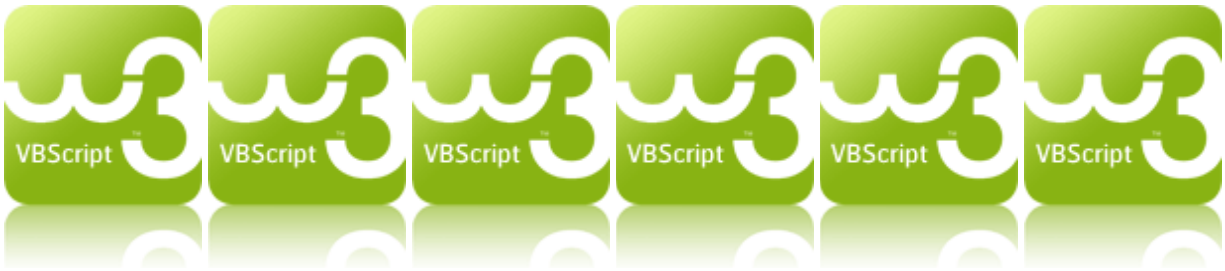
w3schools.com

Vbscript Complete Tutorial



VBScript Tutorial

« W3Schools Home Next Chapter »

VBScript is a Microsoft scripting language.

VBScript is the default scripting language in ASP.

Client-side VBScript only works in Internet Explorer !!!

Start learning VBScript now!

VBScript Editor

With our online editor, you can edit the VBScript code, and click on a button to view the result.

Example (IE Only)

```
<html>
<body>

<script type="text/vbscript">
document.write("This is my first VBScript!")
</script>

</body>
</html>
```

Try it yourself »

Click on the "Try it Yourself" button to see how it works.

VBScript Examples

Learn by examples! With our editor, you can edit the source code, and click on a test button to view the result.

Try-it-Yourself!

VBScript Reference

At W3Schools you will find a complete VBScript reference.

VBScript Reference

« W3Schools Home Next Chapter »

VBScript Introduction

« Previous Next Chapter »

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML / XHTML

If you want to study these subjects first, find the tutorials on our Home page.

What is VBScript?

- VBScript is a scripting language
- A scripting language is a lightweight programming language
- VBScript is a light version of Microsoft's programming language Visual Basic
- **VBScript is only supported by Microsoft's browsers (Internet Explorer)**

How Does it Work?

When a VBScript is inserted into an HTML document, Internet Explorer browser will read the HTML and interpret the VBScript. The VBScript can be executed immediately, or at a later event.

VBScript only works in Microsoft browsers (Internet Explorer).

« Previous Next Chapter »

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

Put a VBScript into an HTML Page

The example below shows how to use VBScript to write text on a web page:

Example (IE Only)

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

Try it yourself »

The example below shows how to add HTML tags to the VBScript:

Example (IE Only)

```
<html>
<body>
<script type="text/vbscript">
document.write("<h1>Hello World!</h1>")
</script>
</body>
</html>
```

Try it yourself »

Example Explained

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

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

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

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

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

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

How to Handle Simple Browsers

Browsers that do not support scripting, will display VBScript as page content.

To prevent them from doing this, the HTML comment tag should be used to "hide" the VBScript.

Just add an HTML comment tag <!-- before the first VBScript statement, and a --> (end of comment) after the last VBScript statement, like this:

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

```
<!--
document.write("Hello World!")
-->
</script>
</body>
</html>
```

« Previous

Next Chapter »

VBScript Where To ...

VBScripts can be placed in the body and in the head section of an HTML document.

Where to Put the VBScript

VBScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, or at a later event, such as when a user clicks a button. When this is the case we put the script inside a function or a sub procedure, you will learn about procedures in a later chapter.

Scripts in <head>

Put your functions and sub procedures in the head section, this way they are all in one place, and they do not interfere with page content.

Example (IE Only)

```
<html>
<head>
<script type="text/vbscript">
function myFunction()
alert("Hello World!")
end function
</script>
</head>

<body onload="myFunction()">
</body>
</html>
```

Try it yourself »

Scripts in <body>

If you don't want your script to be placed inside a function, and especially if your script should write page content, it should be placed in the body section.

Example (IE Only)

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

<body>
<script type="text/vbscript">
document.write("This message is written by VBScript")
</script>
</body>

</html>
```

Try it yourself »

Scripts in <head> and <body>

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

Example (IE Only)

```
<html>
<head>
<script type="text/vbscript">
function myFunction()
alert("Hello World!")
end function
</script>
</head>

<body>
<button onclick="myFunction()">Click me</button>
<script type="text/vbscript">
document.write("This message is written by VBScript")
</script>
</body>
</html>
```

Try it yourself »

Using an External VBScript

If you want to run the same VBScript on several pages, without having to write the same script on every page, you can write a VBScript in an external file.

Save the external VBScript file with a .vbs file extension.

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

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

Example

```
<html>
<head>
<script type="text/vbscript" src="ex.vbs"></script>
</head>
<body>
</body>
</html>
```

Try it yourself »

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

« Previous

Next Chapter »

VBScript Variables

« Previous Next Chapter »

Variables are "containers" for storing information.

Do You Remember Algebra from School?

Do you remember algebra 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 to calculate the value of z to be 11?

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

VBScript Variables

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

A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for VBScript variable names:

- Must begin with a letter
- Cannot contain a period (.)
- Cannot exceed 255 characters

In VBScript, all variables are of type *variant*, that can store different types of data.

Declaring (Creating) VBScript Variables

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

You can declare VBScript variables with the Dim, Public or the Private statement. Like this:

```
Dim x
Dim carname
```

Now you have created two variables. The name of the variables are "x" and "carname".

You can also declare variables by using its name in a script. Like this:

```
carname="Volvo"
```

Now you have also created a variable. The name of the variable is "carname". However, this method is not a good practice, because you can misspell the variable name later in your script, and that can cause strange results when your script is running.

If you misspell for example the "carname" variable to "carnime", the script will automatically create a new variable called "carnime". To prevent your script from doing this, you can use the Option Explicit statement. This statement forces you to declare all your variables with the dim, public or private statement.

Put the Option Explicit statement on the top of your script. Like this:

```
Option Explicit
Dim carname
carname=some value
```

Assigning Values to Variables

You assign a value to a variable like this:

```
carname="Volvo"
x=10
```

The variable name is on the left side of the expression and the value you want to assign to the variable is on the right. Now the variable "carname" has the value of "Volvo", and the variable "x" has the value of "10".

Lifetime of Variables

How long a variable exists is its lifetime.

When you declare a variable within a procedure, the variable can only be accessed within that procedure. When the procedure exits, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different procedures, because each is recognized only by the procedure in which it is declared.

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

VBScript Array Variables

An array variable is used to store multiple values in a single variable.

In the following example, an array containing 3 elements is declared:

```
Dim names(2)
```

The number shown in the parentheses is 2. We start at zero so this array contains 3 elements. This is a fixed-size array. You assign data to each of the elements of the array like this:

```
names(0)="Tove"
names(1)="Jani"
names(2)="Stale"
```

Similarly, the data can be retrieved from any element using the index of the particular array element you want. Like this:

```
mother=names(0)
```

You can have up to 60 dimensions in an array. Multiple dimensions are declared by separating the numbers in the parentheses with commas. Here we have a two-dimensional array consisting of 5 rows and 7 columns:

```
Dim table(4,6)
```

Asign data to a two-dimensional array:

Example (IE Only)

```
<html>
<body>

<script type="text/vbscript">
Dim x(2,2)
x(0,0)="Volvo"
x(0,1)="BMW"
x(0,2)="Ford"
x(1,0)="Apple"
x(1,1)="Orange"
x(1,2)="Banana"
x(2,0)="Coke"
x(2,1)="Pepsi"
x(2,2)="Sprite"
for i=0 to 2
    document.write("<p>")
    for j=0 to 2
        document.write(x(i,j) & "<br />")
    next
    document.write("</p>")
next
</script>

</body>
</html>
```

Try it yourself »

« Previous Next Chapter »

VBScript Procedures

« Previous Next Chapter »

VBScript has two kinds procedures:

- Sub procedure
- Function procedure

VBScript Sub Procedures

A Sub procedure:

- is a series of statements, enclosed by the Sub and End Sub statements
- can perform actions, but **does not return** a value
- can take arguments

```
Sub mysub()
  some statements
End Sub
```

or

```
Sub mysub(argument1,argument2)
  some statements
End Sub
```

Example (IE Only)

```
Sub mysub()
  alert("Hello World")
End Sub
```

Try it yourself »

VBScript Function Procedures

A Function procedure:

- is a series of statements, enclosed by the Function and End Function statements
- can perform actions and **can return** a value
- can take arguments that are passed to it by a calling procedure
- without arguments, must include an empty set of parentheses ()
- returns a value by assigning a value to its name

```
Function myfunction()
  some statements
  myfunction=some value
End Function
```

or

```
Function myfunction(argument1,argument2)
  some statements
  myfunction=some value
End Function
```

Example (IE Only)

```
function myfunction()
  myfunction=Date()
end function
```

Try it yourself »

How to Call a Procedure

There are different ways to call a procedure. You can call it from within another procedure, on an event, or call it within a script.

Example (IE Only)

Call a procedure when the user clicks on a button:

```
<body>
<button onclick="myfunction()">Click me</button>
</body>
```

Try it yourself »

Procedures can be used to get a variable value:

```
carname=findname()
```

Here you call a Function called "findname", the Function returns a value that will be stored in the variable "carname".

Function procedures can calculate the sum of two arguments:

Example (IE Only)

```
Function myfunction(a,b)
myfunction=a+b
End Function

document.write(myfunction(5,9))
```

Try it yourself »

The function "myfunction" will return the sum of argument "a" and argument "b". In this case 14.

When you call a procedure you can use the Call statement, like this:

```
Call MyProc(argument)
```

Or, you can omit the Call statement, like this:

```
MyProc argument
```

« Previous Next Chapter »

VBScript Conditional Statements

« Previous Next Chapter »

Conditional Statements

Conditional statements are used to perform different actions for different decisions.

In VBScript we have four conditional statements:

- **If statement** - executes a set of code when a condition is true
- **If...Then...Else statement** - select one of two sets of lines to execute
- **If...Then...ElseIf statement** - select one of many sets of lines to execute
- **Select Case statement** - select one of many sets of lines to execute

If...Then...Else

Use the If...Then...Else statement if you want to

- execute some code if a condition is true
- select one of two blocks of code to execute

If you want to execute only **one** statement when a condition is true, you can write the code on one line:

```
If i=10 Then alert("Hello")
```

There is no ..Else.. in this syntax. You just tell the code to perform **one action** if a condition is true (in this case If i=10).

If you want to execute **more than one** statement when a condition is true, you must put each statement on separate lines, and end the statement with the keyword "End If":

```
If i=10 Then
alert("Hello")
i = i+1
End If
```

There is no ..Else.. in the example above either. You just tell the code to perform **multiple actions** if the condition is true.

If you want to execute a statement if a condition is true and execute another statement if the condition is not true, you must add the "Else" keyword:

Example (IE Only)

```
<html>
<body>
<head>
<script type="text/vbscript">
Function greeting()
i=hour(time)
If i < 10 Then
  document.write("Good morning!")
Else
  document.write("Have a nice day!")
End If
End Function
</script>
</head>

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

</html>
```

Try it yourself »

In the example above, the first block of code will be executed if the condition is true, and the other block will be executed otherwise (if i is greater than 10).

If...Then...ElseIf

You can use the If...Then...ElseIf statement if you want to select one of many blocks of code to execute:

Example (IE Only)

```
<html>
<body>
<head>
<script type="text/vbscript">
Function greeting()
i=hour(time)
If i = 10 Then
  document.write("Just started...!")
ElseIf i = 11 then
  document.write("Hungry!")
ElseIf i = 12 then
  document.write("Ah, lunch-time!")
ElseIf i = 16 then
  document.write("Time to go home!")
Else
  document.write("Unknown")
End If
End Function
</script>
</head>

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

</html>
```

Try it yourself »

Select Case

You can also use the "Select Case" statement if you want to select one of many blocks of code to execute:

Example (IE Only)

```
<html>
<body>
<script type="text/vbscript">
d=weekday(date)
Select Case d
  Case 1
    document.write("Sleepy Sunday")
  Case 2
    document.write("Monday again!")
  Case 3
    document.write("Just Tuesday!")
  Case 4
    document.write("Wednesday!")
  Case 5
    document.write("Thursday...")
  Case 6
    document.write("Finally Friday!")
  Case else
    document.write("Super Saturday!!!!")
End Select
</script>

</body>
</html>
```

Try it yourself »

This is how it works: First we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each Case in the structure. If there is a match, the block of code associated with that Case is executed.

« Previous Next Chapter »

VBScript Looping

« Previous Next Chapter »

Looping Statements

Looping statements are used to run the same block of code a specified number of times.

In VBScript we have four looping statements:

- **For...Next statement** - runs code a specified number of times
- **For Each...Next statement** - runs code for each item in a collection or each element of an array
- **Do...Loop statement** - loops while or until a condition is true
- **While...Wend statement** - Do not use it - use the Do...Loop statement instead

For...Next Loop

Use the **For...Next** statement to run a block of code a specified number of times.

The **For** statement specifies the counter variable (**i**), and its start and end values. The **Next** statement increases the counter variable (**i**) by one.

Example

```
<html>
<body>

<script type="text/vbscript">
For i = 0 To 5
  document.write("The number is " & i & "<br />")
Next
</script>

</body>
</html>
```

Try it yourself »

The Step Keyword

With the **Step** keyword, you can increase or decrease the counter variable by the value you specify.

In the example below, the counter variable (**i**) is INCREASED by two, each time the loop repeats.

```
For i=2 To 10 Step 2
  some code
Next
```

To decrease the counter variable, you must use a negative **Step** value. You must specify an end value that is less than the start value.

In the example below, the counter variable (**i**) is DECREASED by two, each time the loop repeats.

```
For i=10 To 2 Step -2
  some code
Next
```

Exit a For...Next

You can exit a For...Next statement with the Exit For keyword.

```
For i=1 To 10
  If i=5 Then Exit For
  some code
Next
```

For Each...Next Loop

A **For Each...Next** loop repeats a block of code for each item in a collection, or for each element of an array.

Example

```
<html>
<body>

<script type="text/vbscript">
Dim cars(2)
cars(0)="Volvo"
cars(1)="Saab"
cars(2)="BMW"

For Each x In cars
  document.write(x & "<br />")
Next
</script>

</body>
</html>
```

Try it yourself »

Do...Loop

If you don't know how many repetitions you want, use a Do...Loop statement.

The Do...Loop statement repeats a block of code while a condition is true, or until a condition becomes true.

Repeat Code While a Condition is True

You use the While keyword to check a condition in a Do...Loop statement.

```
Do While i>10
  some code
Loop
```

If **i** equals 9, the code inside the loop above will never be executed.

```
Do
  some code
Loop While i>10
```

The code inside this loop will be executed at least one time, even if **i** is less than 10.

Repeat Code Until a Condition Becomes True

You use the Until keyword to check a condition in a Do...Loop statement.

```
Do Until i=10
  some code
Loop
```

If **i** equals 10, the code inside the loop will never be executed.

```
Do
  some code
Loop Until i=10
```

The code inside this loop will be executed at least one time, even if **i** is equal to 10.

Exit a Do...Loop

You can exit a Do...Loop statement with the Exit Do keyword.

```
Do Until i=10
  i=i-1
  If i<10 Then Exit Do
Loop
```

The code inside this loop will be executed as long as **i** is different from 10, and as long as **i** is greater than 10.

« Previous Next Chapter »

You Have Learned VBScript, Now What?

VBScript Summary

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

You have learned how to create variables and functions, and how to make different scripts run in response to different scenarios.

For more information on VBScript, please look at our VBScript examples and our VBScript references.

Now You Know VBScript, What's Next?

The next step is to learn 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.

VBScript Examples

« Previous Next Chapter »

Basic

Write text using VBScript
Format text with HTML tags
A function in the head section
A script in the body section

Variables

Create a variable
Insert a variable value in a text
Create an array

Procedures

Sub procedure
Function procedure

Conditional Statements

If...Then..Else statement
If...Then..ElseIf statement
Select Case statement
Random link

Looping

For..Next loop
Looping through the HTML headers
For..Each loop
Do...While loop

Date and Time Functions

Display Date and Time
Display the days
Display the months
Display the current month and day
Countdown to year 3000
Add a time interval to a date
Format Date and Time
Is this a date?

Other Built-in Functions

Uppercase or lowercase characters?
Remove leading or trailing spaces from a string
Reverse a string
Round a number
Return a random number
Return a random number between 0-99
Return a specified number of characters from the left or right side of a string
Replace some characters in a string
Return a specified number of characters from a string

« Previous Next Chapter »

Edit and Click Me »

Your Result:

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

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

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
document.write("<h1>Hello World!</h1>")
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

```
<!DOCTYPE html>
<html>
<head>
<script type="text/vbscript">
function myFunction()
alert("Hello World!")
end function
</script>
</head>

<body onload="myFunction()">

<p>We usually use the head section for functions (to be sure that the functions are loaded before they are called).</p>

</body>
</html>
```

Your Result:

We usually use the head section for functions (to be sure that the functions are loaded before they are called).

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
document.write("This message is written by VBScript")
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
Dim 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, and displays the value. Then, it changes the value, and displays the value again.</p>

</body>
</html>
```

The script above declares a variable, assigns a value to it, and displays the value. Then, it changes the value, and displays the value again.

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
Dim name
name="Jan Egil"
document.write("My name is: " & name)
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
Dim famname(5)
famname(0)="Jan Egil"
famname(1)="Tove"
famname(2)="Hege"
famname(3)="Stale"
famname(4)="Kai Jim"
famname(5)="Borge"
For i=0 To 5
  document.write(famname(i) & "<br />")
Next
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

```
<!DOCTYPE html>
<html>

<head>
<script type="text/vbscript">
Sub mySub()
  msgbox("This is a Sub procedure")
End Sub
</script>
</head>

<body>
<script type="text/vbscript">
Call mySub()
</script>
<p>A Sub procedure does not return a result.</p>
</body>
</html>
```

Your Result:

A Sub procedure does not return a result.

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

```
<!DOCTYPE html>
<html>

<head>
<script type="text/vbscript">
Function myFunction()
 myFunction = "BLUE"
End Function
</script>
</head>

<body>
<script type="text/vbscript">
document.write("My favorite color is " & myFunction())
</script>
<p>A Function procedure can return a result.</p>
</body>

</html>
```

A Function procedure can return a result.

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

```
<!DOCTYPE html>
<html>

<head>
<script type="text/vbscript">
Function greeting()
i=hour(time)
If i < 10 Then
 document.write("Good morning!")
Else
 document.write("Have a nice day!")
End If
End Function
</script>
</head>

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

</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

```
<!DOCTYPE html>
<html>

<head>
<script type="text/vbscript">
Function greeting()
i=hour(time)
If i = 10 Then
      document.write("Just started...!")
ElseIf i = 11 Then
      document.write("Hungry!")
ElseIf i = 12 Then
      document.write("Ah, lunch-time!")
ElseIf i = 16 Then
      document.write("Time to go home!")
Else
      document.write("Unknown")
End If
End Function
</script>
</head>

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

</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

```
<!DOCTYPE html>
<html>

<body>
<script type="text/vbscript">
d=weekday(Date)

Select Case d
  Case 1
    document.write("Sleepy Sunday")
  Case 2
    document.write("Monday again!")
  Case 3
    document.write("Just Tuesday!")
  Case 4
    document.write("Wednesday!")
  Case 5
    document.write("Thursday...")
  Case 6
    document.write("Finally Friday!")
  Case Else
    document.write("Super Saturday!!!!")
End Select
</script>

<p>This example demonstrates the "Select Case" statement.<br />
You will receive a different greeting based on what day it is.<br />
Note that Sunday=1, Monday=2, Tuesday=3, etc.</p>

</body>
</html>
```

This example demonstrates the "Select Case" statement.
You will receive a different greeting based on what day it is.
Note that Sunday=1, Monday=2, Tuesday=3, etc.

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
randomize()
r=rnd()
If r>0.5 Then
 document.write("<a href='http://www.w3schools.com'>Learn Web
Development!</a>")
Else
 document.write("<a href='http://www.refsnesdata.no'>Visit Refsnes Data!</a>")
End If
</script>

<p>
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.
</p>
</body>
</html>
```

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.

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
For i = 0 To 5
 document.write("The number is " & i & "<br />")
Next
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
For i=1 To 6
 document.write("<h" & i & ">This is header " & i &
"</h" & i & ">")
Next
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
Dim cars(2)
cars(0)="Volvo"
cars(1)="Saab"
cars(2)="BMW"

For Each x In cars
  document.write(x & "<br />")
Next
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
i=0
Do While i < 10
  document.write(i & "<br />")
  i=i+1
Loop
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
document.write("Today's date is " & Date())
document.write("<br />")
document.write("The time is " & Time())
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

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

<p>VBScripts' function <b>WeekdayName</b> is used to get a weekday:</p>
<script type="text/vbscript">
document.write("<p>")
document.write(WeekDayName(1))
document.write("<br />")
document.write(WeekDayName(2))
document.write("</p><p>")

document.write("Get the abbreviated name of a weekday:")
document.write("<br />")
document.write(WeekDayName(1,True))
document.write("<br />")
document.write(WeekDayName(2,True))
document.write("</p><p>")

document.write("Get the current weekday:")
document.write("<br />")
document.write(WeekdayName(weekday(Date)))
document.write("<br />")
document.write(WeekdayName(weekday(Date), True))
document.write("</p>")
</script>

</body>
</html>
```

Your Result:

VBScripts' function **WeekdayName** is used to get a weekday:

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

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

<p>VBScripts' function <b>MonthName</b> is used to get a month:</p>
<script type="text/vbscript">
document.write("<p>")
document.write(MonthName(1))
document.write("<br />")
document.write(MonthName(2))
document.write("</p><p>")

document.write("Here is how you get the abbreviated name of a month:")
document.write("<br />")
document.write(MonthName(1,True))
document.write("<br />")
document.write(MonthName(2,True))
document.write("</p><p>")

document.write("Here is how you get the current month:")
document.write("<br />")
document.write(MonthName(Month(Date)))
document.write("<br />")
document.write(MonthName(Month(Date),True))
document.write("</p>")
</script>

</body>
</html>
```

Your Result:

VBScripts' function **MonthName** is used to get a month:

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
document.write("Today's day is " & WeekdayName
(Weekday(Date)))
document.write("<br />")
document.write("The month is " & MonthName(Month
(Date)))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<p>Countdown to year 3000:</p>

<p>
<script type="text/vbscript">
millennium=CDate("1/1/3000 00:00:00")
document.write("It is " & DateDiff("yyyy", Now(), millennium) & " years to year 3000!
<br />")
document.write("It is " & DateDiff("m", Now(), millennium) & " months to year 3000!
<br />")
document.write("It is " & DateDiff("d", Now(), millennium) & " days to year 3000!<br />")
document.write("It is " & DateDiff("h", Now(), millennium) & " hours to year 3000!<br />")
document.write("It is " & DateDiff("n", Now(), millennium) & " minutes to year 3000!
<br />")
document.write("It is " & DateDiff("s", Now(), millennium) & " seconds to year 3000!
<br />")
</script>
</p>

</body>
</html>
```

Countdown to year 3000:

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
document.write(DateAdd("d",30,Date()))
</script>

<p>
This example uses <b>DateAdd</b> to calculate a date 30 days from today.
</p>
<p>
Syntax for DateAdd: DateAdd(interval,number,date).
</p>

</body>
</html>
```

This example uses **DateAdd** to calculate a date 30 days from today.

Syntax for DateAdd: DateAdd(interval,number,date).

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
document.write(FormatDateTime(Date(),vbGeneralDate))
document.write("<br />")
document.write(FormatDateTime(Date(),vbLongDate))
document.write("<br />")
document.write(FormatDateTime(Date(),vbShortDate))
document.write("<br />")
document.write(FormatDateTime(Now(),vbLongTime))
document.write("<br />")
document.write(FormatDateTime(Now(),vbShortTime))
</script>

<p>Syntax for FormatDateTime: FormatDateTime(Date,namedformat).</p>

</body>
</html>
```

Syntax for FormatDateTime: FormatDateTime(Date,namedformat).

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
somedate="10/30/99"
document.write(IsDate(somedate))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
txt="Have a nice day!"
document.write(UCase(txt))
document.write("<br />")
document.write(LCase(txt))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
fname=" Bill "
document.write("Hello" & Trim(fname) & "Gates<br />")
document.write("Hello" & RTrim(fname) &
"Gates<br />")
document.write("Hello" & LTrim(fname) &
"Gates<br />")
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
sometext = "Hello Everyone!"
document.write(StrReverse(sometext))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
i = 48.66776677
j = 48.3333333
document.write(Round(i))
document.write("<br />")
document.write(Round(j))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
Randomize()
document.write(Rnd())
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
Randomize()
randomNumber=Int(100 * Rnd())
document.write("A random number: <b>" &
randomNumber & "</b>")
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
sometext="Welcome to our Web Site!!"
document.write(Left(sometext,5))
document.write("<br />")
document.write(Right(sometext,5))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
sometext="Welcome to this Web!!"
document.write(Replace(sometext, "Web", "Page"))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

Edit and Click Me »

Your Result:

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

<script type="text/vbscript">
sometext="Welcome to our Web Site!!"
document.write(Mid(sometext, 9, 2))
</script>

</body>
</html>
```

Edit the code above and click to see the result.

W3Schools.com - Try it yourself

VBScript Functions

This page contains all the built-in VBScript functions. The page is divided into following sections:

Date/Time Functions

Function	Description
CDate	Converts a valid date and time expression to the variant of subtype Date
Date	Returns the current system date
DateAdd	Returns a date to which a specified time interval has been added
DateDiff	Returns the number of intervals between two dates
DatePart	Returns the specified part of a given date
DateSerial	Returns the date for a specified year, month, and day
DateValue	Returns a date
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
FormatDateTime	Returns an expression formatted as a date or time
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
MonthName	Returns the name of a specified month
Now	Returns the current system date and time
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
Time	Returns the current system time
Timer	Returns the number of seconds since 12:00 AM
TimeSerial	Returns the time for a specific hour, minute, and second
TimeValue	Returns a time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)
WeekdayName	Returns the weekday name of a specified day of the week
Year	Returns a number that represents the year

Conversion Functions

Top

Function	Description
Asc	Converts the first letter in a string to ANSI code
CBool	Converts an expression to a variant of subtype Boolean
CByte	Converts an expression to a variant of subtype Byte
CCur	Converts an expression to a variant of subtype Currency
CDate	Converts a valid date and time expression to the variant of subtype Date
CDbl	Converts an expression to a variant of subtype Double
Chr	Converts the specified ANSI code to a character
CInt	Converts an expression to a variant of subtype Integer
CLng	Converts an expression to a variant of subtype Long
CSng	Converts an expression to a variant of subtype Single
CStr	Converts an expression to a variant of subtype String
Hex	Returns the hexadecimal value of a specified number
Oct	Returns the octal value of a specified number

Format Functions

Top

Function	Description
FormatCurrency	Returns an expression formatted as a currency value

FormatDateTime	Returns an expression formatted as a date or time
FormatNumber	Returns an expression formatted as a number
FormatPercent	Returns an expression formatted as a percentage

Math Functions

Top

Function	Description
Abs	Returns the absolute value of a specified number
Atn	Returns the arctangent of a specified number
Cos	Returns the cosine of a specified number (angle)
Exp	Returns *e* raised to a power
Hex	Returns the hexadecimal value of a specified number
Int	Returns the integer part of a specified number
Fix	Returns the integer part of a specified number
Log	Returns the natural logarithm of a specified number
Oct	Returns the octal value of a specified number
Rnd	Returns a random number less than 1 but greater or equal to 0
Sgn	Returns an integer that indicates the sign of a specified number
Sin	Returns the sine of a specified number (angle)
Sqr	Returns the square root of a specified number
Tan	Returns the tangent of a specified number (angle)

Array Functions

Top

Function	Description
Array	Returns a variant containing an array
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria
IsArray	Returns a Boolean value that indicates whether a specified variable is an array
Join	Returns a string that consists of a number of substrings in an array
LBound	Returns the smallest subscript for the indicated dimension of an array
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings
UBound	Returns the largest subscript for the indicated dimension of an array

String Functions

Top

Function	Description
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string
LCase	Converts a specified string to lowercase
Left	Returns a specified number of characters from the left side of a string
Len	Returns the number of characters in a string
LTrim	Removes spaces on the left side of a string
RTrim	Removes spaces on the right side of a string
Trim	Removes spaces on both the left and the right side of a string
Mid	Returns a specified number of characters from a string
Replace	Replaces a specified part of a string with another string a specified number of times
Right	Returns a specified number of characters from the right side of a string
Space	Returns a string that consists of a specified number of spaces
StrComp	Compares two strings and returns a value that represents the result of the comparison
String	Returns a string that contains a repeating character of a specified length
StrReverse	Reverses a string
UCase	Converts a specified string to uppercase

Other Functions

Top

Function	Description
CreateObject	Creates an object of a specified type
Eval	Evaluates an expression and returns the result
GetLocale	Returns the current locale ID
GetObject	Returns a reference to an automation object from a file
GetRef	Allows you to connect a VBScript procedure to a DHTML event on your pages
InputBox	Displays a dialog box, where the user can write some input and/or click on a button, and returns the contents
IsEmpty	Returns a Boolean value that indicates whether a specified variable has been initialized or not
IsNull	Returns a Boolean value that indicates whether a specified expression contains no valid data (Null)
IsNumeric	Returns a Boolean value that indicates whether a specified expression can be evaluated as a number
IsObject	Returns a Boolean value that indicates whether the specified expression is an automation object
LoadPicture	Returns a picture object. Available only on 32-bit platforms
MsgBox	Displays a message box, waits for the user to click a button, and returns a value that indicates which button the user clicked
RGB	Returns a number that represents an RGB color value
Round	Rounds a number
ScriptEngine	Returns the scripting language in use
ScriptEngineBuildVersion	Returns the build version number of the scripting engine in use
ScriptEngineMajorVersion	Returns the major version number of the scripting engine in use
ScriptEngineMinorVersion	Returns the minor version number of the scripting engine in use
SetLocale	Sets the locale ID and returns the previous locale ID
TypeName	Returns the subtype of a specified variable
VarType	Returns a value that indicates the subtype of a specified variable

« Previous Next Chapter »

VBScript CDate Function

Complete VBScript Reference

The CDate function converts a valid date and time expression to type Date, and returns the result.

Tip: Use the IsDate function to determine if date can be converted to a date or time.

Note: The IsDate function uses local setting to determine if a string can be converted to a date ("January" is not a month in all languages.)

Syntax

```
CDate(date)
```

Parameter	Description
date	Required. Any valid date expression (like Date() or Now())

Examples

Example 1

How to convert a string to a date:

```
<script type="text/vbscript">
d=CDate("April 22, 2010")
</script>
```

Try it yourself »

Example 2

How to convert numbers with separators to a date:

```
<script type="text/vbscript">
d=CDate(#4/22/10#)
</script>
```

Try it yourself »

Example 3

How to use CDate to convert a string to a time object:

```
<script type="text/vbscript">
d=CDate("3:18:40 AM")
</script>
```

Try it yourself »

Complete VBScript Reference

VBScript Date Function

Complete VBScript Reference

The Date function returns the current system date.

Syntax

```
Date
```

Example

Example (IE Only)

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

document.write("The current system date is: ")
document.write(Date)

</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript DateAdd Function

Complete VBScript Reference

The DateAdd function returns a date to which a specified time interval has been added.

Syntax

```
DateAdd(interval,number,date)
```

Parameter	Description
interval	Required. The interval you want to add Can take the following values: • yyyy - Year • q - Quarter • m - Month • y - Day of year • d - Day • w - Weekday • ww - Week of year • h - Hour • n - Minute • s - Second
number	Required. The number of interval you want to add. Can either be positive, for dates in the future, or negative, for dates in the past
date	Required. Variant or literal representing the date to which interval is added

Examples

Example 1

How to use the parameters:

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

document.write(DateAdd("yyyy",1,"31-Jan-10") & "<br />")
document.write(DateAdd("q",1,"31-Jan-10") & "<br />")
document.write(DateAdd("m",1,"31-Jan-10") & "<br />")
document.write(DateAdd("y",1,"31-Jan-10") & "<br />")
document.write(DateAdd("d",1,"31-Jan-10") & "<br />")
document.write(DateAdd("w",1,"31-Jan-10") & "<br />")
document.write(DateAdd("ww",1,"31-Jan-10") & "<br />")
document.write(DateAdd("h",1,"31-Jan-10 08:50:00") & "<br />")
document.write(DateAdd("n",1,"31-Jan-10 08:50:00") & "<br />")
document.write(DateAdd("s",1,"31-Jan-10 08:50:00") & "<br />")

</script>
```

The output of the code above will be:

```
1/31/2011
4/30/2010
2/28/2010
2/1/2010
2/1/2010
2/1/2010
2/7/2010
1/31/2010 9:50:00 AM
1/31/2010 8:51:00 AM
1/31/2010 8:50:01 AM
```

Try it yourself »

Example 2

Subtract one month from January 31, 2010

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

document.write(DateAdd("m",-1,"31-Jan-10"))

</script>
```

The output of the code above will be:

```
12/31/2009
```

Try it yourself »

Example 3

Add one day from now:

```
<script type="text/vbscript">
document.write(DateAdd("d",1,Now()))
</script>
```

The output of the code above will be:

Try it yourself »


Complete VBScript Reference

VBScript DateDiff Function

Complete VBScript Reference

The DateDiff function returns the number of intervals between two dates.

Syntax

```
DateDiff(interval,date1,date2[,firstdayofweek[,firstweekofyear]])
```

Parameter	Description
interval	Required. The interval you want to use to calculate the differences between date1 and date2 Can take the following values: • yyyy - Year • q - Quarter • m - Month • y - Day of year • d - Day • w - Weekday • ww - Week of year • h - Hour • n - Minute • s - Second
date1,date2	Required. Date expressions. Two dates you want to use in the calculation
firstdayofweek	Optional. Specifies the day of the week. Can take the following values: • 0 = vbUseSystemDayOfWeek - Use National Language Support (NLS) API setting • 1 = vbSunday - Sunday (default) • 2 = vbMonday - Monday • 3 = vbTuesday - Tuesday • 4 = vbWednesday - Wednesday • 5 = vbThursday - Thursday • 6 = vbFriday - Friday • 7 = vbSaturday - Saturday
firstweekofyear	Optional. Specifies the first week of the year. Can take the following values: • 0 = vbUseSystem - Use National Language Support (NLS) API setting • 1 = vbFirstJan1 - Start with the week in which January 1 occurs (default) • 2 = vbFirstFourDays - Start with the week that has at least four days in the new year • 3 = vbFirstFullWeek - Start with the first full week of the new year

Examples

Example 1

The difference between January 31 2009, and January 31 2010:

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

fromDate="31-Jan-09 00:00:00"
toDate="31-Jan-10 23:59:00"
document.write(DateDiff("yyyy",fromDate,toDate) & "<br />")
document.write(DateDiff("q",fromDate,toDate) & "<br />")
document.write(DateDiff("m",fromDate,toDate) & "<br />")
document.write(DateDiff("y",fromDate,toDate) & "<br />")
document.write(DateDiff("d",fromDate,toDate) & "<br />")
document.write(DateDiff("w",fromDate,toDate) & "<br />")
document.write(DateDiff("ww",fromDate,toDate) & "<br />")
document.write(DateDiff("h",fromDate,toDate) & "<br />")
document.write(DateDiff("n",fromDate,toDate) & "<br />")
document.write(DateDiff("s",fromDate,toDate) & "<br />")

</script>
```

The output of the code above will be:

```
1
4
12
365
365
52
53
8783
527039
31622340
```

Try it yourself »

Example 2

How many weeks (start on Monday),
between December 31 2009 and December 31 2012:

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

fromDate=CDate("2009/12/31")
toDate=CDate("2012/12/31")
document.write(DateDiff("w",fromDate,toDate,vbMonday))

</script>
```

The output of the code above will be:

```
156
```

Try it yourself »

Complete VBScript Reference

VBScript DatePart Function

Complete VBScript Reference

The DatePart function returns the specified part of a given date.

Syntax

```
DatePart(interval,date[,firstdayofweek[,firstweekofyear]])
```

Parameter	Description
interval	Required. The interval of time to return. Can take the following values: • yyyy - Year • q - Quarter • m - Month • y - Day of year • d - Day • w - Weekday • ww - Week of year • h - Hour • n - Minute • s - Second
date	Required. Date expression to evaluate
firstdayofweek	Optional. Specifies the day of the week. Can take the following values: • 0 = vbUseSystemDayOfWeek - Use National Language Support (NLS) API setting • 1 = vbSunday - Sunday (default) • 2 = vbMonday - Monday • 3 = vbTuesday - Tuesday • 4 = vbWednesday - Wednesday • 5 = vbThursday - Thursday • 6 = vbFriday - Friday • 7 = vbSaturday - Saturday
firstweekofyear	Optional. Specifies the first week of the year. Can take the following values: • 0 = vbUseSystem - Use National Language Support (NLS) API setting • 1 = vbFirstJan1 - Start with the week in which January 1 occurs (default) • 2 = vbFirstFourDays - Start with the week that has at least four days in the new year • 3 = vbFirstFullWeek - Start with the first full week of the new year

Examples

Example 1

Get the month from a date:

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

d=CDate("2010-02-16")
document.write(DatePart("m",d))

</script>
```

The output of the code above will be:

```
2
```

Try it yourself »

Example 2

Get the month we are in:

```
<script type="text/vbscript">
document.write(DatePart("m",Now()))
</script>
```

The output of the code above will be:

Try it yourself »

Example 3

Get the hour:

```
<script type="text/vbscript">
document.write(DatePart("h",Now())
</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript The DateSerial Function

Complete VBScript Reference

The DateSerial function returns a Variant of subtype Date for a specified year, month, and day.

Syntax

```
DateSerial(year,month,day)
```

Parameter	Description
year	Required. A number between 100 and 9999, or a numeric expression. Values between 0 and 99 are interpreted as the years 1900–1999. For all other year arguments, use a complete four-digit year
month	Required. Any numeric expression
day	Required. Any numeric expression

Examples

Example 1

```
<script type="text/vbscript">
document.write(DateSerial(2010,2,3))
</script>
```

The output of the code above will be:

```
2/3/2010
```

Try it yourself »

Example 2

Subtract 10 days:

```
<script type="text/vbscript">
document.write(DateSerial(2010,2,3-10))
</script>
```

The output of the code above will be:

```
1/24/2010
```

Try it yourself »

Complete VBScript Reference

VBScript DateValue Function

Complete VBScript Reference

The DateValue function converts a string to a Date.

Note: If the year part of date is omitted this function will use the current year from the computer's system date.

Note: If the date parameter includes time information it will not be returned. However, if date includes invalid time information, a run-time error will occur.

Syntax

```
DateValue(date)
```

Parameter	Description
date	Required. A date from January 1, 100 through December 31, 9999 or any expression that can represent a date, a time, or both a date and time

Example

Example

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

document.write(DateValue("31-Jan-10"))
document.write("<br />")
document.write(DateValue("31-Jan"))

</script>
```

The output of the code above will be:

```
1/31/2010
1/31/2010
```

Try it yourself »

Complete VBScript Reference

VBScript Day Function

Complete VBScript Reference

The Day function returns a number between 1 and 31 that represents the day of the month.

Syntax

```
Day(date)
```

Parameter	Description
date	Required. Any expression that can represent a date

Example

Example

```
<script type="text/vbscript">
document.write(Day("2010-02-16"))
</script>
```

The output of the code above will be:

```
16
```

Try it yourself »

Complete VBScript Reference

VBScript FormatDateTime Function

Complete VBScript Reference

The FormatDateTime function formats and returns a valid date or time expression.

Syntax

```
FormatDateTime(date,format)
```

Parameter	Description
date	Required. Any valid date expression (like Date() or Now())
format	Optional. A value that specifies the date/time format to use Can take the following values: • 0 = vbGeneralDate - Default. Returns date: mm/dd/yy and time if specified: hh:mm:ss PM/AM. • 1 = vbLongDate - Returns date: weekday, monthname, year • 2 = vbShortDate - Returns date: mm/dd/yy • 3 = vbLongTime - Returns time: hh:mm:ss PM/AM • 4 = vbShortTime - Return time: hh:mm

Example

Example

Display a date in different formats:

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

d=CDate("2010-02-16 13:45")
document.write(FormatDateTime(d) & "<br />")
document.write(FormatDateTime(d,1) & "<br />")
document.write(FormatDateTime(d,2) & "<br />")
document.write(FormatDateTime(d,3) & "<br />")
document.write(FormatDateTime(d,4) & "<br />")

</script>
```

The output of the code above will be:

```
2/16/2010 1:45:00 PM
Tuesday, February 16, 2010
2/16/2010
1:45:00 PM
13:45
```

Try it yourself »

Complete VBScript Reference

VBScript Hour Function

Complete VBScript Reference

The Hour function returns a number between 0 and 23 that represents the hour of the day.

Syntax

```
Hour(time)
```

Parameter	Description
time	Required. Any expression that can represent a time

Examples

Example 1

```
<script type="text/vbscript">
document.write(Hour("13:45"))
</script>
```

The output of the code above will be:

```
13
```

Try it yourself »

Example 2

```
<script type="text/vbscript">
document.write(Hour(Now()))
</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript IsDate Function

Complete VBScript Reference

The IsDate function returns a Boolean value that indicates if the evaluated expression can be converted to a date. It returns True if the expression is a date or can be converted to a date; otherwise, it returns False.

Note: The IsDate function uses local setting to determine if a string can be converted to a date ("January" is not a month in all languages.)

Syntax

```
IsDate(expression)
```

Parameter	Description
expression	Required. The expression to be evaluated

Example 1

Legal dates:

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

document.write(IsDate("April 22, 1947"))
document.write("<br />")
document.write(IsDate(#01/31/10#))

</script>
```

The output of the code above will be:

```
True
True
```

Try it yourself »

Example 2

Illegal dates:

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

document.write(IsDate("#01/31/10#"))
document.write("<br />")
document.write(IsDate("52/17/2010"))
document.write("<br />")
document.write(IsDate("Hello World!"))

</script>
```

The output of the code above will be:

```
False
False
False
```

Try it yourself »

Complete VBScript Reference

VBScript Minute Function

Complete VBScript Reference

The Minute function returns a number between 0 and 59 that represents the minute of the hour.

Syntax

```
Minute(time)
```

Parameter	Description
time	Required. Any expression that can represent a time

Examples

Example 1

```
<script type="text/vbscript">
document.write(Minute("13:45"))
</script>
```

The output of the code above will be:

```
45
```

Try it yourself »

Example 2

```
<script type="text/vbscript">
document.write(Minute(Now()))
</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript Month Function

Complete VBScript Reference

The Month function returns a number between 1 and 12 that represents the month of the year.

Syntax

```
Month(date)
```

Parameter	Description
date	Required. Any expression that can represent a date

Example

Example

```
<script type="text/vbscript">
document.write(Month("2010-02-16"))
</script>
```

The output of the code above will be:

```
2
```

Try it yourself »

Complete VBScript Reference

VBScript MonthName Function

Complete VBScript Reference

The MonthName function returns the name of the specified month.

Syntax

```
MonthName(month[,abbreviate])
```

Parameter	Description
month	Required. Specifies the number of the month (January is 1, February is 2, etc.)
abbreviate	Optional. A Boolean value that indicates if the month name is to be abbreviated. Default is False

Examples

Example 1

Get the name of the 8th month:

```
<script type="text/vbscript">
document.write(MonthName(8))
</script>
```

The output of the code above will be:

```
August
```

Try it yourself »

Example 2

Get the short name of the 8th month:

```
<script type="text/vbscript">
document.write(MonthName(8,True))
</script>
```

The output of the code above will be:

```
Aug
```

Try it yourself »

Complete VBScript Reference

VBScript Now Function

Complete VBScript Reference

The Now function returns the current date and time according to the setting of your computer's system date and time.

Syntax

```
Now
```

Example

Example (IE Only)

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

document.write("The current system date and time is: ")
document.write(Now)

</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript Second Function

Complete VBScript Reference

The Second function returns a number between 0 and 59 that represents the second of the minute.

Syntax

```
Second(time)
```

Parameter	Description
time	Required. Any expression that can represent a time

Examples

Example 1

```
<script type="text/vbscript">
document.write(Second("13:45:21"))
</script>
```

The output of the code above will be:

```
21
```

Try it yourself »

Example 2

```
<script type="text/vbscript">
document.write(Second(Now()))
</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript Time Function

Complete VBScript Reference

The Time function returns the current system time.

Syntax

```
Time
```

Example

Example (IE Only)

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

document.write("The current system time is: ")
document.write(Time)

</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript Timer Function

Complete VBScript Reference

The Timer function returns the number of seconds, and milliseconds, since 12:00 AM.

Syntax

```
Timer
```

Example

Example (IE Only)

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

document.write("Number of seconds and milliseconds since 12:00 AM: ")
document.write(Timer)

</script>
```

The output of the code above will be:

Try it yourself »

Complete VBScript Reference

VBScript TimeSerial Function

Complete VBScript Reference

The TimeSerial function returns the time for a specific hour, minute, and second.

Syntax

```
TimeSerial(hour,minute,second)
```

Parameter	Description
hour	Required. A number between 0 and 23, or a numeric expression
minute	Required. Any numeric expression
second	Required. Any numeric expression

Example

Example

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

document.write(TimeSerial(23,2,3) & "<br />")
document.write(TimeSerial(0,9,11) & "<br />")
document.write(TimeSerial(14+2,9-2,1-1))

</script>
```

The output of the code above will be:

```
11:02:03 PM
12:09:11 AM
4:07:00 PM
```

Try it yourself »

Complete VBScript Reference

VBScript TimeValue Function

Complete VBScript Reference

The TimeValue function returns a Variant of subtype Date that contains the time.

Syntax

```
TimeValue(time)
```

Parameter	Description
time	Required. A time from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.) or any expression that represents a time in that range

Example

Example

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

document.write(TimeValue("5:55:59 PM") & "<br />")
document.write(TimeValue(#5:55:59 PM#) & "<br />")
document.write(TimeValue("15:34"))

</script>
```

The output of the code above will be:

```
5:55:59 PM
5:55:59 PM
3:34:00 PM
```

Try it yourself »

Complete VBScript Reference

VBScript Weekday Function

Complete VBScript Reference

The Weekday function returns a number between 1 and 7, that represents the day of the week.

Syntax

```
Weekday(date[,firstdayofweek])
```

Parameter	Description
date	Required. The date expression to evaluate
firstdayofweek	Optional. Specifies the first day of the week. Can take the following values: • 0 = vbUseSystemDayOfWeek - Use National Language Support (NLS) API setting • 1 = vbSunday - Sunday (default) • 2 = vbMonday - Monday • 3 = vbTuesday - Tuesday • 4 = vbWednesday - Wednesday • 5 = vbThursday - Thursday • 6 = vbFriday - Friday • 7 = vbSaturday - Saturday

Example

Example

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

document.write(Weekday("2010-02-16",1) & "<br />")
document.write(Weekday("2010-02-16",2) & "<br />")
document.write(Weekday("2010-02-16",3) & "<br />")
document.write(Weekday("2010-02-16",4) & "<br />")
document.write(Weekday("2010-02-16",5) & "<br />")
document.write(Weekday("2010-02-16",6) & "<br />")

</script>
```

The output of the code above will be:

```
3
2
1
7
6
5
```

Try it yourself »

Complete VBScript Reference

VBScript WeekdayName Function

Complete VBScript Reference

The WeekdayName function returns the weekday name of a specified day of the week.

Syntax

```
WeekdayName(weekday[,abbreviate[,firstdayofweek]])
```

Parameter	Description
weekday	Required. The number of the weekday
abbreviate	Optional. A Boolean value that indicates if the weekday name is to be abbreviated
firstdayofweek	Optional. Specifies the first day of the week. Can take the following values: • 0 = vbUseSystemDayOfWeek - Use National Language Support (NLS) API setting • 1 = vbSunday - Sunday (default) • 2 = vbMonday - Monday • 3 = vbTuesday - Tuesday • 4 = vbWednesday - Wednesday • 5 = vbThursday - Thursday • 6 = vbFriday - Friday • 7 = vbSaturday - Saturday

Examples

Example 1

Get the name of the 3rd day of the week:

```
<script type="text/vbscript">
document.write(WeekdayName(3))
</script>
```

The output of the code above will be:

```
Tuesday
```

Try it yourself »

Example 2

Get the short name of the 3rd day of the week:

```
<script type="text/vbscript">
document.write(WeekdayName(3,True))
</script>
```

The output of the code above will be:

```
Tue
```

Try it yourself »

Example 3

Get the name of the 3rd day of the week, where the first day is monday:

```
<script type="text/vbscript">
document.write(WeekdayName(3,False,2))
</script>
```

The output of the code above will be:

```
Wednesday
```

Try it yourself »


Complete VBScript Reference

VBScript Year Function

Complete VBScript Reference

The Year function returns a number that represents the year.

Syntax

```
Year(date)
```

Parameter	Description
date	Required. Any expression that can represent a date

Example

Example

```
<script type="text/vbscript">
document.write(Year("2010-02-16"))
</script>
```

The output of the code above will be:

```
2010
```

Try it yourself »

Complete VBScript Reference

VBScript Asc Function

Complete VBScript Reference

The Asc function converts the first character in a string to ANSI code, and returns the result.

Syntax

```
Asc(string)
```

Parameter	Description
string	Required. A string expression. Cannot be an empty string!

Examples

Example 1

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

document.write(Asc("A") & "<br />")
document.write(Asc("a") & "<br />")
document.write(Asc("F") & "<br />")
document.write(Asc("f") & "<br />")
document.write(Asc("2") & "<br />")
document.write(Asc("#") & "<br />")

</script>
```

The output of the code above will be:

```
65
97
70
102
50
35
```

Try it yourself »

Example 2

Asc returns the ANSI code of only the first character in a string:

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

document.write(Asc("W") & "<br />")
document.write(Asc("W3Schools.com"))

</script>
```

The output of the code above will be:

```
87
87
```

Try it yourself »

Example 3

How to return the ANSI code of all the characters in a string:

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

txt="W3schools.com"
for i=1 to len(txt)
   document.write(Asc(mid(txt,i,1)) & "<br />")
next

</script>
```

The output of the code above will be:

```
87
51
115
```

```
99
104
111
111
108
115
46
99
111
109
```

Try it yourself »

Complete VBScript Reference

VBScript CBool Function

Complete VBScript Reference

The CBool function converts an expression to type Boolean.

The expression must be a numeric value.

Syntax

```
CBool(expression)
```

Parameter	Description
expression	Required. Any valid expression. A nonzero value returns True, zero returns False. A run-time error occurs if the expression can not be interpreted as a numeric value

Example

Example

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

document.write(CBool(5) & "<br />")
document.write(CBool(0) & "<br />")
document.write(CBool(-5) & "<br />")

</script>
```

The output of the code above will be:

```
True
False
True
```

Try it yourself »

Complete VBScript Reference

VBScript CByte Function

Complete VBScript Reference

The CByte function converts an expression to type Byte.

The expression must be a number between 0 and 255

Syntax

```
CByte(expression)
```

Parameter	Description
expression	Required. Any valid expression

Example

Example

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

document.write(CByte(0) & "<br />")
document.write(CByte(56.8) & "<br />")
document.write(CByte(123.2) & "<br />")
document.write(CByte(255) & "<br />")

</script>
```

The output of the code above will be:

```
0
57
123
255
```

Try it yourself »

Complete VBScript Reference

VBScript CCur Function

Complete VBScript Reference

The CCur function converts an expression to type Currency.

The expression must be a numeric value.

Syntax

```
CCur(expression)
```

Parameter	Description
expression	Required. Any valid expression

Examples

Example 1

```
<script type="text/vbscript">
document.write(CCur(30) & "<br />")
</script>
```

The output of the code above will be:

```
30
```

Try it yourself »

Example 2

Note that CCur rounds off to 4 decimals:

```
<script type="text/vbscript">
document.write(CCur(5.9555555555555) & "<br />")
</script>
```

The output of the code above will be:

```
5.9556
```

Try it yourself »

Complete VBScript Reference

VBScript CDate Function

Complete VBScript Reference

The CDate function converts a valid date and time expression to type Date, and returns the result.

Tip: Use the IsDate function to determine if date can be converted to a date or time.

Note: The IsDate function uses local setting to determine if a string can be converted to a date ("January" is not a month in all languages.)

Syntax

```
CDate(date)
```

Parameter	Description
date	Required. Any valid date expression (like Date() or Now())

Examples

Example 1

How to convert a string to a date:

```
<script type="text/vbscript">
d=CDate("April 22, 2010")
</script>
```

Try it yourself »

Example 2

How to convert numbers with separators to a date:

```
<script type="text/vbscript">
d=CDate(#4/22/10#)
</script>
```

Try it yourself »

Example 3

How to use CDate to convert a string to a time object:

```
<script type="text/vbscript">
d=CDate("3:18:40 AM")
</script>
```

Try it yourself »

Complete VBScript Reference

VBScript CDbl Function

Complete VBScript Reference

The CDbl function converts an expression to type Double.

The expression must be a numeric value.

Syntax

```
CDbl(expression)
```

Parameter	Description
expression	Required. Any valid expression

Examples

Example 1

```
<script type="text/vbscript">
document.write(CDbl(134.345) & "<br />")
</script>
```

The output of the code above will be:

```
134.345
```

Try it yourself »

Example 2

```
<script type="text/vbscript">
document.write(CDbl(14111111113353355.345455) & "<br />")
</script>
```

The output of the code above will be:

```
1.41111111133534E+16
```

Try it yourself »

Complete VBScript Reference

VBScript Chr Function

Complete VBScript Reference

The Chr function converts the specified ANSI character code to a character.

Note: The numbers from 0 to 31 represents nonprintable ASCII codes, i.e. Chr(10) will return a linefeed character.

Syntax

```
Chr(charcode)
```

Parameter	Description
charcode	Required. A number that identifies a character

Examples

Example 1

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

document.write(Chr(65) & "<br />")
document.write(Chr(66) & "<br />")
document.write(Chr(67) & "<br />")
document.write(Chr(97) & "<br />")
document.write(Chr(98) & "<br />")
document.write(Chr(99) & "<br />")

</script>
```

The output of the code above will be:

```
A
B
C
a
b
c
```

Try it yourself »

Example 2

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

document.write(Chr(34) & "<br />")
document.write(Chr(35) & "<br />")
document.write(Chr(36) & "<br />")
document.write(Chr(37) & "<br />")

</script>
```

The output of the code above will be:

```
"
#
$
%
```

Try it yourself »

Complete VBScript Reference

VBScript CInt Function

Complete VBScript Reference

The CInt function converts an expression to type Integer.

Note: The value must be a number between -32768 and 32767.

Syntax

```
CInt(expression)
```

Parameter	Description
expression	Required. Any valid expression

Example

Example

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

document.write(CInt("300") & "<br />")
document.write(CInt(36.75) & "<br />")
document.write(CInt(-67) & "<br />")

</script>
```

The output of the code above will be:

```
300
37
-67
```

Try it yourself »

Complete VBScript Reference

VBScript CLng Function

Complete VBScript Reference

The CLng function converts an expression to type Long.

Note: The value must be a number between -2147483648 and 2147483647.

Syntax

```
CLng(expression)
```

Parameter	Description
expression	Required. Any valid expression

Example

Example

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

document.write(CLng("300000") & "<br />")
document.write(CLng(1536.750) & "<br />")
document.write(CLng(-6700000) & "<br />")

</script>
```

The output of the code above will be:

```
300000
1537
-6700000
```

Try it yourself »

Complete VBScript Reference

VBScript CSng Function

Complete VBScript Reference

The CSng function converts an expression to type Single.

The expression must be a numeric value.

Syntax

```
CSng(expression)
```

Parameter	Description
expression	Required. Any valid expression

Example

Example

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

document.write(CSng("300000") & "<br />")
document.write(CSng(1536.75263541) & "<br />")
document.write(CSng(-6700000) & "<br />")

</script>
```

The output of the code above will be:

```
300000
1536.753
-6700000
```

Try it yourself »

Complete VBScript Reference

VBScript CStr Function

Complete VBScript Reference

The CStr function converts an expression to type String.

Syntax

```
CStr(expression)
```

Parameter	Description
expression	Required. Any valid expression If expression is: • Boolean - then the CStr function will return a string containing true or false. • Date - then the CStr function will return a string that contains a date in the short-date format. • Null - then a run-time error will occur. • Empty - then the CStr function will return an empty string (""). • Error - then the CStr function will return a string that contains the word "Error" followed by an error number. • Other numeric - then the CStr function will return a string that contains the number.

Example

Example

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

document.write(CStr("300000") & "<br />")
document.write(CStr(#10-05-25#) & "<br />")

</script>
```

The output of the code above will be:

```
300000
10/5/2025
```

Try it yourself »

Complete VBScript Reference

VBScript Hex Function

Complete VBScript Reference

The Hex function returns a string that represents the hexadecimal value of a specified number.

Note: If number is not a whole number, it is rounded to the nearest whole number before being evaluated.

Syntax

```
Hex(number)
```

Parameter	Description
number	Required. Any valid expression If number is: • Null - then the Hex function returns Null. • Empty - then the Hex function returns zero (0). • Any other number - then the Hex function returns up to eight hexadecimal characters.

Example

Example

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

document.write(Hex(3) & "<br />")
document.write(Hex(5) & "<br />")
document.write(Hex(9) & "<br />")
document.write(Hex(10) & "<br />")
document.write(Hex(11) & "<br />")
document.write(Hex(12) & "<br />")
document.write(Hex(400) & "<br />")
document.write(Hex(459) & "<br />")
document.write(Hex(460) & "<br />")

</script>
```

The output of the code above will be:

```
3
5
9
A
B
C
190
1CB
1CC
```

Try it yourself »

Complete VBScript Reference

VBScript Oct Function

Complete VBScript Reference

The Oct function returns a string that represents the octal value of a specified number.

Note: If number is not already a whole number, it is rounded to the nearest whole number before being evaluated.

Syntax

```
Oct(number)
```

Parameter	Description
number	Required. Any valid expression If number is: • Null - then the Oct function returns Null. • Empty - then the Oct function returns zero (0). • Any other number - then the Oct function returns up to 11 octal characters.

Example

Example

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

document.write(Oct(3) & "<br />")
document.write(Oct(5) & "<br />")
document.write(Oct(9) & "<br />")
document.write(Oct(10) & "<br />")
document.write(Oct(11) & "<br />")
document.write(Oct(12) & "<br />")
document.write(Oct(400) & "<br />")
document.write(Oct(459) & "<br />")
document.write(Oct(460) & "<br />")

</script>
```

The output of the code above will be:

```
3
5
11
12
13
14
620
713
714
```

Try it yourself »

Complete VBScript Reference

VBScript FormatCurrency Function

Complete VBScript Reference

The FormatCurrency function returns an expression formatted as a currency value using the currency symbol defined in the computer's control panel.

Syntax

```
FormatCurrency(Expression[,NumDigAfterDec[,
IncLeadingDig[,UseParForNegNum[,GroupDig]]]])
```

Parameter	Description
expression	Required. The expression to be formatted
NumDigAfterDec	Optional. Indicates how many places to the right of the decimal are displayed. Default is -1 (the computer's regional settings are used)
IncLeadingDig	Optional. Indicates whether or not a leading zero is displayed for fractional values: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False
UseParForNegNum	Optional. Indicates whether or not to place negative values within parentheses: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False
GroupDig	Optional. Indicates whether or not numbers are grouped using the group delimiter specified in the computer's regional settings: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False

Examples

Example 1

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

document.write(FormatCurrency(20000))

</script>
```

The output of the code above will be:

```
$20,000.00
```

Try it yourself »

Example 2

Setting number of decimals:

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

document.write(FormatCurrency(20000,2) & "<br />")
document.write(FormatCurrency(20000,5))

</script>
```

The output of the code above will be:

```
$20,000.00
$20,000.00000
```

Try it yourself »

Example 3

Fractional values with or without a leading zero:

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

document.write(FormatCurrency(.20,,0) & "<br />")
document.write(FormatCurrency(.20,,-1))

</script>
```

The output of the code above will be:

```
$.20
$0.20
```

Try it yourself »

Example 4

Negative values inside parentheses or not:

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

document.write(FormatCurrency(-50,,,0) & "<br />")
document.write(FormatCurrency(-50,,,-1))

</script>
```

The output of the code above will be:

```
-$50.00
($50.00)
```

Try it yourself »

Example 5

Grouping a million dollars - or not:

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

document.write(FormatCurrency(1000000,,,,0) & "<br />")
document.write(FormatCurrency(1000000,,,,-1))

</script>
```

The output of the code above will be:

```
$1000000.00
$1,000,000.00
```

Try it yourself »

Complete VBScript Reference

VBScript FormatDateTime Function

Complete VBScript Reference

The FormatDateTime function formats and returns a valid date or time expression.

Syntax

```
FormatDateTime(date,format)
```

Parameter	Description
date	Required. Any valid date expression (like Date() or Now())
format	Optional. A value that specifies the date/time format to use Can take the following values: • 0 = vbGeneralDate - Default. Returns date: mm/dd/yy and time if specified: hh:mm:ss PM/AM. • 1 = vbLongDate - Returns date: weekday, monthname, year • 2 = vbShortDate - Returns date: mm/dd/yy • 3 = vbLongTime - Returns time: hh:mm:ss PM/AM • 4 = vbShortTime - Return time: hh:mm

Example

Example

Display a date in different formats:

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

d=CDate("2010-02-16 13:45")
document.write(FormatDateTime(d) & "<br />")
document.write(FormatDateTime(d,1) & "<br />")
document.write(FormatDateTime(d,2) & "<br />")
document.write(FormatDateTime(d,3) & "<br />")
document.write(FormatDateTime(d,4) & "<br />")

</script>
```

The output of the code above will be:

```
2/16/2010 1:45:00 PM
Tuesday, February 16, 2010
2/16/2010
1:45:00 PM
13:45
```

Try it yourself »

Complete VBScript Reference

VBScript FormatNumber Function

Complete VBScript Reference

The FormatNumber function returns an expression formatted as a number.

Syntax

```
FormatNumber(Expression[,NumDigAfterDec[,
IncLeadingDig[,UseParForNegNum[,GroupDig]]]])
```

Parameter	Description
expression	Required. The expression to be formatted
NumDigAfterDec	Optional. Indicates how many places to the right of the decimal are displayed. Default is -1 (the computer's regional settings are used)
IncLeadingDig	Optional. Indicates whether or not a leading zero is displayed for fractional values: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False
UseParForNegNum	Optional. Indicates whether or not to place negative values within parentheses: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False
GroupDig	Optional. Indicates whether or not numbers are grouped using the group delimiter specified in the computer's regional settings: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False

Examples

Example 1

```
<script type="text/vbscript">
document.write(FormatNumber(20000))
</script>
```

The output of the code above will be:

```
20,000.00
```

Try it yourself »

Example 2

Setting number of decimals:

```
<script type="text/vbscript">
document.write(FormatNumber(20000,2) & "<br />")
document.write(FormatNumber(20000,5))
</script>
```

The output of the code above will be:

```
20,000.00
20,000.00000
```

Try it yourself »

Example 3

Fractional values with or without a leading zero:

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

document.write(FormatNumber(.20,,0) & "<br />")
document.write(FormatNumber(.20,,-1))

</script>
```

The output of the code above will be:

```
.20
0.20
```

Try it yourself »

Example 4

Negative values inside parentheses or not:

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

document.write(FormatNumber(-50,,,0) & "<br />")
document.write(FormatNumber(-50,,,-1))

</script>
```

The output of the code above will be:

```
-50.00
(50.00)
```

Try it yourself »

Example 5

Grouping numbers - or not:

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

document.write(FormatNumber(1000000,,,,0) & "<br />")
document.write(FormatNumber(1000000,,,,-1))

</script>
```

The output of the code above will be:

```
1000000.00
1,000,000.00
```

Try it yourself »

Complete VBScript Reference

VBScript FormatPercent Function

Complete VBScript Reference

The FormatPercent function returns an expression formatted as a percentage (multiplied by 100) with a trailing % character.

Syntax

```
FormatPercent(Expression[,NumDigAfterDec[,
IncLeadingDig[,UseParForNegNum[,GroupDig]]]])
```

Parameter	Description
expression	Required. The expression to be formatted
NumDigAfterDec	Optional. Indicates how many places to the right of the decimal are displayed. Default is -1 (the computer's regional settings are used)
IncLeadingDig	Optional. Indicates whether or not a leading zero is displayed for fractional values: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False
UseParForNegNum	Optional. Indicates whether or not to place negative values within parentheses: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False
GroupDig	Optional. Indicates whether or not numbers are grouped using the group delimiter specified in the computer's regional settings: • -2 = TristateUseDefault - Use the computer's regional settings • -1 = TristateTrue - True • 0 = TristateFalse - False

Example 1

```
'How many percent is 6 of 345?
document.write(FormatPercent(6/345))

Output:

1.74%
```

Example 2

```
'How many percent is 6 of 345?
document.write(FormatPercent(6/345,1))

Output:

1.7%
```

Complete VBScript Reference

VBScript Abs Function

Complete VBScript Reference

The Abs function returns the absolute value of a specified number.

Note: If the number parameter contains Null, Null will be returned

Note: If the number parameter is an uninitialized variable, zero will be returned.

Syntax

```
Abs(number)
```

Parameter	Description
number	Required. A numeric expression

Examples

Example 1

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

document.write(Abs(1) & "<br />")
document.write(Abs(-1))

</script>
```

The output of the code above will be:

```
1
1
```

Try it yourself »

Example 2

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

document.write(Abs(48.4) & "<br />")
document.write(Abs(-48.4))

</script>
```

The output of the code above will be:

```
48.4
48.4
```

Try it yourself »

Complete VBScript Reference

VBScript Atn Function

Complete VBScript Reference

The Atn function returns the arctangent of a specified number.

Syntax

```
Atn(number)
```

Parameter	Description
number	Required. A numeric expression

Examples

Example 1

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

document.write(Atn(89) & "<br />")
document.write(Atn(8.9))

</script>
```

The output of the code above will be:

```
1.55956084453693
1.45890606062322
```

Try it yourself »

Example 2

Calculate the value of pi:

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

Dim pi
pi=4*Atn(1)
document.write(pi)

</script>
```

The output of the code above will be:

```
3.14159265358979
```

Try it yourself »

Complete VBScript Reference

VBScript Cos Function

Complete VBScript Reference

The Cos function returns the cosine of a specified number (angle).

Syntax

```
Cos(number)
```

Parameter	Description
number	Required. A numeric expression that expresses an angle in radians

Example

Example

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

document.write(Cos(50.0) & "<br />")
document.write(Cos(-50.0))

</script>
```

The output of the code above will be:

```
0.964966028492113
0.964966028492113
```

Try it yourself »

Complete VBScript Reference

VBScript Exp Function

Complete VBScript Reference

The Exp function returns *e* raised to a power.

Note: The value of number cannot exceed 709.782712893.

Tip: Also look at the Log function.

Syntax

```
Exp(number)
```

Parameter	Description
number	Required. A valid numeric expression

Example

Example

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

document.write(Exp(6.7) & "<br />")
document.write(Exp(-6.7))

</script>
```

The output of the code above will be:

```
812.405825167543
1.23091190267348E-03
```

Try it yourself »

Complete VBScript Reference

VBScript Hex Function

Complete VBScript Reference

The Hex function returns a string that represents the hexadecimal value of a specified number.

Note: If number is not a whole number, it is rounded to the nearest whole number before being evaluated.

Syntax

```
Hex(number)
```

Parameter	Description
number	Required. Any valid expression If number is: • Null - then the Hex function returns Null. • Empty - then the Hex function returns zero (0). • Any other number - then the Hex function returns up to eight hexadecimal characters.

Example

Example

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

document.write(Hex(3) & "<br />")
document.write(Hex(5) & "<br />")
document.write(Hex(9) & "<br />")
document.write(Hex(10) & "<br />")
document.write(Hex(11) & "<br />")
document.write(Hex(12) & "<br />")
document.write(Hex(400) & "<br />")
document.write(Hex(459) & "<br />")
document.write(Hex(460) & "<br />")

</script>
```

The output of the code above will be:

```
3
5
9
A
B
C
190
1CB
1CC
```

Try it yourself »

Complete VBScript Reference

VBScript Int Function

Complete VBScript Reference

The Int function returns the integer part of a specified number.

Note: If the number parameter contains Null, Null will be returned.

Tip: Also look at the Fix function.

Syntax

```
Int(number)
```

Parameter	Description
number	Required. A valid numeric expression

Examples

Example 1

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

document.write(Int(6.83227) & "<br />")
document.write(Int(6.23443))

</script>
```

The output of the code above will be:

```
6
6
```

Try it yourself »

Example 2

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

document.write(Int(-6.13443) & "<br />")
document.write(Int(-6.93443))

</script>
```

The output of the code above will be:

```
-7
-7
```

Try it yourself »

Complete VBScript Reference

VBScript Fix Function

Complete VBScript Reference

The Fix function returns the integer part of a specified number.

Note: If the number parameter contains Null, Null will be returned.

Tip: Also look at the Int function.

Syntax

```
Fix(number)
```

Parameter	Description
number	Required. A valid numeric expression

Examples

Example 1

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

document.write(Fix(6.83227) & "<br />")
document.write(Fix(6.23443))

</script>
```

The output of the code above will be:

```
6
6
```

Try it yourself »

Example 2

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

document.write(Fix(-6.13443) & "<br />")
document.write(Fix(-6.93443))

</script>
```

The output of the code above will be:

```
-6
-6
```

Try it yourself »

Complete VBScript Reference

VBScript Log Function

Complete VBScript Reference

The Log function returns the natural logarithm of a specified number. The natural logarithm is the logarithm to the base e.

Note: Negative values are not allowed.

Tip: Also look at the Exp function.

Syntax

```
Log(number)
```

Parameter	Description
number	Required. A valid numeric expression > 0

Example

Example

```
<script type="text/vbscript">
document.write(Log(38.256783227))
</script>
```

The output of the code above will be:

```
3.64432088381777
```

Try it yourself »

Complete VBScript Reference

VBScript Oct Function

Complete VBScript Reference

The Oct function returns a string that represents the octal value of a specified number.

Note: If number is not already a whole number, it is rounded to the nearest whole number before being evaluated.

Syntax

```
Oct(number)
```

Parameter	Description
number	Required. Any valid expression If number is: • Null - then the Oct function returns Null. • Empty - then the Oct function returns zero (0). • Any other number - then the Oct function returns up to 11 octal characters.

Example

Example

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

document.write(Oct(3) & "<br />")
document.write(Oct(5) & "<br />")
document.write(Oct(9) & "<br />")
document.write(Oct(10) & "<br />")
document.write(Oct(11) & "<br />")
document.write(Oct(12) & "<br />")
document.write(Oct(400) & "<br />")
document.write(Oct(459) & "<br />")
document.write(Oct(460) & "<br />")

</script>
```

The output of the code above will be:

```
3
5
11
12
13
14
620
713
714
```

Try it yourself »

Complete VBScript Reference

VBScript Rnd Function

Complete VBScript Reference

The Rnd function returns a random number. The number is always less than 1 but greater or equal to 0.

Syntax

```
Rnd[(number)]
```

Parameter	Description
number	Optional. A valid numeric expression If number is: • <0 - Rnd returns the same number every time • >0 - Rnd returns the next random number in the sequence • =0 - Rnd returns the most recently generated number • Not supplied - Rnd returns the next random number in the sequence

Examples

Example 1

A random number:

```
<script type="text/vbscript">
document.write(Rnd)
</script>
```

Note that you will get the same number every time. To avoid this, use the Randomize statement like in Example 2

The output of the code above will be:

```
0.7055475
```

Try it yourself »

Example 2

To avoid getting the same number every time, like in Example 1, use the Randomize statement:

```
<script type="text/vbscript">
Randomize
document.write(Rnd)
</script>
```

The output of the code above will be:

```
0.4758112
```

Try it yourself »

Example 3

Here is how to produce random integers in a given range:

```
<script type="text/vbscript">
Dim max,min
max=100
min=1
Randomize
document.write(Int((max-min+1)*Rnd+min))
</script>
```

The output of the code above will be:

```
71
```

Try it yourself »


Complete VBScript Reference

VBScript Sgn Function

Complete VBScript Reference

The Sgn function returns an integer that indicates the sign of a specified number.

Syntax

```
Sgn(number)
```

Parameter	Description
number	Required. A valid numeric expression If number is: • >0 - Sgn returns 1 • =0 - Sgn returns 0 • <0 - Sgn returns -1

Example

Example

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

document.write(Sgn(15) & "<br />")
document.write(Sgn(-5.67) & "<br />")
document.write(Sgn(0))

</script>
```

The output of the code above will be:

```
1
-1
0
```

Try it yourself »

Complete VBScript Reference

VBScript Sin Function

Complete VBScript Reference

The Sin function returns the sine of a specified number (angle).

Syntax

```
Sin(number)
```

Parameter	Description
number	Required. A valid numeric expression that expresses an angle in radians

Example

Example

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

document.write(Sin(47) & "<br />")
document.write(Sin(-47))

</script>
```

The output of the code above will be:

```
0.123573122745224
-0.123573122745224
```

Try it yourself »

Complete VBScript Reference

VBScript Sqr Function

Complete VBScript Reference

The Sqr function returns the square root of a number.

Note: The number parameter cannot be a negative value.

Syntax

```
Sqr(number)
```

Parameter	Description
number	Required. A valid numeric expression >= 0

Example

Example

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

document.write(Sqr(9) & "<br />")
document.write(Sqr(0) & "<br />")
document.write(Sqr(47))

</script>
```

The output of the code above will be:

```
3
0
6.85565460040104
```

Try it yourself »

Complete VBScript Reference

VBScript Tan Function

Complete VBScript Reference

The Tan function returns the tangent of a specified number (angle).

Syntax

```
Tan(number)
```

Parameter	Description
number	Required. A valid numeric expression that expresses an angle in radians

Example

Example

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

document.write(Tan(40) & "<br />")
document.write(Tan(-40))

</script>
```

The output of the code above will be:

```
-1.1172149309239
1.1172149309239
```

Try it yourself »

Complete VBScript Reference

VBScript Array Function

Complete VBScript Reference

The Array function returns a variant containing an array.

Note: The first element in the array is zero.

Syntax

```
Array(arglist)
```

Parameter	Description
arglist	Required. A list (separated by commas) of values that is the elements in the array

Examples

Example 1

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

a=Array(5,10,15,20)
document.write(a(3))

</script>
```

The output of the code above will be:

```
20
```

Try it yourself »

Example 2

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

a=Array(5,10,15,20)
document.write(a(0))

</script>
```

The output of the code above will be:

```
5
```

Try it yourself »

Example 3

Looping through all items in an array:

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

a=Array(5,10,15,20)
for each x in a
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
5
10
15
20
```

Try it yourself »

Complete VBScript Reference

VBScript Filter Function

Complete VBScript Reference

The Filter function returns a zero-based array that contains a subset of a string array based on a filter criteria.

Note: If no matches of the value parameter are found, the Filter function will return an empty array.

Note: If the parameter inputstrings is Null or is NOT a one-dimensional array, an error will occur.

Syntax

```
Filter(inputstrings,value[,include[,compare]])
```

Parameter	Description
inputstrings	Required. A one-dimensional array of strings to be searched
value	Required. The string to search for
include	Optional. A Boolean value that indicates whether to return the substrings that include or exclude value. True returns the subset of the array that contains value as a substring. False returns the subset of the array that does not contain value as a substring. Default is True.
compare	Optional. Specifies the string comparison to use. Can have one of the following values: • 0 = vbBinaryCompare - Perform a binary comparison • 1 = vbTextCompare - Perform a textual comparison

Examples

Example 1

Filter: items that contains "S"

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

a=Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")

b=Filter(a,"S")
for each x in b
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
Sunday
Saturday
```

Try it yourself »

Example 2

Filter: items that does NOT contain "S" (include=False):

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

a=Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")

b=Filter(a,"S",False)
for each x in b
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
Monday
Tuesday
Wednesday
Thursday
Friday
```

Try it yourself »

Example 3

Filter: items that contains "S", with a textual comparison (compare=1):

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

a=Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")

b=Filter(a,"S",True,1)
for each x in b
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
Sunday
Tuesday
Wednesday
Thursday
Saturday
```

Try it yourself »

Complete VBScript Reference

VBScript IsArray Function

Complete VBScript Reference

The IsArray function returns a Boolean value that indicates whether a specified variable is an array. If the variable is an array, it returns True, otherwise, it returns False.

Syntax

```
IsArray(variable)
```

Parameter	Description
variable	Required. Any variable

Examples

Example 1

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

days=Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat")
document.write(IsArray(days))

</script>
```

The output of the code above will be:

```
True
```

Try it yourself »

Example 2

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

Dim a(5)
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
document.write(IsArray(a))

</script>
```

The output of the code above will be:

```
True
```

Try it yourself »

Example 3

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

a="Saturday"
document.write(IsArray(a))

</script>
```

The output of the code above will be:

```
False
```

Try it yourself »

Complete VBScript Reference

VBScript Join Function

Complete VBScript Reference

The Join function returns a string that consists of a number of substrings in an array.

Syntax

```
Join(list[,delimiter])
```

Parameter	Description
list	Required. A one-dimensional array that contains the substrings to be joined
delimiter	Optional. The character(s) used to separate the substrings in the returned string. Default is the space character

Example

Example

Separating items in an array, with and without using the delimeter parameter:

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

days=Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat")
document.write(Join(days) & "<br />")
document.write(Join(days,",") & "<br />")
document.write(Join(days," ### "))

</script>
```

The output of the code above will be:

```
Sun Mon Tue Wed Thu Fri Sat
Sun,Mon,Tue,Wed,Thu,Fri,Sat
Sun ### Mon ### Tue ### Wed ### Thu ### Fri ### Sat
```

Try it yourself »

Complete VBScript Reference

VBScript LBound Function

Complete VBScript Reference

The LBound function returns the smallest subscript for the indicated dimension of an array.

Note: The LBound for any dimension is ALWAYS 0.

Tip: Use the LBound function with the UBound function to determine the size of an array.

Syntax

```
LBound(arrayname[,dimension])
```

Parameter	Description
arrayname	Required. The name of the array variable
dimension	Optional. Which dimension's lower bound to return. 1 = first dimension, 2 = second dimension, and so on. Default is 1

Examples

Example 1

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

days=Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat")
document.write(LBound(days) & "<br />")
document.write(UBound(days) & "<br />")

</script>
```

The output of the code above will be:

```
0
6
```

Try it yourself »

Example 2

A two dimensional array:

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

Dim food(2,3)
food(0,0)="Apple"
food(0,1)="Banana"
food(0,2)="Orange"
food(0,3)="Lemon"
food(1,0)="Pizza"
food(1,1)="Hamburger"
food(1,2)="Spaghetti"
food(1,3)="Meatloaf"
food(2,0)="Cake"
food(2,1)="Cookie"
food(2,2)="Icecream"
food(2,3)="Chocolate"
document.write(LBound(food,1) & "<br />")
document.write(UBound(food,1) & "<br />")
document.write(LBound(food,2) & "<br />")
document.write(UBound(food,2) & "<br />")

</script>
```

The output of the code above will be:

```
0
2
0
3
```

Try it yourself »

Complete VBScript Reference

VBScript Split Function

Complete VBScript Reference

The Split function returns a zero-based, one-dimensional array that contains a specified number of substrings.

Syntax

```
Split(expression[,delimiter[,count[,compare]]])
```

Parameter	Description
expression	Required. A string expression that contains substrings and delimiters
delimiter	Optional. A string character used to identify substring limits. Default is the space character
count	Optional. The number of substrings to be returned. -1 indicates that all substrings are returned
compare	Optional. Specifies the string comparison to use. Can have one of the following values: • 0 = vbBinaryCompare - Perform a binary comparison • 1 = vbTextCompare - Perform a textual comparison

Examples

Example 1

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

a=Split("W3Schools is my favourite website")
for each x in a
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
W3Schools
is
my
favourite
website
```

Try it yourself »

Example 2

Splitting the text using the delimeter parameter

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

a=Split("Brown cow, White horse, Yellow chicken",",")
for each x in a
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
Brown cow
White horse
Yellow chicken
```

Try it yourself »

Example 3

Splitting the text using the delimeter parameter, and the count parameter

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

a=Split("W3Schools is my favourite website"," ",2)
for each x in a
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
W3Schools
is my favourite website
```

Try it yourself »

Example 4

Splitting the text using the delimeter parameter with a textual comparison:

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

a=Split("SundayMondayTuesdayWEDNESDAYThursdayFridaySaturday","day",-1,1)
for each x in a
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
Sun
Mon
Tues
WEDNES
Thurs
Fri
Satur
```

Try it yourself »

Example 5

Splitting the text using the delimeter parameter with a binary comparison:

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

a=Split("SundayMondayTuesdayWEDNESDAYThursdayFridaySaturday","day",-1,0)
for each x in a
    document.write(x & "<br />")
next

</script>
```

The output of the code above will be:

```
Sun
Mon
Tues
WEDNESDAYThurs
Fri
Satur
```

Try it yourself »

Complete VBScript Reference

VBScript UBound Function

Complete VBScript Reference

The UBound function returns the largest subscript for the indicated dimension of an array.

Tip: Use the UBound function with the LBound function to determine the size of an array.

Syntax

```
UBound(arrayname[,dimension])
```

Parameter	Description
arrayname	Required. The name of the array variable
dimension	Optional. Which dimension's upper bound to return. 1 = first dimension, 2 = second dimension, and so on. Default is 1

Examples

Example 1

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

days=Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat")
document.write(LBound(days) & "<br />")
document.write(UBound(days) & "<br />")

</script>
```

The output of the code above will be:

```
0
6
```

Try it yourself »

Example 2

A two dimensional array:

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

Dim food(2,3)
food(0,0)="Apple"
food(0,1)="Banana"
food(0,2)="Orange"
food(0,3)="Lemon"
food(1,0)="Pizza"
food(1,1)="Hamburger"
food(1,2)="Spaghetti"
food(1,3)="Meatloaf"
food(2,0)="Cake"
food(2,1)="Cookie"
food(2,2)="Icecream"
food(2,3)="Chocolate"
document.write(LBound(food,1) & "<br />")
document.write(UBound(food,1) & "<br />")
document.write(LBound(food,2) & "<br />")
document.write(UBound(food,2) & "<br />")

</script>
```

The output of the code above will be:

```
0
2
0
3
```

Try it yourself »

Complete VBScript Reference

VBScript InStr Function

Complete VBScript Reference

The InStr function returns the position of the first occurrence of one string within another.

The InStr function can return the following values:

- If string1 is "" - InStr returns 0
- If string1 is Null - InStr returns Null
- If string2 is "" - InStr returns start
- If string2 is Null - InStr returns Null
- If string2 is not found - InStr returns 0
- If string2 is found within string1 - InStr returns the position at which match is found
- If start > Len(string1) - InStr returns 0

Tip: Also look at the InStrRev function

Syntax

```
InStr([start,]string1,string2[,compare])
```

Parameter	Description
start	Optional. Specifies the starting position for each search. The search begins at the first character position (1) by default. This parameter is required if compare is specified
string1	Required. The string to be searched
string2	Required. The string expression to search for
compare	Optional. Specifies the string comparison to use. Default is 0 Can have one of the following values: • 0 = vbBinaryCompare - Perform a binary comparison • 1 = vbTextCompare - Perform a textual comparison

Examples

Example 1

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

txt="This is a beautiful day!"
document.write(InStr(txt,"beautiful"))

</script>
```

The output of the code above will be:

```
11
```

Try it yourself »

Example 2

Finding the letter "i", using different starting positions:

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

txt="This is a beautiful day!"
document.write(InStr(1,txt,"i") & "<br />")
document.write(InStr(7,txt,"i") & "<br />")

</script>
```

The output of the code above will be:

```
3
16
```

Try it yourself »

Example 3

Finding the letter "t", with textual, and binary, comparison:

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

txt="This is a beautiful day!"
document.write(InStr(1,txt,"t",1) & "<br />")
document.write(InStr(1,txt,"t",0) & "<br />")

</script>
```

The output of the code above will be:

```
1
15
```

Try it yourself »


Complete VBScript Reference

VBScript InStrRev Function

Complete VBScript Reference

The InStrRev function returns the position of the first occurrence of one string within another. The search begins from the end of string, but the position returned counts from the beginning of the string.

The InStrRev function can return the following values:

- If string1 is "" - InStrRev returns 0
- If string1 is Null - InStrRev returns Null
- If string2 is "" - InStrRev returns start
- If string2 is Null - InStrRev returns Null
- If string2 is not found - InStrRev returns 0
- If string2 is found within string1 - InStrRev returns the position at which match is found
- If start > Len(string1) - InStrRev returns 0

Tip: Also look at the InStr function

Syntax

```
InStrRev(string1,string2[,start[,compare]])
```

Parameter	Description
string1	Required. The string to be searched
string2	Required. The string expression to search for
start	Optional. Specifies the starting position for each search. The search begins at the last character position by default (-1)
compare	Optional. Specifies the string comparison to use. Default is 0 Can have one of the following values: • 0 = vbBinaryCompare - Perform a binary comparison • 1 = vbTextCompare - Perform a textual comparison

Examples

Example 1

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

txt="This is a beautiful day!"
document.write(InStrRev(txt,"beautiful"))

</script>
```

The output of the code above will be:

```
11
```

Try it yourself »

Example 2

Finding the letter "i", using different starting positions:

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

txt="This is a beautiful day!"
document.write(InStrRev(txt,"i",-1) & "<br />")
document.write(InStrRev(txt,"i",7) & "<br />")

</script>
```

The output of the code above will be:

```
16
6
```

Try it yourself »

Example 3

Finding the letter "T", with textual, and binary, comparison:

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

txt="This is a beautiful day!"
document.write(InStrRev(txt,"T",-1,1) & "<br />")
document.write(InStrRev(txt,"T",-1,0) & "<br />")

</script>
```

The output of the code above will be:

```
15
1
```

Try it yourself »


Complete VBScript Reference

VBScript LCase Function

Complete VBScript Reference

The LCase function converts a specified string to lowercase.

Tip: Also look at the UCase function.

Syntax

```
LCase(string)
```

Parameter	Description
string	Required. The string to be converted to lowercase

Examples

Example 1

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

txt="THIS IS A BEAUTIFUL DAY!"
document.write(LCase(txt))

</script>
```

The output of the code above will be:

```
this is a beautiful day!
```

Try it yourself »

Example 2

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

txt="This is a BEAUTIFUL day!"
document.write(LCase(txt))

</script>
```

The output of the code above will be:

```
this is a beautiful day!
```

Try it yourself »

Complete VBScript Reference

VBScript Left Function

Complete VBScript Reference

The Left function returns a specified number of characters from the left side of a string.

Tip: Use the Len function to find the number of characters in a string.

Tip: Also look at the Right function.

Syntax

```
Left(string,length)
```

Parameter	Description
string	Required. The string to return characters from
length	Required. Specifies how many characters to return. If set to 0, an empty string ("") is returned. If set to greater than or equal to the length of the string, the entire string is returned

Examples

Example 1

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

txt="This is a beautiful day!"
document.write(Left(txt,15))

</script>
```

The output of the code above will be:

```
This is a beaut
```

Try it yourself »

Example 2

Return the whole string:

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

txt="This is a beautiful day!"
x=Len(txt)
document.write(Left(txt,x))

</script>
```

The output of the code above will be:

```
This is a beautiful day!
```

Try it yourself »

Complete VBScript Reference

VBScript Len Function

Complete VBScript Reference

The Len function returns the number of characters in a string.

Syntax

```
Len(string)
```

Parameter	Description
string	A string expression

Examples

Example 1

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

txt="This is a beautiful day!"
document.write(Len(txt))

</script>
```

The output of the code above will be:

```
24
```

Try it yourself »

Example 2

You can also put the string directly into the Len function:

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

document.write(Len("This is a beautiful day!"))

</script>
```

The output of the code above will be:

```
24
```

Try it yourself »

Complete VBScript Reference

VBScript LTrim Function

Complete VBScript Reference

The LTrim function removes spaces on the left side of a string.

Tip: Also look at the RTrim and the Trim functions.

Syntax

```
LTrim(string)
```

Parameter	Description
string	Required. A string expression

Example

Example

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

fname=" Jack "
document.write("Hello" & LTrim(fname) & "and welcome.")

</script>
```

The output of the code above will be:

```
HelloJack and welcome.
```

Try it yourself »

Complete VBScript Reference

VBScript RTrim Function

Complete VBScript Reference

The RTrim function removes spaces on the right side of a string.

Tip: Also look at the LTrim and the Trim functions.

Syntax

```
RTrim(string)
```

Parameter	Description
string	Required. A string expression

Example

Example

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

fname=" Jack "
document.write("Hello" & RTrim(fname) & "and welcome.")

</script>
```

The output of the code above will be:

```
Hello Jackand welcome.
```

Try it yourself »

Complete VBScript Reference

VBScript Trim Function

Complete VBScript Reference

The Trim function removes spaces on both sides of a string.

Tip: Also look at the LTrim and the RTrim functions.

Syntax

```
Trim(string)
```

Parameter	Description
string	Required. A string expression

Example

Example

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

fname=" Jack "
document.write("Hello" & Trim(fname) & "and welcome.")

</script>
```

The output of the code above will be:

```
HelloJackand welcome.
```

Try it yourself »

Complete VBScript Reference

VBScript Mid Function

Complete VBScript Reference

The Mid function returns a specified number of characters from a string.

Tip: Use the Len function to determine the number of characters in a string.

Syntax

```
Mid(string,start[,length])
```

Parameter	Description
string	Required. The string expression from which characters are returned
start	Required. Specifies the starting position. If set to greater than the number of characters in string, it returns an empty string ("")
length	Optional. The number of characters to return

Examples

Example 1

Return 1 character, starting at postion 1:

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

txt="This is a beautiful day!"
document.write(Mid(txt,1,1))

</script>
```

The output of the code above will be:

```
T
```

Try it yourself »

Example 2

Return 15 characters, starting at postion 1:

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

txt="This is a beautiful day!"
document.write(Mid(txt,1,15))

</script>
```

The output of the code above will be:

```
This is a beaut
```

Try it yourself »

Example 3

Return all characters, starting at postion 1:

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

txt="This is a beautiful day!"
document.write(Mid(txt,1))

</script>
```

The output of the code above will be:

```
This is a beautiful day!
```

Try it yourself »

Example 4

Return all characters, starting at postion 12:

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

txt="This is a beautiful day!"
document.write(Mid(txt,12))

</script>
```

The output of the code above will be:

```
eautiful day!
```

Try it yourself »


Complete VBScript Reference

VBScript Replace Function

Complete VBScript Reference

The Replace function replaces a specified part of a string with another string a specified number of times.

Syntax

```
Replace(string,find,replacewith[,start[,count[,compare]]])
```

Parameter	Description
string	Required. The string to be searched
find	Required. The part of the string that will be replaced
replacewith	Required. The replacement substring
start	Optional. Specifies the start position. Default is 1. All characters before the start position will be removed.
count	Optional. Specifies the number of substitutions to perform. Default value is -1, which means make all possible substitutions
compare	Optional. Specifies the string comparison to use. Default is 0 Can have one of the following values: • 0 = vbBinaryCompare - Perform a binary comparison • 1 = vbTextCompare - Perform a textual comparison

Examples

Example 1

Replace the word "beautiful" with "fantastic":

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

txt="This is a beautiful day!"
document.write(Replace(txt,"beautiful","fantastic"))

</script>
```

The output of the code above will be:

```
This is a fantastic day!
```

Try it yourself »

Example 2

Replace the letter "i" with "##":

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

txt="This is a beautiful day!"
document.write(Replace(txt,"i","##"))

</script>
```

The output of the code above will be:

```
Th##s ##s a beaut##ful day!
```

Try it yourself »

Example 3

Replace the letter "i" with "##", starting at position 15:

Note that all characters before position 15 are removed.

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

```
txt="This is a beautiful day!"
document.write(Replace(txt,"i","##",15))

</script>
```

The output of the code above will be:

```
t##ful day!
```

Try it yourself »

Example 4

Replace the 2 first occurences of the letter "i" with "##", starting at position 1:

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

txt="This is a beautiful day!"
document.write(Replace(txt,"i","##",1,2))

</script>
```

The output of the code above will be:

```
Th##s ##s a beautiful day!
```

Try it yourself »

Example 5

Replace the letter "t" with "##", with textual, and binary, comparison:

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

txt="This is a beautiful day!"
document.write(Replace(txt,"t","##",1,-1,1) & "<br />")
document.write(Replace(txt,"t","##",1,-1,0))

</script>
```

The output of the code above will be:

```
##his is a beau##iful day!
This is a beau##iful day!
```

Try it yourself »

Complete VBScript Reference

VBScript Right Function

Complete VBScript Reference

The Right function returns a specified number of characters from the right side of a string.

Tip: Use the Len function to find the number of characters in a string.

Tip: Also look at the Left function.

Syntax

```
Right(string,length)
```

Parameter	Description
string	Required. The string to return characters from
length	Required. Specifies how many characters to return. If set to 0, an empty string ("") is returned. If set to greater than or equal to the length of the string, the entire string is returned

Examples

Example 1

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

txt="This is a beautiful day!"
document.write(Right(txt,10))

</script>
```

The output of the code above will be:

```
tiful day!
```

Try it yourself »

Example 2

Return the whole string:

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

txt="This is a beautiful day!"
x=Len(txt)
document.write(Right(txt,x))

</script>
```

The output of the code above will be:

```
This is a beautiful day!
```

Try it yourself »

Complete VBScript Reference

VBScript Space Function

Complete VBScript Reference

The Space function returns a string that consists of a specified number of spaces.

Syntax

```
Space(number)
```

Parameter	Description
number	Required. The number of spaces you want in the string

Example 1

```
Dim txt
txt=Space(10)
document.write(txt)

Output:

"          "
```

Complete VBScript Reference

VBScript StrComp Function

Complete VBScript Reference

The StrComp function compares two strings and returns a value that represents the result of the comparison.

The StrComp function can return one of the following values:

- -1 (if string1 < string2)
- 0 (if string1 = string2)
- 1 (if string1 > string2)
- Null (if string1 or string2 is Null)

Syntax

```
StrComp(string1,string2[,compare])
```

Parameter	Description
string1	Required. A string expression
string2	Required. A string expression
compare	Optional. Specifies the string comparison to use. Default is 0 Can have one of the following values: • 0 = vbBinaryCompare - Perform a binary comparison • 1 = vbTextCompare - Perform a textual comparison

Example 1

```
document.write(StrComp("VBScript","VBScript"))
Output:
0
```

Example 2

```
document.write(StrComp("VBScript","vbscript"))
Output:
-1
```

Example 3

```
document.write(StrComp("VBScript","vbscript",1))
Output:
0
```

Complete VBScript Reference

VBScript String Function

Complete VBScript Reference

The String function returns a string that contains a repeating character of a specified length.

Syntax

```
String(number,character)
```

Parameter	Description
number	Required. The length of the returned string
character	Required. The character that will be repeated

Example 1

```
document.write(String(10,"#"))
Output:
##########
```

Example 2

```
document.write(String(4,"*"))
Output:
****
```

Example 3

```
document.write(String(4,42))
Output:
****
```

Example 4

```
document.write(String(4,"XYZ"))
Output:
XXXX
```

Complete VBScript Reference

VBScript StrReverse Function

Complete VBScript Reference

The StrReverse function reverses a string.

Syntax

```
StrReverse(string)
```

Parameter	Description
string	Required. The string to be reversed

Example 1

```
Dim txt
txt="This is a beautiful day!"
document.write(StrReverse(txt))

Output:

!yad lufituaeb a si sihT
```

Complete VBScript Reference

VBScript UCase Function

Complete VBScript Reference

The UCase function converts a specified string to uppercase.

Tip: Also look at the LCase function.

Syntax

```
UCase(string)
```

Parameter	Description
string	Required. The string to be converted to uppercase

Examples

Example 1

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

txt="This is a beautiful day!"
document.write(UCase(txt))

</script>
```

The output of the code above will be:

```
THIS IS A BEAUTIFUL DAY!
```

Try it yourself »

Example 2

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

txt="This is a BEAUTIFUL day!"
document.write(UCase(txt))

</script>
```

The output of the code above will be:

```
THIS IS A BEAUTIFUL DAY!
```

Try it yourself »

Complete VBScript Reference

VBScript CreateObject Function

Complete VBScript Reference

The CreateObject function creates an object of a specified type.

Syntax

```
CreateObject(servername.typename[,location])
```

Parameter	Description
servername	Required. The name of the application that provides the object
typename	Required. The type/class of the object
location	Optional. Where to create the object

Example

Example

Creating a regular expression object:

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

txt="This is a beautiful day"
Set objReg=CreateObject("vbscript.regexp")
objReg.Pattern="i"
document.write(objReg.Replace(txt,"##"))

</script>
```

The output of the code above will be:

```
Th##s is a beautiful day
```

Try it yourself »

Complete VBScript Reference

VBScript Eval Function

Complete VBScript Reference

The Eval function evaluates an expression, like a function, and returns the result.

Syntax

```
Eval(expression)
```

Parameter	Description
expression	Required. The expression to evaluate

Example

Example

Creating a regular expression object:

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

function myFunction()
alert("Hello world")
end function

eval("myFunction()")

</script>
```

Try it yourself »

Complete VBScript Reference

VBScript GetLocale Function

Complete VBScript Reference

The GetLocale function returns the current locale ID.

A locale contains a set of user preference information: like language, country, region, and cultural conventions. The locale determines such things as keyboard layout, sort order, date, time, number, and currency formats.

The return value can be one of the 32-bit values shown in the Locale ID chart.

Tip: Use the SetLocale function to set the local ID.

Syntax

```
GetLocale()
```

Example

Example (IE Only)

```
<script type="text/vbscript">
document.write(GetLocale)
</script>
```

The output of the code above will be:

Try it yourself »

Locale ID Chart

Locale Description	Short String	Hex Value	Decimal Value
Afrikaans	af	0x0436	1078
Albanian	sq	0x041C	1052
Arabic – United Arab Emirates	ar-ae	0x3801	14337
Arabic - Bahrain	ar-bh	0x3C01	15361
Arabic - Algeria	ar-dz	0x1401	5121
Arabic - Egypt	ar-eg	0x0C01	3073
Arabic - Iraq	ar-iq	0x0801	2049
Arabic - Jordan	ar-jo	0x2C01	11265
Arabic - Kuwait	ar-kw	0x3401	13313
Arabic - Lebanon	ar-lb	0x3001	12289
Arabic - Libya	ar-ly	0x1001	4097
Arabic - Morocco	ar-ma	0x1801	6145
Arabic - Oman	ar-om	0x2001	8193
Arabic - Qatar	ar-qa	0x4001	16385
Arabic - Saudi Arabia	ar-sa	0x0401	1025
Arabic - Syria	ar-sy	0x2801	10241
Arabic - Tunisia	ar-tn	0x1C01	7169
Arabic - Yemen	ar-ye	0x2401	9217
Armenian	hy	0x042B	1067
Azeri – Latin	az-az	0x042C	1068
Azeri – Cyrillic	az-az	0x082C	2092
Basque	eu	0x042D	1069
Belarusian	be	0x0423	1059
Bulgarian	bg	0x0402	1026
Catalan	ca	0x0403	1027
Chinese - China	zh-cn	0x0804	2052
Chinese - Hong Kong S.A.R.	zh-hk	0x0C04	3076
Chinese – Macau S.A.R	zh-mo	0x1404	5124

Chinese - Singapore	zh-sg	0x1004	4100
Chinese - Taiwan	zh-tw	0x0404	1028
Croatian	hr	0x041A	1050
Czech	cs	0x0405	1029
Danish	da	0x0406	1030
Dutch – The Netherlands	nl-nl	0x0413	1043
Dutch - Belgium	nl-be	0x0813	2067
English - Australia	en-au	0x0C09	3081
English - Belize	en-bz	0x2809	10249
English - Canada	en-ca	0x1009	4105
English – Carribbean	en-cb	0x2409	9225
English - Ireland	en-ie	0x1809	6153
English - Jamaica	en-jm	0x2009	8201
English - New Zealand	en-nz	0x1409	5129
English – Phillippines	en-ph	0x3409	13321
English - South Africa	en-za	0x1C09	7177
English - Trinidad	en-tt	0x2C09	11273
English - United Kingdom	en-gb	0x0809	2057
English - United States	en-us	0x0409	1033
Estonian	et	0x0425	1061
Farsi	fa	0x0429	1065
Finnish	fi	0x040B	1035
Faroese	fo	0x0438	1080
French - France	fr-fr	0x040C	1036
French - Belgium	fr-be	0x080C	2060
French - Canada	fr-ca	0x0C0C	3084
French - Luxembourg	fr-lu	0x140C	5132
French - Switzerland	fr-ch	0x100C	4108
Gaelic – Ireland	gd-ie	0x083C	2108
Gaelic - Scotland	gd	0x043C	1084
German - Germany	de-de	0x0407	1031
German - Austria	de-at	0x0C07	3079
German - Liechtenstein	de-li	0x1407	5127
German - Luxembourg	de-lu	0x1007	4103
German - Switzerland	de-ch	0x0807	2055
Greek	el	0x0408	1032
Hebrew	he	0x040D	1037
Hindi	hi	0x0439	1081
Hungarian	hu	0x040E	1038
Icelandic	is	0x040F	1039
Indonesian	id	0x0421	1057
Italian - Italy	it-it	0x0410	1040
Italian - Switzerland	it-ch	0x0810	2064
Japanese	ja	0x0411	1041
Korean	ko	0x0412	1042
Latvian	lv	0x0426	1062
Lithuanian	lt	0x0427	1063
FYRO Macedonian	mk	0x042F	1071
Malay - Malaysia	ms-my	0x043E	1086
Malay – Brunei	ms-bn	0x083E	2110
Maltese	mt	0x043A	1082
Marathi	mr	0x044E	1102
Norwegian - Bokmål	no-no	0x0414	1044
Norwegian – Nynorsk	no-no	0x0814	2068
Polish	pl	0x0415	1045
Portuguese - Portugal	pt-pt	0x0816	2070

Portuguese - Brazil	pt-br	0x0416	1046
Raeto-Romance	rm	0x0417	1047
Romanian - Romania	ro	0x0418	1048
Romanian - Moldova	ro-mo	0x0818	2072
Russian	ru	0x0419	1049
Russian - Moldova	ru-mo	0x0819	2073
Sanskrit	sa	0x044F	1103
Serbian - Cyrillic	sr-sp	0x0C1A	3098
Serbian – Latin	sr-sp	0x081A	2074
Setsuana	tn	0x0432	1074
Slovenian	sl	0x0424	1060
Slovak	sk	0x041B	1051
Sorbian	sb	0x042E	1070
Spanish - Spain	es-es	0x0C0A	1034
Spanish - Argentina	es-ar	0x2C0A	11274
Spanish - Bolivia	es-bo	0x400A	16394
Spanish - Chile	es-cl	0x340A	13322
Spanish - Colombia	es-co	0x240A	9226
Spanish - Costa Rica	es-cr	0x140A	5130
Spanish - Dominican Republic	es-do	0x1C0A	7178
Spanish - Ecuador	es-ec	0x300A	12298
Spanish - Guatemala	es-gt	0x100A	4106
Spanish - Honduras	es-hn	0x480A	18442
Spanish - Mexico	es-mx	0x080A	2058
Spanish - Nicaragua	es-ni	0x4C0A	19466
Spanish - Panama	es-pa	0x180A	6154
Spanish - Peru	es-pe	0x280A	10250
Spanish - Puerto Rico	es-pr	0x500A	20490
Spanish - Paraguay	es-py	0x3C0A	15370
Spanish - El Salvador	es-sv	0x440A	17418
Spanish - Uruguay	es-uy	0x380A	14346
Spanish - Venezuela	es-ve	0x200A	8202
Sutu	sx	0x0430	1072
Swahili	sw	0x0441	1089
Swedish - Sweden	sv-se	0x041D	1053
Swedish - Finland	sv-fi	0x081D	2077
Tamil	ta	0x0449	1097
Tatar	tt	0X0444	1092
Thai	th	0x041E	1054
Turkish	tr	0x041F	1055
Tsonga	ts	0x0431	1073
Ukrainian	uk	0x0422	1058
Urdu	ur	0x0420	1056
Uzbek – Cyrillic	uz-uz	0x0843	2115
Uzbek – Latin	uz-uz	0x0443	1091
Vietnamese	vi	0x042A	1066
Xhosa	xh	0x0434	1076
Yiddish	yi	0x043D	1085
Zulu	zu	0x0435	1077

VBScript GetObject Function

Complete VBScript Reference

The GetObject function returns a reference to an automation object from a file.

Syntax

```
GetObject([pathname][,class])
```

Parameter	Description
pathname	Optional. The full path and name of the file that contains the automation object. If this parameter is omitted, the class parameter is required
class	Optional. The class of the automation object. This parameter uses this syntax: appname.objectype

Complete VBScript Reference

VBScript GetRef Function

Complete VBScript Reference

The GetRef function allows you to connect a VBScript procedure to an HTML event on your page.

Syntax

```
Set object.event=GetRef(procname)
```

Parameter	Description
object	Required. The HTML object the event is associated with.
event	Required. The event to which the procedure is to be bound.
procname	Required. The name of a Sub or a Function procedure to be associated with the HTML event.

Example

Example

```
<button id="myBtn">Click me</button>

<script type="text/vbscript">

document.getElementById("myBtn").onclick=GetRef("mySub")

Sub mySub()
alert("Hello world")
End Sub

</script>
```

Try it yourself »

Complete VBScript Reference

VBScript InputBox Function

Complete VBScript Reference

The InputBox function displays a dialog box, where the user can write some input and/or click on a button. If the user clicks the OK button or presses ENTER on the keyboard, the InputBox function will return the text in the text box. If the user clicks on the Cancel button, the function will return an empty string ("").

Note: A Help button is added to the dialog box when both the helpfile and the context parameter are specified.

Tip: Also look at the MsgBox function.

Syntax

```
InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])
```

Parameter	Description
prompt	Required. The message to show in the dialog box. Maximum length is 1024 characters. You can separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or carriage return–linefeed character combination (Chr(13) & Chr(10)) between each line
title	Optional. The title of the dialog box. Default is the application name
default	Optional. A default text in the text box
xpos	Optional. The prompt box' distance from the left edge of the screen. Measurement unit: twips*. If omitted, the dialog box is horizontally centered.
ypos	Optional. The prompt box' distance from the top edge of the screen. Measurement unit: twips*. If omitted, the dialog box is vertically positioned one-third of the way down the screen
helpfile	Optional. The name of a Help file to use. Must be used with the context parameter
context	Optional. The Help context number to the Help topic. Must be used with the helpfile parameter

* A twip is a measurement unit that is visually the same on all display systems.
1 twip is 1/1440 of an inch.

Examples

Example 1

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

Function myFunction()
fname=InputBox("Enter your name")
End Function

</script>
```

Try it yourself »

Example 2

A prompt box with a title:

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

Function myFunction()
fname=InputBox("Enter your name","Userinput")
End Function

</script>
```

Try it yourself »

Example 3

A prompt box with a deafult text in the inputbox:

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

```
Function myFunction()
fname=InputBox("Enter your name",,"Donald Duck")
End Function

</script>
```

Try it yourself »

Example 4

A prompt box wich is positioned 700 twips* from the left edge of your screen.

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

Function myFunction()
fname=InputBox("Enter your name",,,700)
End Function

</script>
```

Try it yourself »

Example 5

A prompt box wich is positioned 500 twips* from the top edge of your screen.

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

Function myFunction()
fname=InputBox("Enter your name",,,,500)
End Function

</script>
```

Try it yourself »

Complete VBScript Reference

VBScript IsEmpty Function

Complete VBScript Reference

The IsEmpty function returns a Boolean value that indicates whether a specified variable has been initialized or not. It returns true if the variable is uninitialized; otherwise, it returns False.

Syntax

```
IsEmpty(expression)
```

Parameter	Description
expression	Required. An expression (most often a variable name)

Example

Example

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

Dim x
document.write(IsEmpty(x) & "<br />")
x=10
document.write(IsEmpty(x) & "<br />")
x=Empty
document.write(IsEmpty(x) & "<br />")
x=Null
document.write(IsEmpty(x))

</script>
```

The output of the code above will be:

```
True
False
True
False
```

Try it yourself »

Complete VBScript Reference

VBScript IsNull Function

Complete VBScript Reference

The IsNull function returns a Boolean value that indicates whether a specified expression contains no valid data (Null). It returns True if expression is Null; otherwise, it returns False.

Syntax

```
IsNull(expression)
```

Parameter	Description
expression	Required. An expression

Example

Example

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

Dim x
document.write(IsNull(x) & "<br />")
x=10
document.write(IsNull(x) & "<br />")
x=Empty
document.write(IsNull(x) & "<br />")
x=Null
document.write(IsNull(x))

</script>
```

The output of the code above will be:

```
False
False
False
True
```

Try it yourself »

Complete VBScript Reference

VBScript IsNumeric Function

Complete VBScript Reference

The IsNumeric function returns a Boolean value that indicates whether a specified expression can be evaluated as a number. It returns True if the expression is recognized as a number; otherwise, it returns False.

Note: If expression is a date the IsNumeric function will return False.

Syntax

```
IsNumeric(expression)
```

Parameter	Description
expression	Required. An expression

Example

Example

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

Dim x
x=10
document.write(IsNumeric(x) & "<br />")
x=Empty
document.write(IsNumeric(x) & "<br />")
x=Null
document.write(IsNumeric(x) & "<br />")
x="10"
document.write(IsNumeric(x) & "<br />")
x="911 Help"
document.write(IsNumeric(x))

</script>
```

The output of the code above will be:

```
True
True
False
True
False
```

Try it yourself »

Complete VBScript Reference

VBScript IsObject Function

Complete VBScript Reference

The IsObject function returns a Boolean value that indicates whether the specified expression is an automation object. It returns True if expression is an automation object; otherwise, it returns False.

Syntax

```
IsObject(expression)
```

Parameter	Description
expression	Required. An expression

Examples

Example 1

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

Set x=document
document.write(IsObject(x))

</script>
```

The output of the code above will be:

```
True
```

Try it yourself »

Example 2

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

x="Peter"
document.write(IsObject(x))

</script>
```

The output of the code above will be:

```
False
```

Try it yourself »

Complete VBScript Reference

VBScript LoadPicture Function

Complete VBScript Reference

The LoadPicture function returns a picture object.

Graphics formats that is recognized by the LoadPicture function:

- bitmap files (.bmp)
- icon files (.ico)
- run-length encoded files (.rle)
- metafile files (.wmf)
- enhanced metafiles (.emf)
- GIF files (.gif)
- JPEG files (.jpg)

Note: This function is available only on 32-bit platforms.

Syntax

```
LoadPicture(picturename)
```

Parameter	Description
picturename	Required. The name of the picture file to be loaded

Complete VBScript Reference

VBScript MsgBox Function

Complete VBScript Reference

The MsgBox function displays a message box, waits for the user to click a button, and returns a value that indicates which button the user clicked.

The MsgBox function can return one of the following values:

- 1 = vbOK - OK was clicked
- 2 = vbCancel - Cancel was clicked
- 3 = vbAbort - Abort was clicked
- 4 = vbRetry - Retry was clicked
- 5 = vbIgnore - Ignore was clicked
- 6 = vbYes - Yes was clicked
- 7 = vbNo - No was clicked

Note: The user can press F1 to view the Help topic when both the helpfile and the context parameter are specified.

Tip: Also look at the InputBox function.

Syntax

```
MsgBox(prompt[,buttons][,title][,helpfile,context])
```

Parameter	Description
prompt	Required. The message to show in the message box. Maximum length is 1024 characters. You can separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or carriage return–linefeed character combination (Chr(13) & Chr(10)) between each line
buttons	Optional. A value or a sum of values that specifies the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. Default value is 0 • 0 = vbOKOnly - OK button only • 1 = vbOKCancel - OK and Cancel buttons • 2 = vbAbortRetryIgnore - Abort, Retry, and Ignore buttons • 3 = vbYesNoCancel - Yes, No, and Cancel buttons • 4 = vbYesNo - Yes and No buttons • 5 = vbRetryCancel - Retry and Cancel buttons • 16 = vbCritical - Critical Message icon • 32 = vbQuestion - Warning Query icon • 48 = vbExclamation - Warning Message icon • 64 = vbInformation - Information Message icon • 0 = vbDefaultButton1 - First button is default • 256 = vbDefaultButton2 - Second button is default • 512 = vbDefaultButton3 - Third button is default • 768 = vbDefaultButton4 - Fourth button is default • 0 = vbApplicationModal - Application modal (the current application will not work until the user responds to the message box) • 4096 = vbSystemModal - System modal (all applications wont work until the user responds to the message box) We can divide the buttons values into four groups: The first group (0–5) describes the buttons to be displayed in the message box, the second group (16, 32, 48, 64) describes the icon style, the third group (0, 256, 512, 768) indicates which button is the default; and the fourth group (0, 4096) determines the modality of the message box. When adding numbers to create a final value for the buttons parameter, use only one number from each group
title	Optional. The title of the message box. Default is the application name
helpfile	Optional. The name of a Help file to use. Must be used with the context parameter
context	Optional. The Help context number to the Help topic. Must be used with the helpfile parameter

Examples

Example 1

```
<script type="text/vbscript">
MsgBox("Hello world")
</script>
```

Try it yourself »

Example 2

A messagebox with a line feed:

```
<script type="text/vbscript">
MsgBox("Hello" & chr(13) & "world")
</script>
```

Try it yourself »

Example 3

Different buttonsets and different icons. Returns the value of the clicked button:

```
<script type="text/vbscript">
x=MsgBox("Hello world",n)
document.getElementById("myDiv").innerHTML="You clicked: " & x
</script>
```

Try it yourself »

Example 4

A messagebox with a title:

```
<script type="text/vbscript">
x=MsgBox("Are you a programmer",4,"Please answer")
</script>
```

Try it yourself »

Complete VBScript Reference

VBScript RGB Function

Complete VBScript Reference

The RGB function returns a number that represents an RGB color value.

Syntax

```
RGB(red,green,blue)
```

Parameter	Description
red	Required. A number from 0 to 255, inclusive, representing the red component of the color
green	Required. A number from 0 to 255, inclusive, representing the green component of the color
blue	Required. A number from 0 to 255, inclusive, representing the blue component of the color

Examples

Example 1

```
<script type="text/vbscript">
document.write(rgb(255,0,0))
</script>
```

The output of the code above will be:

```
255
```

Try it yourself »

Example 2

```
<script type="text/vbscript">
document.write(rgb(255,30,30))
</script>
```

The output of the code above will be:

```
1974015
```

Try it yourself »

Complete VBScript Reference

VBScript Round Function

Complete VBScript Reference

The Round function rounds a number.

Syntax

```
Round(expression[,numdecimalplaces])
```

Parameter	Description
expression	Required. The numeric expression to be rounded
numdecimalplaces	Optional. Specifies how many places to the right of the decimal are included in the rounding. Default is 0

Examples

Example 1

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

document.write(Round(24.13278) & "<br />")
document.write(Round(24.75122))

</script>
```

The output of the code above will be:

```
24
25
```

Try it yourself »

Example 2

How to round a number, keeping 2 decimals:

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

document.write(Round(24.13278,2))

</script>
```

The output of the code above will be:

```
24.13
```

Try it yourself »

Complete VBScript Reference

VBScript ScriptEngine, ScriptEngineBuildVersion, ScriptEngineMajorVersion, and ScriptEngineMinorVersion Functions

Complete VBScript Reference

ScriptEngine Function

The ScriptEngine function returns the scripting language in use.

This function can return one of the following strings:

- VBScript - Indicates that Microsoft Visual Basic Scripting Edition is the current scripting engine
- JScript - Indicates that Microsoft JScript is the current scripting engine
- VBA - Indicates that Microsoft Visual Basic for Applications is the current scripting engine

ScriptEngineBuildVersion Function

The ScriptEngineBuildVersion function returns the build version number of the scripting engine in use.

ScriptEngineMajorVersion Function

The ScriptEngineMajorVersion function returns the major version number of the scripting engine in use.

ScriptEngineMinorVersion Function

The ScriptEngineMinorVersion function returns the minor version number of the scripting engine in use.

Syntax

```
ScriptEngine

ScriptEngineBuildVersion

ScriptEngineMajorVersion

ScriptEngineMinorVersion
```

Example

Example

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

document.write(ScriptEngine & "<br />")
document.write(ScriptEngineBuildVersion & "<br />")
document.write(ScriptEngineMajorVersion & "<br />")
document.write(ScriptEngineMinorVersion)

</script>
```

The output of the code above could be:

```
VBScript
18702
5
8
```

Try it yourself »

Complete VBScript Reference

VBScript SetLocale Function

Complete VBScript Reference

The SetLocale function sets the locale ID and returns the previous locale ID.

A locale contains a set of user preferences, like language, country, region, and cultural conventions. The locale also determines such things as keyboard layout, sort order, date, time, number, and currency formats.

Tip: Use the GetLocale function to get the local ID.

Syntax

```
SetLocale(lcid)
```

Parameter	Description
lcid	Required. A short string, hex value, or decimal value in the Locale ID chart, that identifies a geographic locale. If the lcid parameter is set to 0, the locale will be set by the system

Examples

Example 1 (IE Only)

Check the current local ID using the GetLocale function. Set the local ID = 2057, and check the local ID again:

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

document.write(GetLocale() & "<br />")
SetLocale(2057)
document.write(GetLocale())

</script>
```

The output of the code above will be:

Try it yourself »

Example 2 (IE Only)

The SetLocale function also *returns* the previous ID:

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

document.write(SetLocale(1044))

</script>
```

The output of the code above will be:

Try it yourself »

Locale ID Chart

Locale Description	Short String	Hex Value	Decimal Value
Afrikaans	af	0x0436	1078
Albanian	sq	0x041C	1052
Arabic – United Arab Emirates	ar-ae	0x3801	14337
Arabic - Bahrain	ar-bh	0x3C01	15361
Arabic - Algeria	ar-dz	0x1401	5121
Arabic - Egypt	ar-eg	0x0C01	3073
Arabic - Iraq	ar-iq	0x0801	2049
Arabic - Jordan	ar-jo	0x2C01	11265
Arabic - Kuwait	ar-kw	0x3401	13313

Arabic - Lebanon	ar-lb	0x3001	12289
Arabic - Libya	ar-ly	0x1001	4097
Arabic - Morocco	ar-ma	0x1801	6145
Arabic - Oman	ar-om	0x2001	8193
Arabic - Qatar	ar-qa	0x4001	16385
Arabic - Saudi Arabia	ar-sa	0x0401	1025
Arabic - Syria	ar-sy	0x2801	10241
Arabic - Tunisia	ar-tn	0x1C01	7169
Arabic - Yemen	ar-ye	0x2401	9217
Armenian	hy	0x042B	1067
Azeri – Latin	az-az	0x042C	1068
Azeri – Cyrillic	az-az	0x082C	2092
Basque	eu	0x042D	1069
Belarusian	be	0x0423	1059
Bulgarian	bg	0x0402	1026
Catalan	ca	0x0403	1027
Chinese - China	zh-cn	0x0804	2052
Chinese - Hong Kong S.A.R.	zh-hk	0x0C04	3076
Chinese – Macau S.A.R	zh-mo	0x1404	5124
Chinese - Singapore	zh-sg	0x1004	4100
Chinese - Taiwan	zh-tw	0x0404	1028
Croatian	hr	0x041A	1050
Czech	cs	0x0405	1029
Danish	da	0x0406	1030
Dutch – The Netherlands	nl-nl	0x0413	1043
Dutch - Belgium	nl-be	0x0813	2067
English - Australia	en-au	0x0C09	3081
English - Belize	en-bz	0x2809	10249
English - Canada	en-ca	0x1009	4105
English – Carribbean	en-cb	0x2409	9225
English - Ireland	en-ie	0x1809	6153
English - Jamaica	en-jm	0x2009	8201
English - New Zealand	en-nz	0x1409	5129
English – Phillippines	en-ph	0x3409	13321
English - South Africa	en-za	0x1C09	7177
English - Trinidad	en-tt	0x2C09	11273
English - United Kingdom	en-gb	0x0809	2057
English - United States	en-us	0x0409	1033
Estonian	et	0x0425	1061
Farsi	fa	0x0429	1065
Finnish	fi	0x040B	1035
Faroese	fo	0x0438	1080
French - France	fr-fr	0x040C	1036
French - Belgium	fr-be	0x080C	2060
French - Canada	fr-ca	0x0C0C	3084
French - Luxembourg	fr-lu	0x140C	5132
French - Switzerland	fr-ch	0x100C	4108
Gaelic – Ireland	gd-ie	0x083C	2108
Gaelic - Scotland	gd	0x043C	1084
German - Germany	de-de	0x0407	1031
German - Austria	de-at	0x0C07	3079
German - Liechtenstein	de-li	0x1407	5127
German - Luxembourg	de-lu	0x1007	4103
German - Switzerland	de-ch	0x0807	2055
Greek	el	0x0408	1032
Hebrew	he	0x040D	1037

Hindi	hi	0x0439	1081
Hungarian	hu	0x040E	1038
Icelandic	is	0x040F	1039
Indonesian	id	0x0421	1057
Italian - Italy	it-it	0x0410	1040
Italian - Switzerland	it-ch	0x0810	2064
Japanese	ja	0x0411	1041
Korean	ko	0x0412	1042
Latvian	lv	0x0426	1062
Lithuanian	lt	0x0427	1063
FYRO Macedonian	mk	0x042F	1071
Malay - Malaysia	ms-my	0x043E	1086
Malay – Brunei	ms-bn	0x083E	2110
Maltese	mt	0x043A	1082
Marathi	mr	0x044E	1102
Norwegian - Bokmål	no-no	0x0414	1044
Norwegian – Nynorsk	no-no	0x0814	2068
Polish	pl	0x0415	1045
Portuguese - Portugal	pt-pt	0x0816	2070
Portuguese - Brazil	pt-br	0x0416	1046
Raeto-Romance	rm	0x0417	1047
Romanian - Romania	ro	0x0418	1048
Romanian - Moldova	ro-mo	0x0818	2072
Russian	ru	0x0419	1049
Russian - Moldova	ru-mo	0x0819	2073
Sanskrit	sa	0x044F	1103
Serbian - Cyrillic	sr-sp	0x0C1A	3098
Serbian – Latin	sr-sp	0x081A	2074
Setsuana	tn	0x0432	1074
Slovenian	sl	0x0424	1060
Slovak	sk	0x041B	1051
Sorbian	sb	0x042E	1070
Spanish - Spain	es-es	0x0C0A	1034
Spanish - Argentina	es-ar	0x2C0A	11274
Spanish - Bolivia	es-bo	0x400A	16394
Spanish - Chile	es-cl	0x340A	13322
Spanish - Colombia	es-co	0x240A	9226
Spanish - Costa Rica	es-cr	0x140A	5130
Spanish - Dominican Republic	es-do	0x1C0A	7178
Spanish - Ecuador	es-ec	0x300A	12298
Spanish - Guatemala	es-gt	0x100A	4106
Spanish - Honduras	es-hn	0x480A	18442
Spanish - Mexico	es-mx	0x080A	2058
Spanish - Nicaragua	es-ni	0x4C0A	19466
Spanish - Panama	es-pa	0x180A	6154
Spanish - Peru	es-pe	0x280A	10250
Spanish - Puerto Rico	es-pr	0x500A	20490
Spanish - Paraguay	es-py	0x3C0A	15370
Spanish - El Salvador	es-sv	0x440A	17418
Spanish - Uruguay	es-uy	0x380A	14346
Spanish - Venezuela	es-ve	0x200A	8202
Sutu	sx	0x0430	1072
Swahili	sw	0x0441	1089
Swedish - Sweden	sv-se	0x041D	1053
Swedish - Finland	sv-fi	0x081D	2077
Tamil	ta	0x0449	1097

Tatar	tt	0X0444	1092
Thai	th	0x041E	1054
Turkish	tr	0x041F	1055
Tsonga	ts	0x0431	1073
Ukrainian	uk	0x0422	1058
Urdu	ur	0x0420	1056
Uzbek – Cyrillic	uz-uz	0x0843	2115
Uzbek – Latin	uz-uz	0x0443	1091
Vietnamese	vi	0x042A	1066
Xhosa	xh	0x0434	1076
Yiddish	yi	0x043D	1085
Zulu	zu	0x0435	1077

 Complete VBScript Reference

VBScript TypeName Function

Complete VBScript Reference

The TypeName function returns the subtype of a specified variable.

The TypeName function can return one of the following values:

- Byte - Indicates a byte value
- Integer - Indicates an integer value
- Long - Indicates a long integer value
- Single - Indicates a single-precision floating-point value
- Double - Indicates a double-precision floating-point value
- Currency - Indicates a currency value
- Decimal - Indicates a decimal value
- Date - Indicates a date or time value
- String - Indicates a character string value
- Boolean - Indicates a boolean value; True or False
- Empty - Indicates an unitialized variable
- Null - Indicates no valid data
- <object type> - Indicates the actual type name of an object
- Object - Indicates a generic object
- Unknown - Indicates an unknown object type
- Nothing - Indicates an object variable that doesn't yet refer to an object instance
- Error - Indicates an error

Syntax

```
TypeName(varname)
```

Parameter	Description
varname	Required. A variable name

Example

Example

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

x="Hello World!"
document.write(TypeName(x) & "<br />")
x=4
document.write(TypeName(x) & "<br />")
x=4.675
document.write(TypeName(x) & "<br />")
x=Null
document.write(TypeName(x) & "<br />")
x=Empty
document.write(TypeName(x) & "<br />")
x=True
document.write(TypeName(x))

</script>
```

The output of the code above will be:

```
String
Integer
Double
Null
Empty
Boolean
```

Try it yourself »

Complete VBScript Reference

VBScript VarType Function

Complete VBScript Reference

The VarType function returns a value that indicates the subtype of a specified variable.

The VarType function can return one of the following values:

- 0 = vbEmpty - Indicates Empty (uninitialized)
- 1 = vbNull - Indicates Null (no valid data)
- 2 = vbInteger - Indicates an integer
- 3 = vbLong - Indicates a long integer
- 4 = vbSingle - Indicates a single-precision floating-point number
- 5 = vbDouble - Indicates a double-precision floating-point number
- 6 = vbCurrency - Indicates a currency
- 7 = vbDate - Indicates a date
- 8 = vbString - Indicates a string
- 9 = vbObject - Indicates an automation object
- 10 = vbError - Indicates an error
- 11 = vbBoolean - Indicates a boolean
- 12 = vbVariant - Indicates a variant (used only with arrays of Variants)
- 13 = vbDataObject - Indicates a data-access object
- 17 = vbByte - Indicates a byte
- 8192 = vbArray - Indicates an array

Note: If the variable is an array VarType() returns 8192 + VarType(array_element). Example: for an array of integer VarType() will return 8192 + 2 = 8194.

Syntax

```
VarType(varname)
```

Parameter	Description
varname	Required. A variable name

Example

Example

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

x="Hello World!"
document.write(VarType(x) & "<br />")
x=4
document.write(VarType(x) & "<br />")
x=4.675
document.write(VarType(x) & "<br />")
x=Null
document.write(VarType(x) & "<br />")
x=Empty
document.write(VarType(x) & "<br />")
x=True
document.write(VarType(x))

</script>
```

The output of the code above will be:

```
8
2
5
1
0
11
```

Try it yourself »

Complete VBScript Reference