VBSCRIPT MISCELLANEOUS STATEMENTS

There are few other important statements which helps the developers to develop an efficient script. Below are the list of statements tabulated and explained in detail with examples.

Category	Function Name/Statement Name
Options	Option Explicit
Script Engine ID	ScriptEngine
variants	IsArray, IsEmpty, IsNull, IsNumeric, IsObject, TypeName
Expression	Eval,Execute
Control Statement	With...End With
Math Function	Randomize

Option Explicit

Option Explicit forces the developer to declare the variables using **Dim** statement before they are used in some part of the code.

Syntax

```
Option Explicit
```

Example

If we use **Option Explicit** and if we don't declare the variables then the interpreter will throw and error.

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

  Option Explicit
  Dim x,y,z,a
  x = 10
  y = 20
  z = fnadd(x,y)
  a = fnmultiply(x,y)

  Function fnadd(x,y)
      fnadd = x+y
  End Function

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

ScriptEngine

ScriptEngine represents the details of the scripting language in use. It is also used in combination with **ScriptEngineMajorVersion**, **ScriptEngineMinorVersion**, **ScriptEngineBuildVersion** which gives the major version of the vbscript engine, minor version the vbscript engine and the

build version of vbscript respectively.

Syntax

```
ScriptEngine
```

Example

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

  Dim scriptdetails
  scriptdetails =  " Version " & ScriptEngine & " - "
  'For getting Major version, use ScriptEngineMajorVersion'

  scriptdetails = scriptdetails & ScriptEngineMajorVersion & "."

  'For getting Minor version, use ScriptEngineMinorVersion'
  scriptdetails = scriptdetails & ScriptEngineMinorVersion & "."

  'For getting Build version, use ScriptEngineBuildVersion'
  scriptdetails = scriptdetails & ScriptEngineBuildVersion

  Document.write scriptdetails

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

```
Version VBScript - 5.8.16996
```

IsEmpty

The Function IsEmpty is used to check whether or not the expression is empty. It returns a boolean value. **IsEmpty** returns True if the variable is uninitialized or explicitly set to Empty. Otherwise the expression returns False.

Syntax

```
IsEmpty(expression)
```

Example

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

  Dim var, MyCheck
  MyCheck = IsEmpty(var)
  Document.write "Line 1 : " & MyCheck & "<br />"

  var = Null   ' Assign Null.
  MyCheck = IsEmpty(var)
  Document.write "Line 2 : " & MyCheck & "<br />"

  var = Empty   ' Assign Empty.
  MyCheck = IsEmpty(var)
  Document.write "Line 3 : " & MyCheck & "<br />"
```

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

```
Line 1 : True
Line 2 : False
Line 3 : True
```

IsNull

The Function IsNull is used to check whether or not the expression has a valid data. It returns a boolean value. **IsNull** returns True if the variable is Null otherwise the expression returns False.

Syntax

```
IsNull(expression)
```

Example

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

   Dim var, res
   res = IsNull(var)
   document.write "Line 1 : " & res & "<br />"

   var = Null
   res = IsNull(var)
   document.write "Line 2 : " & res & "<br />"

   var = Empty
   res = IsNull(var)
   document.write "Line 3 : " & res & "<br />"

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

```
Line 1 : False
Line 2 : True
Line 3 : False
```

IsObject

The IsObject Function is used to check whether or not the expression has a valid Object. It returns a boolean value. **IsObject** returns True if the expression contains an object subtype otherwise the expression returns False.

Syntax

```
IsObject(expression)
```

Example

```
<!DOCTYPE html>
<html>
<body>
<script language="vbscript" type="text/vbscript">
  Dim fso,b
  b = 10
  set fso = createobject("Scripting.Filesystemobject")
  x = isobject(fso)
  Document.write "Line 1 : " &  x  & "<br />"
  y = isobject(b)
  Document.write "Line 2 : " &  y  & "<br />"

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

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

IsNumeric

The IsNumeric Function is used to check whether or not the expression has a number subtype. It returns a boolean value. **IsObject** returns True if the expression contains an number subtype otherwise the expression returns False.

Syntax

```
IsNumeric(expression)
```

Example

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

  Dim var, chk
  var = 20
  chk = IsNumeric(var)
  Document.write "Line 1 : " &  chk  & "<br />"

  var = "3.1415935745"
  chk = IsNumeric(var)
  Document.write "Line 2 : " &  chk  & "<br />"

  var = "20 Chapter 23.123 VBScript"
  chk = IsNumeric(var)
  Document.write "Line 3 : " &  chk  & "<br />"

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

```
Line 1 : True
Line 2 : True
Line 3 : False
```

TypeName

The TypeName Function is used to return the variant subtype information of the variable.

Syntax

```
TypeName(varname)
```

The Typename function can return any of the following values.

- Byte - Byte Value
- Integer - Integer Value
- Long - Long Integer Value
- Single - Single-precision floating-point Value
- Double - Double-precision floating-point Value
- Currency - Currency Value
- Decimal - Decimal Value
- Date - Date or Time Value
- String - Character string Value
- Boolean - Boolean Value
- Empty - Uninitialized Value
- Null - No Valid Data
- Object - typename of Object
- Nothing - Object variable that doesn't yet refer to an object instance
- Error

Example

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

  Dim ArrVar(2), vartype
  NullVar = Null   ' Assign Null value.

  vartype = TypeName(3.1450)
  Document.write "Line 1 : " &  vartype  & "<br />"
  vartype = TypeName(432)
  Document.write "Line 2 : " &  vartype  & "<br />"
  vartype = TypeName("Microsoft")
  Document.write "Line 3 : " &  vartype  & "<br />"
  vartype = TypeName(NullVar)
  Document.write "Line 4 : " &  vartype  & "<br />"
  vartype = TypeName(ArrVar)
  Document.write "Line 5 : " &  vartype  & "<br />"

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

```
Line 1 : Double
Line 2 : Integer
Line 3 : String
Line 4 : Null
Line 5 : Variant()
```

Eval

The Eval Function executes an expression and returns the result either as a string or a number.

Syntax

```
Eval(expression)
```

The argument Expression can be a string expression or a number. If you pass to the Eval function a string that doesn't contain a numeric expression or a function name but only a simple text string, a run-time error occurs. For example, Eval " *VBScript* " results in an error.

Example

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

  Document.write Eval("10 + 10") & "<br />"
  Document.write Eval("101 = 200") & "<br />"
  Document.write Eval("5 * 3") & "<br />"

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

```
20
false
15
```

Execute

The Execute statement accepts argument that is a string expression containing one or more statements for execution.

Syntax

```
Execute(expression)
```

In VBScript, a = b can be interpreted two ways. It can be treated as an assignment statement where the value of x is assigned to y. It can also be interpreted as an expression that tests if a and b have the same value. If they do, result is True; if they are not, result is False. The Execute statement always uses the first interpretation while the Eval statement always uses the second.

Example

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

```
  Dim x
  x = "Global"
  y = "VBScript"
  Execute("x=y")
  msgbox x
  msgbox y

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

Save the file with .html extension upon executing the script in IE , the following result is displayed on the screen.

```
VBScript
VBScript
```

With..End With

The With statement allows us to perform a series of operation on a specified object without explicitly mentioning the object name over again and again.

Syntax

```
With (objectname)
   statement 1
   statement 2
   statement 3
   ...
   ...
   statement n
End With
```

Example

Upon Executing the below script, Winword is opened and the specified text is entered.

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

 Msg =  "Vbscript" & vbCrLf & "Programming"
 Set objWord = CreateObject("Word.Application")
 objWord.Visible = True

 ' Objects methods are accessed without requaliyfying the objects again.'
 With objWord
     .Documents.Add
     .Selection.TypeText Msg
     .Selection.WholeStory
 End With


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

Randomize

The Randomize statement initializes the random number generator which is helpful for the developers to generate a random number.

Syntax

```
Randomize [number]
```

Example

Upon Executing the below script, Winword is opened and the specified text is entered.

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

  Dim MyValue
  Randomize
  MyValue = Int((100 * Rnd) + 1)   ' Generate random value between 1 and 100.
  MsgBox MyValue

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

Save the above script as HTML and upon executing the script in IE, the following output is shown.

```
42
```