JAVASCRIPT

Javascript (JS) web sayfalarında kullanılan ve istemci tarafında çalışan (browserlarda çalışan) popüler bir script (programlama) dilidir.

JavaScritp Nedir?

* Javascript Html sayfalarını interaktive (etkileşimli-dinamik) yapmak için kullanılır.
* Bir programlama dilidir.
* Tarayıcı tarafından çalıştırılabilir kodlar içerir.
* Html sayfaları arasına direk gömülür.
* Kodlar yorumlanarak işletiliri. Öncesinde derleme yapılmaz.
* Lisans almadan herkes kullanabilir.
* Java ve JavaScript hem kavram olarak hemde tasarım açısından birbirinden tamamen iki farklı dildir. Java, C# benzeri derlenerek çalışan gelişmiş bir programlama dilidir.

JavaScript ile Ne Yapılabilir?

* JavaScript, Html tasarımlarına programlanabilir bir tasarım imkanı sağlar. Html sayfaları statik sayfalardır. JavaScript dinami sayfalar hazırlama imkanı sağlar
* JS Html sayfalarına dinamik metinler eklemeyi sağlar. Etkileşimli olarak görünümleri değiştirilebilir.
* JS olayları algılar. Örneğin sayfa açıldığında, yada kapandığında, mouse hareketleri vs.
* Html etiketlerinin içeriklerini değiştirebilir.
* Form bilgilerini servera göndermeden önce doğrulamak amacıyla JS kullanılabilir
* Kullanıcının browserını algılayıp, browser tipine göre düzenlemeler yapılabilir.
* Kullanıcının bilgisayarına bazı bilgileri saklamak için Cookies (çerez) atabilir, okuyabilir.

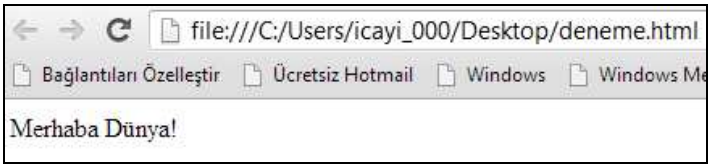
JS 1996 yılında Brendan Eich tarafından gelitirildi ve 1998'de standartlaştırıldı. Standartlarını oluşturma hala devam etmektedir.

JavaScript Nasıl Çalışır

Nasıl çalıştığını test etmek için örneği html sayfası olarak kaydedip deneyin. Örnekte JS ve Html tagları (etiketleri) nasıl kullanıldığını görmüş oluyoruz.

Örnek: Aşağıdaki ilk örneği not defterine atıp deneme.html olarak kaydedip çalıştırın.

```
<html>
<body>
<script type="text/javascript">
document.write("Merhaba Dünya!");
</script>
</body>
</html>
```



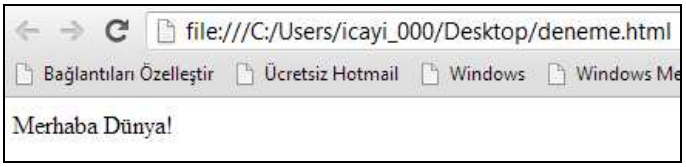
JS kodları Html etiketleri arasına <script> </script> etiketleri içinde konur. "type" parametresi ile script dili ifade edilmiş olur. **document.write** standart sayfaya yazı yazmak için JS kodudur.

JS de Yorum satırı olarak (işletilmeyen satırlar için) C de kullanılan // ifadesi kullanılır. Eğer tarayıcı JS tanımıyor ise o zaman Html de kullanılan <!-- --> ifadesi kullanılmalıdır.

Örnek: Aşağıdaki örnekteki yorum satırlarının nasıl çalıştığını değerlendiriniz.

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

```
<!--
document.write("Merhaba Dünya!");
-->
</script>
</body>
</html>
```



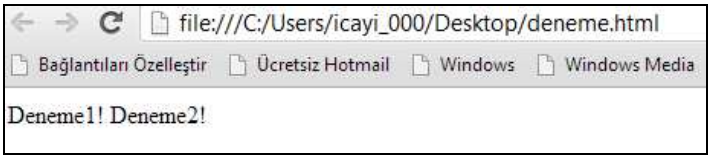
Bu örnekte tarayıcı JS tanıyorsa Html yorum etiketlerini görmeyecektir. Dolayısı ile ekrana Merhaba Dünya yazacaktır. JS yi tanımamış olsaydı o zaman Html etiketini işletecekti. Tarayıcıların JS kodlarını tanımadığında kodları ekrana vermemesi için JS kodlarını Html yorum satırları arasına almak iyi bir yöntem olabilir.

JavaScript Nereye Yazılmalıdır?

JS hem Head hemde Body kısmına yazılabilir. Fakat fonksiyon şeklinde yazılıp çağrılacağı vakit daha çok Head kısmına yazılır.

Örnek: JS nin Body ve Head içerisine yazılması ile ilgili örnekleri deneyiniz.

```
<html>
<head>
<script type="text/javascript">
document.write("Deneme1!");
</script>
</head>
<body>
<script type="text/javascript">
document.write("Deneme2!");
</script>
</body>
```



Kodlar çalışmadı. Çünkü Head içerisindeki JS ler sayfa yüklendiğinde direk çalışmazlar.

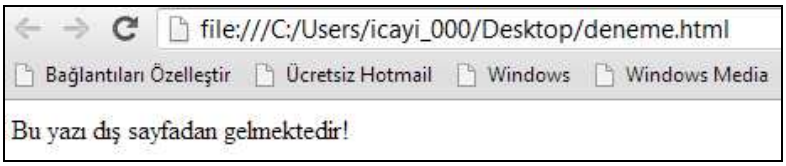
Buraya Fonksiyon ile Çağırma Eklenecek

Sayfa Dışından Çağırma

Bazen hazırlanan JS kodu bir çok sayfa tarafından kullanılabilir. Bu durumda sayfaların dışında ayrı bir dosya olarak JS kodu kaydedilir ve istenilen her sayfanın başında çağrılır. JS kodları .js uzantılı olarak kaydedilir. Dış .js kodlu sayfalar <script> </script> etiketlerini içermemelidir. Bu sayfadaki kodlar direc zaten bu etiketleri bulunduran çağıran sayfanın içerisinde bulunmaktadır.
Örnek: JS kodlarını dış sayfaya yazma ile ilgili aşağıdaki örneği deneyiniz.

Deneme.html

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



Burada dış sayfa çağrılırken src="dene.js" ifadesi kullanılmış ve <script> etiketleri sayfada nerede JS kullanacaksak oraya yazılmıştır. (head yada body) gibi.

Dene.js

```
document.write("Bu yazı dış sayfadan gelmektedir!");
```

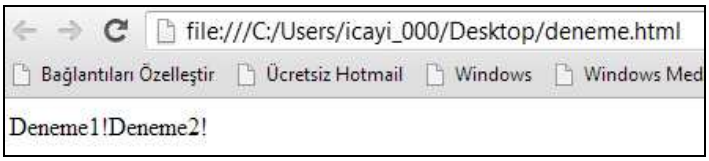
Dış sayfaya yazılan JS kodları dışında <script> etiketi kullanılmamıştır.

JavaScipt İfadeleri

JS ifadeleri tarayıcı tarafından çalıştırılan bir dizi koddan oluşur. Bir JS ifadesi Tarayıcıya ne yapması gerektiğini söyleyen bir komuttur. Çalıştırılabilir her ifadenin sonunda (;) kullanılır. Noktalı virgül bir satırda bir çok satır yazılırken satırları ayırmak kullanılır. Olmadığı zaman tek satır olarak işlem görür.

Örnek: (;) virgül kullanımı ile ilgili aşağıdaki örneği deneyiniz. aradaki noktalıvirgül kaldırılırsa bilgiler görüntülenemeyecektir.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("Deneme1!");document.write("Deneme2!");
</script>
</body>
```



Örnek: JS içerisinde Html tağlarının kullanımını gösteren aşağıdaki örneği deneyiniz.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("<h1>Burası Başlıktır</h1>");
document.write("<p>Burası Bir Paragraftır</p>");
document.write("<p>Burası Başka Bir Paragraftır</p>");
</script>
</body>
```

JavaScript ifadeleri Küme parantezi "{}" ile gruplandırılır. Bu ifadeler arası birlikte çalıştırılır.

JavaScript Comments

JavaScript comments can be used to make the code more readable.

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>
```

Try it yourself.

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>
```

Try it yourself.

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>
```

Try it yourself.

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>
```

Try it yourself.

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:

x=5; length=66.10;

Do You Remember Algebra From School?

Hopefully you remember algebra like this from school: x=5, y=6, z=x+y.

Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above calculate the value of z to be 11?

Sure you do!

These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like **x**, or a more describing name like **length**.

A JavaScript variable can also hold a text value like in **carname="Volvo"**.

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.

Example

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

This example will show you how

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

<script>
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, changes the value,
and displays the value again.</p>

</body>
</html>
```

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 has no values, but you can assign values to the variables while you declare them:

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

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

Assigning Values to JavaScript Variables

You assign values to JavaScript variables with **assignment statements**:

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

The variable name is on the left side of the = sign, and the value you want to assign to the variable is on the right.

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

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that has 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 Arithmetic

As with algebra, you can do arithmetic with JavaScript variables:

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

You will learn more about the operators that can be used between JavaScript variables in the next chapter of this tutorial.

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.

Given that **y=5**, the table below explains the arithmetic operators:

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.

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

<script>
var x;
x=5+5;
document.write(x);
document.write("<br>");
x="5"+"5";
document.write(x);
document.write("<br>");
x=5+"5";
document.write(x);
document.write("<br>");
x="5"+5;
document.write(x);
document.write("<br>");
</script>

<p>The rule is: If you add a number and a string, the result will be a string.</p>

</body>
</html>
```

The rule is:

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

JavaScript Comparison and Logical Operators

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 in 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

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>
```

JavaScript Switch Statement

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>
```

JavaScript Popup Boxes

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");
```

JavaScript Functions

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.

Soru: Ekran çıktısı ne olur

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

<script >

x= 5 + 5 +"5" + 5 + 5 ;

document.write(x);

</script>

</body>
</html>
```

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.

JavaScript For 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.

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.

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.

Examples

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

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

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
```

JavaScript Break and Continue

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

Examples

Break statement
Use the break statement to break the loop.

Continue statement
Use the continue statement to break the current loop and continue with the next value.

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
```

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.

Examples

For...In statement
How to use a for...in statement to loop through the elements of an array.

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>
```

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.

Examples

The try...catch statement
How to write a try...catch statement.

The try...catch statement with a confirm box
Another example of how to write a try...catch statement.

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.

JavaScript Throw Statement

The throw statement allows you to create an exception.

Examples

The throw statement
How to use the throw statement.

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>
```

JavaScript The onerror Event

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

Examples

The onerror event (an example with an error)
How to use the onerror event 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>
```

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

Previous Next

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

Examples

Return the length of a string
How to use the length property to find the length of a string.

Style strings
How to style strings.

The indexOf() method
How to use the indexOf() method to return the position of the first occurrence of a specified string value in a string.

The match() method
How to use the match() method to search for a specified string value within a string and return the string value if found

Replace characters in a string - replace()
How to use the replace() method to replace some characters with some other characters in a string.

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!
```

JavaScript Date Object

Previous Next

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

Examples

Return today's date and time
How to use the Date() method to get today's date.

getTime()
Use getTime() to calculate the years since 1970.

setFullYear()
How to use setFullYear() to set a specific date.

toUTCString()
How to use toUTCString() to convert today's date (according to UTC) to a string.

getDay()
Use getDay() and an array to write a weekday, and not just a number.

Display a clock
How to display a clock on your web page.

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!

Defining Dates

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

We define a Date object with the new keyword. The following code line defines 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!

Manipulate 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!

Comparing 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");
}
```

JavaScript Array Object

Previous Next

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

Examples

Create an array
Create an array, assign values to it, and write the values to the output.

For...In Statement
How to use a for...in statement to loop through the elements of an array.

Join two arrays - concat()
How to use the concat() method to join two arrays.

Put array elements into a string - join()
How to use the join() method to put all the elements of an array into a string.

Literal array - sort()
How to use the sort() method to sort a literal array.

Numeric array - sort()
How to use the sort() method to sort a numeric array.

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!

Defining Arrays

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

We define an Array object with the new keyword. The following code line defines an Array object called myArray:

```
var myArray=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.

Accessing Arrays

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 Existing Arrays

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
```

JavaScript Boolean Object

Previous Next

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

Examples

Check Boolean value
Check if a Boolean object is 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!

Boolean Object

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

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

We define a Boolean object with the new keyword. The following code line defines 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");
```

JavaScript Math Object

Previous Next

The Math object allows you to perform common mathematical tasks.

Examples

round()
How to use round().

random()
How to use random() to return a random number between 0 and 1.

max()
How to use max() to return the number with the highest value of two specified numbers.

min()
How to use min() to return the number with the lowest value of two specified numbers.

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 common mathematical tasks.

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

Mathematical Values

JavaScript provides eight mathematical values (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 values 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 values that can be accessed from the Math object there are also several functions (methods) available.

Examples of functions (methods):

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:

```
0.48860152602999024
```

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:

```
8
```

JavaScript RegExp Object

Previous Next

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

What is RegExp

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
```

Try it yourself

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
```

Try it yourself

---

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
```

Try it yourself

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

JavaScript HTML DOM Objects

Previous Next

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 Browser Detection

Previous Next

The JavaScript Navigator object contains information about the visitor's browser.

Examples

Detect the visitor's browser and browser version

More details about the visitor's browser

All details about the visitor's browser

Alert user, depending on browser

Browser Detection

Almost everything in this tutorial works on all JavaScript-enabled browsers. However, there are some things that just don't work on certain browsers - specially on older browsers.

So, sometimes it can be very useful to detect the visitor's browser type and version, and then serve up the appropriate information.

The best way to do this is to make your web pages smart enough to look one way to some browsers and another way to other browsers.

JavaScript includes an object called the Navigator object, that can be used for this purpose.

The Navigator object contains information about the visitor's browser name, browser version, and more.

The Navigator Object

The JavaScript Navigator object contains all information about the visitor's browser. We are going to look at two properties of the Navigator object:

- appName - holds the name of the browser
- appVersion - holds, among other things, the version of the browser

Example

```
<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>
```

The variable browser in the example above holds the name of the browser, i.e. "Netscape" or "Microsoft Internet Explorer".

The appVersion property in the example above returns a string that contains much more information than just the version number, but for now we are only interested in the version number. To pull the version number out of the string we are using a function called parseFloat(), which pulls the first thing that looks like a decimal number out of a string and returns it.

IMPORTANT! The version number is WRONG in IE 5.0 or later! Microsoft starts the appVersion string with the number 4.0. in IE 5.0 and IE 6.0!!! Why did they do that??? However, JavaScript is the same in IE6, IE5 and IE4, so for most scripts it is ok.

Example

The script below displays a different alert, depending on the visitor's browser:

```
<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

Previous Next

A cookie is often used to identify a user.

Examples

Create a welcome cookie

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.

JavaScript Form Validation

Previous Next

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

Previous Next

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>
```

JavaScript Image Maps

Previous Next

An image-map is an image with clickable regions.

Examples

Simple HTML Image map

Image map with added JavaScript

HTML Image Maps

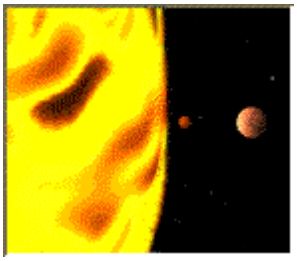
From our HTML tutorial we have learned that an image-map is an image with clickable regions. Normally, each region has an associated hyperlink. Clicking on one of the regions takes you to the associated link.

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>
```

JavaScript Timing Events

Previous Next

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>
```

JavaScript Create Your Own Objects

Previous Next

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").

You Have Learned JavaScript, Now What?

Previous Next

JavaScript Summary

This tutorial has taught you how to add JavaScript to your HTML pages, to make your web site more dynamic and interactive.

You have learned how to create responses to events, validate forms and how to make different scripts run in response to different scenarios.

You have also learned how to create and use objects, and how to use JavaScript's built-in objects.

For more information on JavaScript, please look at our JavaScript examples and our JavaScript reference.

Now You Know JavaScript, What's Next?

The next step is to learn about the HTML DOM and DHTML.

If you want to learn about server-side scripting, the next step is to learn ASP.

HTML DOM

The HTML DOM defines a standard way for accessing and manipulating HTML documents.

The HTML DOM is platform and language independent and can be used by any programming language like Java, JavaScript, and VBScript.

If you want to learn more about the DOM, please visit our HTML DOM tutorial.

DHTML

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

W3C once said: "Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."

If you want to learn more about DHTML, please visit our DHTML tutorial.

ASP

While scripts in an HTML file are executed on the client (in the browser), scripts in an ASP file are executed on the server.

With ASP you can dynamically edit, change or add any content of a Web page, respond to data submitted from HTML forms, access any data or databases and return the results to a browser, customize a Web page to make it more useful for individual users.

Since ASP files are returned as plain HTML, they can be viewed in any browser.

If you want to learn more about ASP, please visit our ASP tutorial.

JavaScript Examples

Previous Next

Basic JavaScript Examples

Write text with JavaScript
Write HTML tags with JavaScript
JavaScript in the body section
JavaScript in the head section
An external JavaScript

Examples explained

JavaScript Statements, Comments and Blocks

JavaScript statements.
JavaScript blocks.
Singe line comments.
Multiple lines comments.
Single line comment to prevent execution.
Multiple lines comment to prevent execution.

Examples explained

JavaScript Variables

Declare a variable, assign a value to it, and display it

Examples explained

JavaScript Conditional If ... Else

If statement
If...else statement
Random link
Switch statement

Examples explained

JavaScript Popup Boxes

Alert box
Alert box with line breaks
Confirm box
Prompt box

Examples explained

JavaScript Functions

Call a function
Function with an argument
Function with an argument 2

Function that returns a value
Function with arguments, that returns a value

Examples explained

JavaScript Loops

For loop
Looping through HTML headers
While loop
Do While loop
Break a loop
Break and continue a loop
Use a for...in statement to loop through the elements of an array

Examples explained

JavaScript Error Handling

The try...catch statement
The try...catch statement with a confirm box
The onerror event

Examples explained

Advanced JavaScript Examples

Detect the visitor's browser and browser version
More details about the visitor's browser
All details about the visitor's browser
Alert user, depending on browser
Create a welcome cookie
Button animation
Image map with added JavaScript
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
Create a direct instance of an object
Create a template for an object

JavaScript Objects Examples

Previous Next

Examples of using the built-in JavaScript objects.

String Object

Return the length of a string
Style strings
Return the position of the first occurrence of a text in a string - indexOf()
Search for a text in a string and return the text if found - match()
Replace characters in a string - replace()

More String object examples in our JavaScript reference.

Date Object

Use Date() to return today's date and time
Use getTime() to calculate the years since 1970
Use setFullYear() to set a specific date
Use toUTCString() to convert today's date (according to UTC) to a string
Use getDay() and an array to write a weekday, and not just a number
Display a clock

More Date object examples in our JavaScript reference.

Array Object

Create an array
Use a for...in statement to loop through the elements of an array
Join two arrays - concat()
Put array elements into a string - join()
Literal array - sort()
Numeric array - sort()

More Array object examples in our JavaScript reference.

Boolean Object

Check Boolean value

More Boolean object examples in our JavaScript reference.

Math Object

Use round() to round a number
Use random() to return a random number between 0 and 1
Use max() to return the number with the highest value of two specified numbers
Use min() to return the number with the lowest value of two specified numbers
Convert Celsius to Fahrenheit

More Math object examples in our JavaScript reference.

JavaScript HTML DOM Examples

Previous Next

Examples of using JavaScript to access and manipulate the HTML DOM objects.

Anchor Object

Change text, URL, and target attribute of a link
Using focus() and blur()
Add an accessKey to a link

Document Object

Write text to the output
Write text with formatting to the output
Return the title of a document
Return the URL of a document
Return the referrer of a document
Return the domain name of the document's server
Use getElementById()
Use getElementsByName()
Open a new document, specify MIME type and add some text
Return the number of anchors in a document
Return the innerHTML of the first anchor in a document
Count the number of forms in a document
Access an item in a collection
Count the number of images in a document

Event Object

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?

Form and Form Input Objects

View and change the action URL of a form
View the method that is to be used when sending form data
Alert id, type, and value of a Button object + disable button
Check and uncheck a checkbox
Checkboxes in a form
Checkbox - If the user clicks in a checkbox, the content of the text fields are converted to uppercase.
Radio buttons
Reset a form
Submit a form
Form validation
Set focus to an input field when the page loads
Select the content of an input field
Dropdown list in a form
Another dropdown list
A dropdown menu
Jump to the next field when the current field's maxlength has been reached
Add accessKeys to form fields

Frame, Frameset, and IFrame Objects

Resizable and not resizable frames
Frames with and without scrollbars

Change the source / URL of two frames
Break out of a frame
Update two iframes

Image Object

Change the height and width of an image
Change the src of an image

Location Object

Send the client to a new location / URL
Reload a page
Break out of a frame
Anchors array - This example opens two windows. The first window contains four buttons and the second window defines four anchors from 0 to 3. When a button is clicked in the first window, the onclick event handler goes to the specified anchor in the second window.

Navigator Object

Detect the visitor's browser and browser version
More details about the visitor's browser
All details about the visitor's browser
Alert user, depending on browser

Option and Select Objects

Disable and enable a dropdown list
Get the id of the form that contains the dropdown list
Get the number of options in the dropdown list
Turn the dropdown list into a multiline list
Select multiple options in a dropdown list
Alert the selected option in a dropdown list
Alert the index of the selected option in a dropdown list
Change the text of the selected option
Remove options from a dropdown list

Screen Object

Detect details about the client's screen

Table, TableHeader, TableRow, TableData Objects

Change the width of a table border
Change the cellPadding and cellSpacing of a table
Specify frames of a table
Specify rules for a table
InnerHTML of a row
InnerHTML of a cell
Create a caption for a table
Delete rows in a table
Add rows to a table
Add cells to a table row
Align the cell content in a table row
Vertical align the cell content in a table row
Align the cell content in a single cell
Vertical align the cell content in a single cell
Change the content of a table cell
Change the colspan of a table row

Window Object

Display an alert box
Alert box with line-breaks
Display a confirm box

JavaScript Quiz Test

Previous | Next

You can test your JavaScript skills with W3Schools' Quiz.

The Test

The test contains 20 questions and there is no time limit.

The test is not official, it's just a nice way to see how much you know, or don't know, about JavaScript.

Your Score Will be Counted

You will get 1 point for each correct answer. At the end of the Quiz, your total score will be displayed. Maximum score is 20 points.

Good luck! Start the JavaScript Quiz

W3Schools JAVASCRIPT Quiz

1. Inside which HTML element do we put the JavaScript?

- <scripting>
- <js>
- <javascript>
- <script>

2. What is the correct JavaScript syntax to write "Hello World"?

- document.write("Hello World")
- "Hello World"
- response.write("Hello World")
- ("Hello World")

3. Where is the correct place to insert a JavaScript?

- The <head> section
- The <body> section
- Both the <head> section and the <body> section are correct

4. What is the correct syntax for referring to an external script called "xxx.js"?

- <script src="xxx.js">
- <script href="xxx.js">
- <script name="xxx.js">

5. An external JavaScript must contain the <script> tag

- False
- True

6. How do you write "Hello World" in an alert box?

- alertBox("Hello World")
- msgBox("Hello World")
- alertBox="Hello World"
- alert("Hello World")

7. How do you create a function?

- function=myFunction()
- function:myFunction()
- function myFunction()

8. How do you call a function named "myFunction"?

- call myFunction()
- call function myFunction
- myFunction()

9. How do you write a conditional statement for executing some statements only if "i" is equal to 5?

- if i=5 then
- if i=5
- if i==5 then
- if (i==5)

10. How do you write a conditional statement for executing some statements only if "i" is NOT equal to 5?

- if <>5
- if =! 5 then
- if (i != 5)
- if (i <> 5)

11. How many different kind of loops are there in JavaScript?

- Two. The "for" loop and the "while" loop
- One. The "for" loop
- Four. The "for" loop, the "while" loop, the "do...while" loop, and the "loop...until" loop

12. How does a "for" loop start?

- for (i = 0; i <= 5; i++)
- for (i = 0; i <= 5)
- for (i <= 5; i++)
- for i = 1 to 5

13. How can you add a comment in a JavaScript?

- 'This is a comment
- <!--This is a comment-->
- //This is a comment

14. What is the correct JavaScript syntax to insert a comment that has more than one line?

- /*This comment has
more than one line*/

- <!--This comment has
more than one line-->
- //This comment has
more than one line//

15. What is the correct way to write a JavaScript array?

- var txt = new Array("tim","kim","jim")
- var txt = new Array="tim","kim","jim"
- var txt = new Array:1=("tim")2=("kim")3=("jim")
- var txt = new Array(1:"tim",2:"kim",3:"jim")

16. How do you round the number 7.25, to the nearest whole number?

- rnd(7.25)
- round(7.25)
- Math.rnd(7.25)
- Math.round(7.25)

17. How do you find the largest number of 2 and 4?

- Math.ceil(2,4)
- top(2,4)
- Math.max(2,4)
- ceil(2,4)

18. What is the correct JavaScript syntax for opening a new window called "window2" ?

- new("http://www.w3schools.com","window2")
- open.new("http://www.w3schools.com","window2")
- window.open("http://www.w3schools.com","window2")
- new.window("http://www.w3schools.com","window2")

19. How do you put a message in the browser's status bar?

- window.status = "put your message here"
- statusbar = "put your message here"
- status("put your message here")
- window.status("put your message here")

20. How do you find the client's browser name?

- client.navName
- browser.name
- navigator.appName