VBScript Procedures

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
- without arguments, it must include an empty set of parentheses ()

```
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
```

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
```

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>
```

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))
```

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

VBScript Conditional Statements

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>
```

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>
```

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>
```

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.

VBScript Looping

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>
```

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>
```

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.

VBScript Keywords

Keyword	**Description**
Empty	Used to indicate an uninitialized variable value. A variable value is uninitialized when it is first created and no value is assigned to it, or when a variable value is explicitly set to empty. Example: Dim x 'the variable x is uninitialized! x="ff" 'the variable x is NOT uninitialized anymore x=Empty 'the variable x is uninitialized! **Note:** This is not the same as Null!!
IsEmpty	Used to test if a variable is uninitialized. Example: If (IsEmpty(x)) 'is x uninitialized?
Nothing	Used to indicate an uninitialized object value, or to disassociate an object variable from an object to release system resources. Example: Set myObject=Nothing
Is Nothing	Used to test if a value is an initialized object. Example: If (myObject Is Nothing) 'is it unset? **Note:** If you compare a value to Nothing, you will not get the right result! Example: If (myObject = Nothing) 'always false!
Null	Used to indicate that a variable contains no valid data.

	One way to think of Null is that someone has explicitly set the value to "invalid", unlike Empty where the value is "not set". **Note:** This is not the same as Empty or Nothing!! Example: x=Null 'x contains no valid data
IsNull	Used to test if a value contains invalid data. Example: if (IsNull(x)) 'is x invalid?
True	Used to indicate a Boolean condition that is correct (True has a value of -1)
False	Used to indicate a Boolean condition that is not correct (False has a value of 0)

VBScript Functions

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