarget	Returns the element whose event listeners triggered the event	No	1	9	Yes
eventPhase	Returns which phase of the event flow is currently being evaluated				Yes
target	Returns the element that triggered the event	No	1	9	Yes
timeStamp	Returns the time stamp, in milliseconds, from the epoch (system start or event trigger)	No	1	9	Yes
type	Returns the name of the event	6	1	9	Ye

```
<html>
<head>
<script type="text/javascript">
function whichButton(event)
{
if (event.button==2)
  {
  alert("You clicked the right mouse button!");
  }
else
  {
  alert("You clicked the left mouse button!");
  }
}
</script>
</head>

<body onmousedown="whichButton(event)">
<p>Click in the document. An alert box will alert which mouse button you clicked.</p>
</body>

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

```
<script type="text/javascript">
function show_coords(event)
{
x=event.clientX;
y=event.clientY;
alert("X coords: " + x + ", Y coords: " + y);
}
</script>
</head>

<body onmousedown="show_coords(event)">

<p>Click in the document. An alert box will alert the x and y coordinates of the mouse pointer.</p>

</body>
</html>

<html>
<head>
<script type="text/javascript">
function whichButton(event)
{
alert(event.keyCode);
}

</script>
```

```
</head>


<body onkeyup="whichButton(event)">

<p><b>Note:</b> Make sure the right frame has focus when trying this example!</p>

<p>Press a key on your keyboard. An alert box will alert the unicode of the key pressed.</p>

</body>


</html>

<html>

<head>


<script type="text/javascript">

function coordinates(event)

{

x=event.screenX;

y=event.screenY;

alert("X=" + x + " Y=" + y);

}


</script>

</head>

<body onmousedown="coordinates(event)">


<p>
```

Click somewhere in the document. An alert box will alert the x and y coordinates of the cursor, relative to the screen.

** **

</body>

</html>

<html>

<head>

<script type="text/javascript">

function coordinates(event)

{

x=event.x;

y=event.y;

alert("X=" + x + " Y=" + y);

}

</script>

</head>

<body onmousedown="coordinates(event)">

** **

Click somewhere in the document. An alert box will alert the x and y coordinates of the cursor.

** **

```
</body>
</html>

<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
if (event.shiftKey==1)
  {
  alert("The shift key was pressed!");
  }
else
  {
  alert("The shift key was NOT pressed!");
  }
}
</script>
</head>

<body onmousedown="isKeyPressed(event)">

<p>Click somewhere in the document. An alert box will tell you if you pressed the shift key or not.</p>

</body>
</html>
```

```
<html>
<head>
<script type="text/javascript">
function whichElement(e)
{
var targ;
if (!e)
  {
  var e=window.event;
  }
if (e.target)
  {
  targ=e.target;
  }
else if (e.srcElement)
  {
  targ=e.srcElement;
  }
if (targ.nodeType==3) // defeat Safari bug
  {
  targ = targ.parentNode;
  }
var tname;
tname=targ.tagName;
alert("You clicked on a " + tname + " element.");
```

```
}
</script>
</head>

<body onmousedown="whichElement(event)">
<p>Click somewhere in the document. An alert box will alert the tag name of the element you clicked on.</p>

<h3>This is a header</h3>
<p>This is a paragraph</p>
<img border="0" src="ball16.gif" width="29" height="28" alt="Ball">
</body>

</html>

<html>
<head>
<script type="text/javascript">
function getEventType(event)
{
alert(event.type);
}
</script>
</head>

<body onmousedown="getEventType(event)">
```

** Click in the document.**

An alert box will tell what type of event

**that was triggered. **

</body>

</html>

onabort Event

Definition and Usage

The onabort event occurs when loading of an image is aborted.

Syntax

```
onabort="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

Supported by the following JavaScript objects:

image

Example 1

In this example an alert box will be displayed if the loading of the image is aborted:

```
<img src="image_w3default.gif"
onabort="alert('Error: Loading of the image was aborted')">
```

Example 2

In this example we will call a function if the loading of the image is aborted:

```
<html>
<head>
<script type="text/javascript">
function abortImage()
{
alert('Error: Loading of the image was aborted')
}
</script>
</head>
<body>
<img src="image_w3default.gif"
onabort="abortImage()">
</body>
</html>
```

onblur Event

Definition and Usage

The onblur event occurs when an object loses focus.

Syntax

```
onblur="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <button>, <caption>, <cite>, <dd>, <del>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <frame>, <frameset>, <h1> to <h6>, <hr>, <i>, <iframe>, <img>, <input>, <ins>, <kbd>, <label>, <legend>, <li>, <object>, <ol>, <p>, <pre>, <q>, <samp>, <select>, <small>, , , <sub>, <sup>,

<table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

button, checkbox, fileUpload, layer, frame, password, radio, reset, submit, text, textarea, window

Example

In this example we will execute some JavaScript code when a user leaves an input field:

```
<html>
<head>
<script type="text/javascript">
function upperCase()
{
var x=document.getElementById("fname").value
document.getElementById("fname").value=x.toUpperCase()
}
</script>
</head>
<body>
Enter your name:
<input type="text" id="fname" onblur="upperCase()">
</body>
</html>
```

The output of the code above will be:

Enter your name:

onchange Event

 Complete Event Object Reference

Definition and Usage

The onchange event occurs when the content of a field changes.

Syntax

```
onchange="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<input type="text">, <select>, <textarea>

Supported by the following JavaScript objects:

fileUpload, select, text, textarea

Example

In this example we will execute some JavaScript code when a user changes the content of an input field:

```
<html>
<head>
<script type="text/javascript">
function upperCase(x)
{
var y=document.getElementById(x).value
document.getElementById(x).value=y.toUpperCase()
}
</script>
</head>
<body>
Enter your name:
<input type="text" id="fname"
onchange="upperCase(this.id)">
</body>
</html>
```

The output of the code above will be:

Enter your name:

Try-It-Yourself Demos

onchange
How to use the onchange event to execute some JavaScript code when a user changes the content of an input field.

onclick Event

Definition and Usage

The onclick event occurs when an object gets clicked.

Syntax

```
onclick="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

button, document, checkbox, link, radio, reset, submit

Example

In this example the text in the first input field will be copied to the second input field when a button is clicked:

```
<html>
<body>
Field1: <input type="text" id="field1" value="Hello World!">
<br />
Field2: <input type="text" id="field2">
<br /><br />
Click the button below to copy the content of Field1 to Field2.
<br />
<button onclick="document.getElementById('field2').value=
```

```
document.getElementById('field1').value">Copy Text</button>
</body>
</html>
```

The output of the code above will be:

```
Field1:
Field2:

Click the button below to copy the content of Field1 to Field2.
Copy Text
```

ondblclick Event

Definition and Usage

The ondblclick event occurs when an object gets double-clicked.

Syntax

```
ondblclick="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

```
<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>,
<cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>, <em>, <fieldset>, <form>, <h1> to <h6>, <hr>,
<i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>,
<samp>, <select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>, <td>,
<textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

Supported by the following JavaScript objects:

```
document, link
```

Example

In this example the second field changes according to the first field when you double-click on the button:

```
<html>
<body>
Field1: <input type="text" id="field1" value="Hello World!">
<br />
Field2: <input type="text" id="field2">
<br /><br />
Click the button below to copy the content of Field1 to Field2.
<br />
<button ondblclick="document.getElementById('field2').value=
document.getElementById('field1').value">Copy Text</button>
</body>
</html>
```

The output of the code above will be:

Field1:
Field2:

Double-click the button below to copy the content of Field1 to Field2.
Copy Text

onerror Event

Definition and Usage

The onerror event is triggered when an error occurs loading a document or an image.

Syntax

```
onerror="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

```
<img>, <object>, <style>
```

Supported by the following JavaScript objects:

```
window, image
```

Example

In this example an alert box will be displayed if an error occurs when loading an image:

```
<img src="image.gif"
onerror="alert('The image could not be loaded.')">
```

onerror
How to use onerror.

Your browser does not support inline frames or is currently configured not to display inline frames.

onfocus Event

Definition and Usage

The onfocus event occurs when an object gets focus.

Syntax

```
onfocus="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <button>, <caption>, <cite>, <dd>, <del>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <frame>, <frameset>, <h1> to <h6>, <hr>, <i>, <iframe>, <img>, <input>, <ins>, <kbd>, <label>, <legend>, <li>, <object>, <ol>, <p>, <pre>, <q>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

button, checkbox, fileUpload, layer, frame, password, radio, reset, select, submit, text, textarea, window

Example

In this example the background color of the input fields change when they get focus:

```
<html>
<head>
<script type="text/javascript">
function setStyle(x)
{
document.getElementById(x).style.background="yellow"
}
</script>
</head>
<body>
First name: <input type="text"
onfocus="setStyle(this.id)" id="fname">
<br />
Last name: <input type="text"
onfocus="setStyle(this.id)" id="lname">
</body>
</html>
```

The output of the code above will be:

First name:
Last name:

onkeydown Event

Definition and Usage

The onkeydown event occurs when a keyboard key is pressed.

Syntax

```
onkeydown="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <del>, <dfn>, , <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>, <q>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

document, image, link, textarea

Tips and Notes

Browser differences: Internet Explorer uses event.keyCode to retrieve the character that was pressed and Netscape/Firefox/Opera uses event.which.

Example

In this example the user cannot type numbers into the input field:

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

```
function noNumbers(e)
{
var keynum
var keychar
var numcheck
if(window.event) // IE
{
keynum = e.keyCode
}
else if(e.which) // Netscape/Firefox/Opera
{
keynum = e.which
}
keychar = String.fromCharCode(keynum)
numcheck = /\d/
return !numcheck.test(keychar)
}
</script>
<form>
<input type="text" onkeydown="return noNumbers(event)" />
</form>
</html>
```

The output of the code above will be:

onkeypress Event

Definition and Usage

The onkeydown event occurs when a keyboard key is pressed or held down.

Syntax

```
onkeypress="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <del>, <dfn>, , <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <input>, <kbd>, <label>, <legend>,   , <map>, <object>,   , <p>, <pre>, <q>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>,   , <var>

Supported by the following JavaScript objects:

document, image, link, textarea

Tips and Note

Browser differences: Internet Explorer uses event.keyCode to retrieve the character that was pressed and Netscape/Firefox/Opera uses event.which.

Example

In this example the user cannot type numbers into the input field:

```
<html>
<body>
<script type="text/javascript">
function noNumbers(e)
{
var keynum
var keychar
var numcheck
if(window.event) // IE
{
keynum = e.keyCode
}
else if(e.which) // Netscape/Firefox/Opera
{
keynum = e.which
}
keychar = String.fromCharCode(keynum)
numcheck = /\d/
return !numcheck.test(keychar)
}
</script>
<form>
<input type="text" onkeypress="return noNumbers(event)" />
</form>
</html>
```

The output of the code above will be:

```

```

onKeyUp Event

Definition and Usage

The onkeyup event occurs when a keyboard key is released.

Syntax

```
onkeyup="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <del>, <dfn>, , <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>, <q>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

document, image, link, textarea

Example

When typing letters in the input field in the following example, the letters will change to uppercase (one by one):

```
<html>
```

```
<head>
<script type="text/javascript">
function upperCase(x)
{
var y=document.getElementById(x).value
document.getElementById(x).value=y.toUpperCase()
}
</script>
</head>
<body>
Enter your name: <input type="text"
id="fname" onkeyup="upperCase(this.id)">
</body>
</html>
```

The output of the code above will be:

Enter your name:

onload Event

Definition and Usage

The onload event occurs immediately after a page or an image is loaded.

Syntax

```
onload="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<body>, <frame>, <frameset>, <iframe>, <img>, <link>, <script>

Supported by the following JavaScript objects:

image, layer, window

Example

In this example the text "Page is loaded" will be displayed in the status bar:

```
<html>
<head>
<script type="text/javascript">
function load()
{
window.status="Page is loaded"
}
</script>
</head>
<body onload="load()">
</body>
</html>
```

onmousedown Event

Definition and Usage

The onmousedown event occurs when a mouse button is clicked.

Syntax

```
onmousedown="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>,

<select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

button, document, link

Example 1

In this example an alert box is displayed when clicking on the picture:

```
<img src="image_w3default.gif"
onmousedown="alert('You clicked the picture!')">
```

The output of the code above will be (click on the picture):

Example 2

In this example an alert box will alert the tag name of the element you clicked on:

```
<html>
<head>
<script type="text/javascript">
function whichElement(e)
{
var targ
if (!e) var e = window.event
if (e.target) targ = e.target
else if (e.srcElement) targ = e.srcElement
if (targ.nodeType == 3) // defeat Safari bug
targ = targ.parentNode
var tname
tname=targ.tagName
alert("You clicked on a " + tname + " element.")
}
</script>
</head>
<body onmousedown="whichElement(event)">
```

```
<h2>This is a header</h2>
<p>This is a paragraph</p>
<img border="0" src="ball16.gif" alt="Ball">
</body>
</html>
```

onmousedown
How to use onmousedown to display an alert box when an image is clicked.

onmousedown 2
How to use onmousedown to alert the tag name of the element you clicked on.

onmousemove Event

Definition and Usage

The onmousemove event occurs when the mouse pointer is moved.

Syntax

```
onmousemove="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

onmousemove is, by default, not an event of any object, because mouse movement happens very frequently.

Tips and Notes

Note: Each time a user moves the mouse one pixel, a mousemove event occurs. It takes system resources to process all mousemove events. Use this event carefully!

Example

In the following example we will display an alert box when the user moves the mouse pointer over the image:

```
<img src="image_w3default.gif" alt="W3Schools"
onmousemove="alert('Visit W3Schools!')" />
```

The output of the code above will be:

onmouseout Event

Definition and Usage

The onmouseout event occurs when the mouse pointer moves away from a specified object.

Syntax

```
onmouseout="SomeJavaScriptCode"
```

Parameter	Description

SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

layer, link

Example 1

In the following example we will display an alert box when the user moves the mouse pointer away from the image:

```
<img src="image_w3default.gif" alt="W3Schools"
onmouseout="alert('Visit W3Schools!')" />
```

The output of the code above will be:

Example 2

In the following example we will add an image that should act as a link button on a web page. We will then add an onMouseOver event and an onMouseOut event that will run two JavaScript functions that will change between two images:

```
<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" /></a>
</body>
</html>
```

The output of the code above will be:

onmouseout
How to use onmouseout.

onmouseover Event

Definition and Usage

The onmouseover event occurs when the mouse pointer moves over a specified object.

Syntax

```
onmouseover="SomeJavaScriptCode"
```

Parameter	Description

SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

layer, link

Example 1

In the following example we will display an alert box when the user moves the mouse pointer over the image:

```
<img src="image_w3default.gif" alt="W3Schools"
onmouseover="alert('Visit W3Schools!')" />
```

The output of the code above will be:

Example 2

In the following example we will add an image that should act as a link button on a web page. We will then add an onMouseOver event and an onMouseOut event that will run two JavaScript functions that will change between two images:

```
<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" /></a>
</body>
</html>
```

The output of the code above will be:

onmouseup Event

Definition and Usage

The onmouseup event occurs when a mouse button is released.

Syntax

```
onmouseup="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>, <caption>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <h1> to <h6>, <hr>,

<i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

button, document, link

Example 1

In this example an alert box is displayed when the mouse button is released after clicking the picture:

```
<img src="image_w3default.gif"
onmouseup="alert('You clicked the picture!')">
```

The output of the code above will be (click on the picture):

Example 2

In this example an alert box will alert the tag name of the element you clicked on:

```
<html>
<head>
<script type="text/javascript">
function whichElement(e)
{
var targ
if (!e) var e = window.event
if (e.target) targ = e.target
else if (e.srcElement) targ = e.srcElement
if (targ.nodeType == 3) // defeat Safari bug
targ = targ.parentNode
var tname
tname=targ.tagName
alert("You clicked on a " + tname + " element.")
}
</script>
```

```
</head>
<body onmouseup="whichElement(event)">
<h2>This is a header</h2>
<p>This is a paragraph</p>
<img border="0" src="ball16.gif" alt="Ball">
</body>
</html>
```

onmouseup
How to use onmouseup to display an alert box when an image is clicked.

onmouseup 2
How to use onmouseup to alert the tag name of the element you clicked on.

onreset Event

Definition and Usage

The onreset event occurs when the reset button in a form is clicked.

Syntax

```
onreset="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

```
<form>
```

Supported by the following JavaScript objects:

```
form
```

Example

In this example the form changes back to the default values and displays an alert box when the reset button is clicked:

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

The output of the code above will be:

Firstname:
Lastname:

onresize Event

Definition and Usage

The onresize event occurs when a window or frame is resized.

Syntax

```
onresize="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<a>, <address>, <b>, <big>, <blockquote>, <body>, <button>, <cite>, <code>, <dd>, <dfn>, , <dl>, <dt>, , <fieldset>, <form>, <frame>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>, <li>, <object>, <ol>, <p>, <pre>, <samp>, <select>, <small>, , , <sub>, <sup>, <table>, <textarea>, <tt>, <ul>, <var>

Supported by the following JavaScript objects:

window

Example

In this example an alert box will be displayed when a user tries to resize the window:

```
<body onresize="alert('You have changed the size of the window')">
</body>
```

onselect event

Definition and Usage

The onselect event occurs when text is selected in a text or textarea field.

Syntax

```
onselect="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<input type="text">, <textarea>

Supported by the following JavaScript objects:

text, textarea

Example

In this example an alert box will be displayed if some of the text is selected:

```
<form>
Select text: <input type="text" value="Hello world!"
onselect="alert('You have selected some of the text.')">
<br /><br />
Select text: <textarea cols="20" rows="5"
onselect="alert('You have selected some of the text.')">
Hello world!</textarea>
</form>
```

The output of the code above will be:

Select text:

Select text:

onsubmit event

Definition and Usage

The onsubmit event occurs when the submit button in a form is clicked.

Syntax

```
onsubmit="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

```
<form>
```

Supported by the following JavaScript objects:

form

Example

In this example an alert box displays when a submit box is used:

```
<form name="testform" action="jsref_onsubmit.asp"
onsubmit="alert('Hello ' + testform.fname.value +'!')">
What is your name?<br />
<input type="text" name="fname" />
<input type="submit" value="Submit" />
</form>
```

The output of the code above will be:

What is your name?

onunload Event

Definition and Usage

The onunload event occurs when a user exits a page.

Syntax

```
onunload="SomeJavaScriptCode"
```

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following HTML tags:

<body>, <frameset>

Supported by the following JavaScript objects:

```
window
```

Example

In this example an alert box will be displayed when the page is closed:

```
<body onunload="alert('The onunload event was triggered')">
</body>
```