VBSCRIPT

scripting language

tutorialspoint

SIMPLYEASYLEARNING

About the Tutorial

Microsoft VBScript (Visual Basic Script) is a general-purpose, lightweight and active scripting language developed by Microsoft that is modelled on Visual Basic. Nowadays, VBScript is the primary scripting language for Quick Test Professional (QTP), which is a test automation tool. This tutorial will teach you how to use VBScript in your day-to-day life of any Web-based or automation project development.

Audience

This tutorial has been prepared for beginners to help them understand the basic-to-advanced functionality of VBScript. After completing this tutorial, you will find yourself at a moderate level of expertise in using Microsoft VBScript from where you can take yourself to the next levels.

Prerequisites

You need to have a good understanding of any computer programming language in order to make the most of this tutorial. If you have done programming in any client-side languages like Javascript, then it will be quite easy for you to learn the ropes of VBScript.

Copyright & Disclaimer

Table of Contents

Part 1: VBScript Basics

1. VBScript– Overview

VBScript stands for **V**isual **B**asic Scripting that forms a subset of Visual Basic for Applications (VBA). VBA is a product of Microsoft which is included NOT only in other Microsoft products such as MS Project and MS Office but also in Third Party tools such as AUTO CAD.

Features of VBScript

- VBScript is a lightweight scripting language, which has a lightning fast interpreter.
- VBScript, for the most part, is case insensitive. It has a very simple syntax, easy to learn and to implement.
- Unlike C++ or Java, VBScript is an object-based scripting language and NOT an Object-Oriented Programming language.
- It uses Component Object Model **(COM)** in order to access the elements of the environment in which it is executing.
- Successful execution of VBScript can happen only if it is executed in Host Environment such as Internet Explorer **(IE)**, Internet Information Services **(IIS)** and Windows Scripting Host **(WSH)**

VBScript – Version History and Uses

VBScript was introduced by Microsoft way back in 1996 and its first version was 1.0. The current stable version of VBScript is 5.8, which is available as part of IE8 or Windows 7. The VBScript usage areas are aplenty and not restricted to the below list.

- VBScript is used as a scripting language in one of the popular Automation testing tools – Quick Test Professional abbreviated as **QTP**.
- Windows Scripting Host, which is used mostly by Windows System administrators for automating the Windows Desktop.
- Active Server Pages **(ASP)**, a server side scripting environment for creating dynamic webpages which uses VBScript or Java Script.
- VBScript is used for Client side scripting in Microsoft Internet Explorer.
- Microsoft Outlook Forms usually runs on VBScript; however, the application level programming relies on VBA (Outlook 2000 onwards).

Disadvantages

- VBScript is used only by IE Browsers. Other browsers such as Chrome, Firefox DONOT Support VBScript. Hence, JavaScript is preferred over VBScript.
- VBScript has a Limited command line support.
- Since there is no development environment available by default, debugging is difficult.

Where VBScript is Today?

The current version of VBScript is 5.8, and with the recent development of .NET framework, Microsoft has decided to provide future support of VBScript within ASP.NET for web development. Hence, there will NOT be any more new versions of VBScript engine but the entire defect fixes and security issues are being addressed by the Microsoft sustaining Engineering Team. However, VBScript engine would be shipped as part of all Microsoft Windows and IIS by default.

2. VBScript– Syntax

Your First VBScript

Let us write a VBScript to print out "Hello World".

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

In the above example, we called a function *document.write*, which writes a string into the HTML document. This function can be used to write text, HTML, or both. So, the above code will display the following result:

```
Hello World!
```

Whitespace and Line Breaks

VBScript ignores spaces, tabs, and newlines that appear within VBScript programs. One can use spaces, tabs, and newlines freely within the program, so you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

Formatting

VBScript is based on Microsoft's Visual Basic. Unlike JavaScript, no statement terminators such as semicolon is used to terminate a particular statement.

Single Line Syntax

Colons are used when two or more lines of VBScript ought to be written in a single line. Hence, in VBScript, Colons act as a line separator.

```
<script language="vbscript" type="text/vbscript">
  var1 = 10 : var2 = 20
```

```
</script>
```

Multiple Line Syntax

When a statement in VBScript is lengthy and if user wishes to break it into multiple lines, then the user has to use underscore "_". This improves the readability of the code. The following example illustrates how to work with multiple lines.

```
<script language="vbscript" type="text/vbscript">
  var1 = 10
  var2 = 20
  Sum = var1 + var2
  document.write("The Sum of two numbers"&_
  "var1 and var2 is " & Sum)
</script>
```

Reserved Words

The following list shows the reserved words in VBScript. These reserved words SHOULD NOT be used as a constant or variable or any other identifier names.

Loop	LSet	Me
Mod	New	Next
Not	Nothing	Null
On	Option	Optional
Or	ParamArray	Preserve
Private	Public	RaiseEvent
ReDim	Rem	Resume
RSet	Select	Set

Shared	Single	Static
Stop	Sub	Then
To	True	Type
And	As	Boolean
ByRef	Byte	ByVal
Call	Case	Class
Const	Currency	Debug
Dim	Do	Double
Each	Else	ElseIf
Empty	End	EndIf
Enum	Eqv	Event
Exit	False	For
Function	Get	GoTo
If	Imp	Implements
In	Integer	Is
Let	Like	Long
TypeOf	Until	Variant
Wend	While	With

Xor	Eval	Execute
Msgbox	Erase	ExecuteGlobal
Option Explicit	Randomize	SendKeys

Case Sensitivity

VBScript is a **case-insensitive language**. This means that language keywords, variables, function names and any other identifiers need NOT be typed with a consistent capitalization of letters. So identifiers int_counter, INT_Counter and INT_COUNTER have the same meaning within VBScript.

Comments in VBScript

Comments are used to document the program logic and the user information with which other programmers can seamlessly work on the same code in future. It can include information such as developed by, modified by and it can also include incorporated logic. Comments are ignored by the interpreter while execution. Comments in VBScript are denoted by two methods.

Any statement that starts with a Single Quote (‘) is treated as comment. Following is the example:

```
<script language="vbscript" type="text/vbscript">
<!—
  ' This Script is invoked after successful login
  ' Written by : TutorialsPoint
  ' Return Value : True / False
//- >
</script>
```

Any statement that starts with the keyword “REM”. Following is the example:

```
<script language="vbscript" type="text/vbscript">
<!—
  REM This Script is written to Validate the Entered Input
  REM Modified by  : Tutorials point/user2
//- > </script>
```

3. VBScript– Enabling in Browsers

Not all the modern browsers support VBScript. VBScript is supported just by Microsoft's Internet Explorer while other browsers (Firefox and Chrome) support just JavaScript. Hence, developers normally prefer JavaScript over VBScript.

Though Internet Explorer (IE) supports VBScript, you may need to enable or disable this feature manually. This tutorial will make you aware of the procedure of enabling and disabling VBScript support in Internet Explorer.

VBScript in Internet Explorer

Here are simple steps to turn on or turn off VBScript in your Internet Explorer:

- Follow Tools -> Internet Options from the menu
- Select Security tab from the dialog box
- Click the Custom Level button
- Scroll down till you find Scripting option
- Select *Enable* radio button under Active scripting
- Finally click OK and come out

To disable VBScript support in your Internet Explorer, you need to select *Disable* radio button under **Active scripting**.

4. VBScript– Placements

VBScript Placement in HTML File

There is a flexibility given to include VBScript code anywhere in an HTML document. But the most preferred way to include VBScript in your HTML file is as follows:

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

In the following section, we will see how we can put VBScript in different ways:

VBScript in <head>...</head> section

If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as follows:

```
<html>
<head>
<script type="text/Vbscript">
<!--
Function sayHello()
   Msgbox("Hello World")
End Function
//-->
</script>
</head>
<body>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

It will produce the following result: A button with the name SayHello. Upon clicking on the Button, the message box is displayed to the user with the message "Hello World".

VBScript in <body>...</body> section

If you need a script to run as the page loads so that the script generates content in the page, the script goes in the <body> portion of the document. In this case, you would not have any function defined using VBScript:

```
<html>
<head>
</head>
<body>
<script type="text/vbscript">
<!--
    document.write("Hello World")
//-->
</script>
<p>This is web page body </p>
</body>
</html>
```

It will produce the following result:

```
Hello World
This is web page body
```

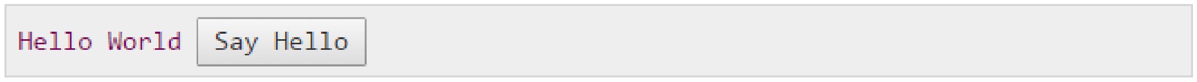
VBScript in <body> and <head> Sections

You can put your VBScript code in <head> and <body> section altogether as follows:

```
<html>
<head>
<script type="text/vbscript">
<!--
Function sayHello()
```

```
   msgbox("Hello World")
End Function
//-->
</script>
</head>
<body>
<script type="text/vbscript">
<!--
document.write("Hello World")
//-->
</script>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

It will produce the following result: Hello World message with a 'Say Hello' button. Upon Clicking on the button a message box with a message "Hello World" is displayed to the user.


Hello World

VBScript in External File

As you begin to work more extensively with VBScript, you will likely find that there are cases, where you are reusing identical VBScript code on multiple pages of a site. You are not restricted to be maintaining identical code in multiple HTML files.

The *script* tag provides a mechanism to allow you to store VBScript in an external file and then include it into your HTML files. Here is an example to show how you can include an external VBScript file in your HTML code using *script* tag and its *src* attribute:

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

```
</html>
```

To use VBScript from an external file source, you need to write your all VBScript source code in a simple text file with extension ".vbs" and then include that file as shown above. For example, you can keep the following content in filename.vbs file and then you can use *sayHello* function in your HTML file after including filename.vbs file.

```
Function sayHello()

   Msgbox "Hello World"

End Function
```

VBScript Placement in QTP

VBScript is placed in QTP (Quick Test Professional) tool but it is NOT enclosed within HTML Tags. The Script File is saved with the extension .vbs and it is executed by Quick Test Professional execution engine.

5. VBScript– Variables

VBScript Variables

A variable is a named memory location used to hold a value that can be changed during the script execution. VBScript has only **ONE** fundamental data type, **Variant**.

Rules for Declaring Variables:

- Variable Name must begin with an alphabet.
- Variable names cannot exceed 255 characters.
- Variables Should NOT contain a period (.)
- Variable Names should be unique in the declared context.

Declaring Variables

Variables are declared using “dim” keyword. Since there is only ONE fundamental data type, all the declared variables are variant by default. Hence, a user **NEED NOT** mention the type of data during declaration.

Example 1: In this Example, IntValue can be used as a String, Integer or even arrays.

```
Dim Var
```

Example 2: Two or more declarations are separated by comma(,)

```
Dim Variable1,Variable2
```

Assigning Values to the Variables

Values are assigned similar to an algebraic expression. The variable name on the left hand side followed by an equal to (=) symbol and then its value on the right hand side.

Rules

- The numeric values should be declared without double quotes.
- The String values should be enclosed within double quotes(")

- Date and Time variables should be enclosed within hash symbol(#)

Examples

```
' Below Example, The value 25 is assigned to the variable.
Value1 = 25

' A String Value ‘VBScript’ is assigned to the variable StrValue.
StrValue = “VBScript”

' The date 01/01/2020 is assigned to the variable DToday.
Date1 = #01/01/2020#

' A Specific Time Stamp is assigned to a variable in the below example.
Time1 = #12:30:44 PM#
```

Scope of the Variables

Variables can be declared using the following statements that determines the scope of the variable. The scope of the variable plays a crucial role when used within a procedure or classes.

- Dim
- Public
- Private

Dim

Variables declared using “Dim” keyword at a Procedure level are available only within the same procedure. Variables declared using “Dim” Keyword at script level are available to all the procedures within the same script.

Example: In the below example, the value of Var1 and Var2 are declared at script level while Var3 is declared at procedure level.

Note: The scope of this chapter is to understand Variables. Functions would be dealt in detail in the upcoming chapters.

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

Dim Var1
Dim Var2

Call add()
Function add()
   Var1 = 10
   Var2 = 15
   Dim Var3
   Var3 = Var1+Var2
   Msgbox Var3 'Displays 25, the sum of two values.
End Function

Msgbox Var1   ' Displays 10 as Var1 is declared at Script level
Msgbox Var2   ' Displays 15 as Var2 is declared at Script level
Msgbox Var3   ' Var3 has No Scope outside the procedure. Prints Empty

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

Public

Variables declared using "Public" Keyword are available to all the procedures across all the associated scripts. When declaring a variable of type "public", Dim keyword is replaced by "Public".

Example: In the following example, Var1 and Var2 are available at script level while Var3 is available across the associated scripts and procedures as it is declared as Public.

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

```
<body>
<script language="vbscript" type="text/vbscript">
Dim Var1
Dim Var2
Public Var3

Call add()

Function add()
   Var1 = 10
   Var2 = 15
   Var3 = Var1+Var2
   Msgbox Var3 'Displays 25, the sum of two values.
End Function

Msgbox Var1   ' Displays 10 as Var1 is declared at Script level
Msgbox Var2   ' Displays 15 as Var2 is declared at Script level
Msgbox Var3   ' Displays 25 as Var3 is declared as Public

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

Private

Variables that are declared as "Private" have scope only within that script in which they are declared. When declaring a variable of type "Private", Dim keyword is replaced by "Private".

Example: In the following example, Var1 and Var2 are available at Script Level. Var3 is declared as Private and it is available only for this particular script. Use of "Private" Variables is more pronounced within the Class.

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

```
Dim Var1
Dim Var2
Private Var3

Call add()
Function add()
   Var1 = 10
   Var2 = 15
   Var3 = Var1+Var2
   Msgbox Var3 'Displays the sum of two values.
End Function

Msgbox Var1   ' Displays 10 as Var1 is declared at Script level
Msgbox Var2   ' Displays 15 as Var2 is declared at Script level
Msgbox Var3   ' Displays 25 but Var3 is available only for this script.
</script>
</body>
</html>
```

6. VBScript– Constants

Constant is a named memory location used to hold a value that CANNOT be changed during the script execution. If a user tries to change a Constant Value, the Script execution ends up with an error. Constants are declared the same way the variables are declared.

Declaring Constants

Syntax

```
[Public | Private] Const Constant_Name = Value
```

The Constant can be of type Public or Private. The Use of Public or Private is Optional. The Public constants are available for all the scripts and procedures while the Private Constants are available within the procedure or Class. One can assign any value such as number, String or Date to the declared Constant.

Example 1

In this example, the value of pi is 3.4 and it displays the area of the circle in a message box.

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

  Dim intRadius
  intRadius = 20
  const pi=3.14
  Area = pi*intRadius*intRadius
  Msgbox Area

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

Example 2

The following example illustrates how to assign a String and Date Value to a Constant.

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

  Const myString = "VBScript"
  Const myDate = #01/01/2050#
  Msgbox myString
  Msgbox myDate
</script>
</body>
</html>
```

Example 3

In the following example, the user tries to change the Constant Value; hence, it will end up with an **Execution Error.**

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
   Dim intRadius
   intRadius = 20
   const pi=3.14
   pi = pi*pi     'pi VALUE CANNOT BE CHANGED.THROWS ERROR'
   Area = pi*intRadius*intRadius
   Msgbox Area
   </script>
</body>
</html>
```

7. VBScript– Operators

What is an Operator?

Let’s take an expression *4 + 5 is equal to 9*. Here, 4 and 5 are called **operands** and + is called the **operator**. VBScript language supports following types of operators:

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Concatenation Operators

The Arithmetic Operators

VBScript supports the following arithmetic operators:

Assume variable A holds 5 and variable B holds 10, then:

Operator	Description	Example
+	Adds two operands	A + B will give 15
-	Subtracts second operand from the first	A - B will give -5
*	Multiply both operands	A * B will give 50
/	Divide numerator by denominator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B MOD A will give 0
^	Exponentiation Operator	B ^ A will give 100000

Example

Try the following example to understand all the arithmetic operators available in VBScript:

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
   Dim a : a = 5
   Dim b : b = 10
   Dim c

   c = a+b
   Document.write ("Addition Result is " &c)
   Document.write ("<br></br>")    'Inserting a Line Break for readability
   c = a-b
   Document.write ("Subtraction Result is " &c)
   Document.write ("<br></br>")   'Inserting a Line Break for readability
   c = a*b
   Document.write ("Multiplication Result is " &c)
   Document.write ("<br></br>")
   c = b/a
   Document.write ("Division Result is " &c)
   Document.write ("<br></br>")
   c = b MOD a
   Document.write ("Modulus Result is " &c)
   Document.write ("<br></br>")
   c = b^a
   Document.write ("Exponentiation Result is " &c)
   Document.write ("<br></br>")
</script>
</body>
```

```
</html>
```

When you save it as .html and execute it in Internet Explorer, then the above script will produce the following result:

```
Addition Result is 15

Subtraction Result is -5

Multiplication Result is 50

Division Result is 2

Modulus Result is 0

Exponentiation Result is 100000
```

The Comparison Operators

VBScript supports the following comparison operators:

Assume variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is False.
<>	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A <> B) is True.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is False.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is True.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is False.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is True.

Example

Try the following example to understand all the Comparison operators available in VBScript:

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
  Dim a : a = 10
  Dim b : b = 20
  Dim c

  If a=b Then
     Document.write ("Operator Line 1 : True")
     Document.write ("<br></br>")  'Inserting a Line Break for readability
  Else
     Document.write ("Operator Line 1 : False")
     Document.write ("<br></br>")  'Inserting a Line Break for readability
  End If

  If a<>b Then
     Document.write ("Operator Line 2 : True")
     Document.write ("<br></br>")
  Else
     Document.write ("Operator Line 2 : False")
     Document.write ("<br></br>")
  End If

  If a>b Then
     Document.write ("Operator Line 3 : True")
     Document.write ("<br></br>")
  Else
     Document.write ("Operator Line 3 : False")
     Document.write ("<br></br>")
  End If

  If a<b Then
     Document.write ("Operator Line 4 : True")
     Document.write ("<br></br>")
  Else
     Document.write ("Operator Line 4 : False")
     Document.write ("<br></br>")
  End If

  If a>=b Then
     Document.write ("Operator Line 5 : True")
     Document.write ("<br></br>")
  Else
     Document.write ("Operator Line 5 : False")
     Document.write ("<br></br>")
  End If

  If a<=b Then
     Document.write ("Operator Line 6 : True")
     Document.write ("<br></br>")
  Else
     Document.write ("Operator Line 6 : False")
     Document.write ("<br></br>")
  End If

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

When you save it as .html and execute it in Internet Explorer, then the above script will produce the following result:

```
Operator Line 1 : False
Operator Line 2 : True

Operator Line 3 : False

Operator Line 4 : True

Operator Line 5 : False

Operator Line 6 : True
```

The Logical Operators

VBScript supports the following logical operators:

Assume variable A holds 10 and variable B holds 0, then:

Operator	Description	Example
AND	Called Logical AND operator. If both the conditions are True then Expression becomes true.	a<>0 AND b<>0 is False.
OR	Called Logical OR Operator. If any of the two conditions are True then condition becomes true.	a<>0 OR b<>0 is true.
NOT	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	NOT(a<>0 OR b<>0) is false.
XOR	Called Logical Exclusion. It is the combination of NOT and OR Operator. If one, and only one, of the expressions evaluates to True, result is True.	(a<>0 XOR b<>0) is false.

Example

Try the following example to understand all the Logical operators available in VBScript:

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
  Dim a : a = 10
  Dim b : b = 0
  Dim c

  If a<>0 AND b<>0 Then
     Document.write ("AND Operator Result is : True")
     Document.write ("<br></br>")  'Inserting a Line Break for readability
  Else
     Document.write ("AND Operator Result is : False")
     Document.write ("<br></br>")  'Inserting a Line Break for readability
  End If

  If a<>0 OR b<>0 Then
     Document.write ("OR Operator Result is : True")
     Document.write ("<br></br>")
  Else
     Document.write ("OR Operator Result is : False")
     Document.write ("<br></br>")
  End If

  If NOT(a<>0 OR b<>0) Then
     Document.write ("NOT Operator Result is : True")
     Document.write ("<br></br>")
  Else
     Document.write ("NOT Operator Result is : False")
     Document.write ("<br></br>")
  End If

  If (a<>0 XOR b<>0) Then
```

```
      Document.write ("XOR Operator Result is : True")
      Document.write ("<br></br>")
   Else
      Document.write ("XOR Operator Result is : False")
      Document.write ("<br></br>")
   End If
</script>
</body>
</html>
```

When you save it as .html and execute it in Internet Explorer, then the above script will produce the following result:

```
AND Operator Result is : False

OR Operator Result is : True

NOT Operator Result is : False

XOR Operator Result is : True
```

The Concatenation Operators

VBScript supports the following Concatenation operators:

Assume variable A holds 5 and variable B holds 10 then:

Operator	Description	Example
+	Adds two Values as Variable Values are Numeric	A + B will give 15
&	Concatenates two Values	A & B will give 510

Example

Try the following example to understand the Concatenation operator available in VBScript:

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
  Dim a : a = 5
  Dim b : b = 10
  Dim c

  c=a+b
  Document.write ("Concatenated value:1 is " &c) 'Numeric addition
  Document.write ("<br></br>")  'Inserting a Line Break for readability
  c=a&b
  Document.write ("Concatenated value:2 is " &c) 'Concatenate two numbers
  Document.write ("<br></br>")  'Inserting a Line Break for readability
</script>
</body>
</html>
```

When you save it as .html and execute it in Internet Explorer, then the above script will produce the following result:

```
Concatenated value:1 is 15
Concatenated value:2 is 510
```

Concatenation can also be used for concatenating two strings. Assume variable A="Microsoft" and variable B="VBScript" then:

Operator	Description	Example
+	Concatenates two Values	A + B will give MicrosoftVBScript
&	Concatenates two Values	A & B will give MicrosoftVBScript

Example

Try the following example to understand the Concatenation operator available in VBScript:

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
  Dim a : a = "Microsoft"
  Dim b : b = "VBScript"
  Dim c

  c=a+b
  Document.write ("Concatenated value:1 is " &c) 'Numeric addition
  Document.write ("<br></br>")  'Inserting a Line Break for readability
  c=a&b
  Document.write ("Concatenated value:2 is " &c) 'Concatenate two numbers
  Document.write ("<br></br>")  'Inserting a Line Break for readability
</script>
</body>
</html>
```

When you save it as .html and execute it in Internet Explorer, then the above script will produce the following result:

```
Concatenated value:1 is MicrosoftVBScript
Concatenated value:2 is MicrosoftVBScript
```