Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

SOGETI

VB Script Basic

Index

- VB Script Introduction
- VB Script Fundamentals
- Variables
- Operators
- Conditional Statements and Loops
- User Defined Functions
- Built in Functions

VB Script Introduction

What is VBScript

- VBScript is a scripting language

- A scripting language is a lightweight programming language

- VBScript, for the most part, is case insensitive. It has a very simple syntax, easy to learn and to implement.

- VBScript is an object-based scripting language and not an Object-Oriented Programming language like Java .

- VBScript is a light version of Microsoft's programming language Visual Basic

VB Script Introduction

VB Script – Version History and Uses

VBScript was introduced by Microsoft way back in 1996 and the 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,

• VBScript is used as a scripting language in one of the popular Automation testing tools – Unified Functional Testing abbreviated as UFT

• 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 web pages which uses VBScript or Java Script.

• VBScript is used for Client side scripting in Microsoft Internet Explorer.

Disadvantages

•VB script 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.

VB Script Fundamentals : Variables

VBScript Variables

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**.

Purpose of Variable:

a) Comparing values Example:

```
for e. g :
Dim x,y,a
x=100
y=100
a=x=y
Msgbox a  'It returns True
```

b) Holding Program Result

for e.g:

```
Cost=  Tickets  *  Price
```

c) Passing parameters – can be passed as parameters

for e.g. :

```
Function abc ( intLength , intBreadth)
```

d) To store data that returned by functions

for e.g:

```
myDate=Now ' It returns current date & time
```

e) To hold data

for e.g:

```
myName="testing"
```

VB Script Fundamentals : Variables

Rules for Declaring Variables

- Variable Name must begin with an alphabet. For e.g Dim X,x9, intX,strX etc
- Variable names cannot exceed 255 characters.
- Variables Should NOT contain a period(.) for e.g. Dim X.Y, a.2,a-b,7a,7b wrong declaration
- 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.

1: Single variable declaration
   Dim Var
2: Two or more declarations are separated by comma(,)
   Dim Variable1,Variable2

VB Script Fundamentals : Variables

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.

for e.g. Dim x = 10, Y= 200

•The String values should be enclosed within double quotes(")

Dim strVar strVar = “Hello”

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

Dim Time1 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.

1. Dim 2. Public 3. Private

VB Script Fundamentals : Variables

1. **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.

```
Dim  temp1  ‘  declared at Script level
Dim  temp2 ‘  declared at Script level
Call add()
Function add()
     temp1 = 2
     temp2 = 4
     Dim temp3
     temp3 = temp1+ temp2
     Msgbox  temp3     ‘declared at procedure level
 End Function

Msgbox  temp1      ‘Displays 2 as temp1 is declared at Script level
Msgbox  temp2      'Displays 4 as temp2 is declared at Script level
Msgbox  temp3      ‘temp3 has No Scope outside the procedure. Prints Empty
```

VB Script Fundamentals : Variables

2. 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".

```
Dim  temp1      ‘  declared at Script level
Dim  temp2      ‘  declared at Script level
Public temp3    ‘ declared at Public level

Call add()
Function add()
     temp1 = 2
     temp2 = 4
     temp3 = temp1+ temp2
     Msgbox  temp3     ‘declared at procedure level
 End Function

Msgbox  temp1      ‘Displays 2 as temp1 is declared at Script level
Msgbox  temp2      'Displays 4 as temp2 is declared at Script level
Msgbox  temp3      ‘Displays 6 as temp3 is declared as Public
```

VB Script Fundamentals : Variables

3. 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".

```
Dim  temp1     ‘  declared at Script level
Dim  temp2     ‘  declared at Script level
Private temp3  ‘  declared at Private level

Call add()
Function add()
     temp1 = 2
     temp2 = 4
     temp3 = temp1+ temp2
     Msgbox  temp3     ‘declared at procedure level
 End Function

Msgbox  temp1      ‘Displays 2 as temp1 is declared at Script level
Msgbox  temp2      'Displays 4 as temp2 is declared at Script level
Msgbox  temp3      ‘Displays 6 but temp3 is available only for this script.
```

VB Script Fundamentals : Variables

Option Explicit ‘ Forces explicit declaration of all variables in a script.

```
Dim strUsername
 strUsername  = “Sam”
```

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

Declaring 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.
Syntax:
[Public | Private] Const Constant Name = Value

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

```
Const myString = "VBScript"
Const myDate = #01/01/2050#
Msgbox myString
Msgbox myDate
```

```
Dim intRadius
intRadius = 20
const pi=3.14
pi = pi*pi
Area = pi * intRadius * intRadius
Msgbox Area
'pi value can not be changed. throws error
```

VB Script Fundamentals - Operators

What is an operator?

Operators are used for performing mathematical, comparison and logical operations. for e.g. 4 + 5 = 9. Here, 4 and 5 are called operands and + is called operator. VBScript language supports following types of operators:

1. Arithmetic Operators
2. Comparison Operators
3. Logical (or Relational) Operators
4. Concatenation Operators

VB Script Fundamentals - Operators

The Arithmetic Operators

Operator	Description	Example	Script Example
+	Adds two operand	A + B = 15	Dim a,b,c a=10 ,b=2 c= a+b Msgbox d
-	Subtracts second operand from the first	A – B = 8	c= a-b Msgbox c '8
*	Multiply both operands	A * B = 20	c=a*b Msgbox c '20
/	Divide numerator by de numerator	B / A = 0.2	C=b/a Msgbox c '0.2
%	Modulus Operator and remainder of after an integer division	A MOD B = 0	c=a mod b Msgbox c '10
^	Exponentiation Operator	B ^ A = 1024	C=b^a Msgbox c '1024

VB Script Fundamentals - Operators

The Comparison Operators

Operator	Description	Example	Script Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is False.	Dim x,y,z x=10 y=20 z=x=y Msgbox z '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.	x=10 y=20 z=x<>y Msgbox z '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.	x=10 y=20 z=x>y Msgbox z '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.	x=10 y=20 z=x<y Msgbox z '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.	x=10 y=20 z=x>=y Msgbox z '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.	x=10 y=20 z=x<=y Msgbox z 'True

VB Script Fundamentals - Operators

The Logical Operators

Operator	Description	Example	Script Example
AND	If both the conditions are True then Expression becomes true.	a<>0 AND b<>0 is False.	Dim shoeSize shoeSize = 10 If shoeSize >= 10 AND shoeSize <= 12 Then 'Some code Endlf
OR	If any of the two conditions are True then condition becomes true.	a<>0 OR b<>0 is true.	Dim shoeSize shoeSize = 10 If shoeSize = 10 OR shoeSize = 12 Then 'Some code Endlf
NOT	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.	Dim shoeSize shoeSize = 10 If NOT (shoeSize <> 10 OR shoeSize <> 12)Then 'Some code Endlf
XOR	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.	Dim shoeSize shoeSize = 10 If (shoeSize <> 10 XOR shoeSize <> 12)Then 'Some code Endlf

VB Script Fundamentals - Operators

The Concatenation Operators

Operator	Description	Example	Script Example
+	Adds two Values as Variable Values are Numeric	A = 10 B = 5 A + B = 15	Dim a,b,c a=10 b=5 c=a+b Msgbox c '15 (if both are numeric, then it adds) a="10" b=5 c=a+b Msgbox c '15 (one is string another numeric, then it adds) a="10" b="5" c=a+b Msgbox c '105 (if both are strings, then it concatenates) a="hydera" b="bad" c=a+b Msgbox c 'hyderabad a="Test" b=2 c=a+b Msgbox c 'error
&	Concatenates two Values	A & B = 105	a=10 b=5 c= a & b Msgbox c '105 a="10" b=5 c= a &b Msgbox c '105 a="10" b="5" c=a &b Msgbox c '105 a="hydera" b="bad" c= a & b Msgbox c 'hyderabad

VB Script Fundamentals – Conditional Statements

Statement	Description	Example

if statement
 	
Syntax :-
If(boolean_expression)
Then
  Statement 1
  .....
  Statement n
End If
	An if statement consists of a boolean expression followed by one or more statements.	Dim a : a = 40
		Dim b : b = 20
		
		If a > b Then
		  msgbox "a is Greater than b"
		End If

if..else statement
 	
Syntax :-
If(boolean_expression)
Then
  Statement 1
  .....
  Statement n
Else
  Statement 1
  ....
  Statement n
End If
	An if else statement consists of a boolean expression followed by one or more statements. If the condition is True, the statements under If statements are executed. If the condition is false, Else part of the script is Executed	Dim a : a = 15
		Dim b : b = 35
		
		If a > b Then
		  msgbox "a is Greater"
		Else
		  msgbox "b is Greater"
		End If

VB Script Fundamentals – Conditional Statements

Statement	Description	Example

if...elseif..else statement
 	
Syntax
 	
If(boolean_expression) Then
  Statement
ElseIf (boolean_expression) Then
  Statement
ElseIf (boolean_expression) Then
  Statement
Else
  Statement
End If
	An if statement followed by one or more ElseIfStatements, that consists of boolean expressions and then followed by an optional else statement, which executes when all the condition becomes false.	Dim a
		a = -5
		
		If a > 0 Then
		  msgbox "a is a POSITIVE Number"
		ElseIf a < 0 Then
		  msgbox "a is a NEGATIVE Number"
		Else
		  msgbox "a is EQUAL than ZERO"
		End If

VB Script Fundamentals – Conditional Statements

Statement	Description	Example

nested if statements
 	
Syntax
 	
If(boolean_expression) Then
Statement
If(boolean_expression) Then
Statement
ElseIf (boolean_expression) Then
Statement
Else
Statement
End If Else
Statement
End If
	An if or elseif statement inside another if or else if statement(s).	Dim a
		a = 10
		
		If a > 0 Then
		msgbox "The Number is a POSITIVE Number"
		If a = 1 Then
		msgbox "The Number is Neither Prime NOR Composite"
		Elseif a = 2 Then
		msgbox "The Number is the Only Even Prime Number"
		Elseif a = 3 Then
		msgbox "The Number is the Least Odd Prime Number"
		Else
		msgbox "The Number is NOT 0,1,2 or 3"
		End If
		ElseIf a < 0 Then
		msgbox "The Number is a NEGATIVE Number"
		Else
		msgbox "The Number is ZERO"
		End If

VB Script Fundamentals – Conditional Statements

Statement	Description	Example
switch statement Syntax Select Case expression Case expressionlist1 statement Case expressionlist2 statement Case expressionlistn statement Case else default statement End Select	A switch statement allows a variable to be tested for equality against a list of values.	Dim MyVar MyVar = 1 Select case MyVar case 1 msgbox"The Number is the Least Composite Number" case 2 msgbox"The Number is the only Even Prime Number" case 3 msgbox"The Number is the Least Odd Prime Number" case else msgbox"Unknown Number" End Select

VB Script Fundamentals –Loops

There may be a situation when you need to execute a block of code several number of times

Looping allows you to run a group of statements repeatedly. Some loops repeat statements until a condition is False; others repeat statements until a condition is True. There are also loops that repeat statements a specific number of times.

The following looping statements are available in VBScript:

Do...Loop: Loops while or until a condition is True.

While...Wend: Loops while a condition is True.

For...Next: Uses a counter to run statements a specified number of times.

For Each...Next: Repeats a group of statements for each item in a collection or each element of an array

VB Script Fundamentals – Loops

Loop Type	Description	Example
Do...Loop **Syntax** **Do** [{**While** | **Until**} condition] [statements] [**Exit Do**] [statements] **Loop** ' or use this syntax **Do** [statements] [**Exit Do**] [statements] **Loop** [{**While** | **Until**} condition]	Repeats a block of statements while a condition is True or until a condition becomes True. Arguments *condition* Numeric or string expression that is **True** or **False**. If condition is **Null**, condition is treated as **False**. *statements* One or more statements that are repeated while or until condition is **True**.	**e.g. 1** **Do While** i < 5 i = i + 1 Msgbox("The value of i is : " & i) **Loop** **e.g 2** i=10 **Do** i = i + 1 Msgbox("The value of i is : " & i) **Loop While** i<3 Condition is false. Hence loop is executed once.

VB Script Fundamentals – Loops

Loop Type	Description	Example
While...Wend Syntax **While** condition Version [statements] **Wend**	Executes a series of statements as long as a given condition is True. Arguments *condition* Numeric or string expression that evaluates to **True** or **False**. If condition is Null, condition is treated as **False**. *statements* One or more statements executed while condition is **True**.	Dim Counter Counter = 0 **While** Counter < 20 Counter = Counter + 1 Msgbox Counter **Wend** ' End While loop when Counter > 19.

VB Script Fundamentals – Loops

Loop Type	Description	Example
For...Next Syntax **For** counter **=** start **To** end [**Step** step] [statements] [**Exit For**] [statements] **Next**	Executes a series of statements as long as a given condition is True.	Dim a : a=10 **For** i=0 **To** a Step 2 Msgbox("The value is i is : " & i) **Next**

Arguments

counter

Numeric variable used as a loop counter. The variable can't be an array element or an element of a user-defined type.

start

Initial value of counter.

end

Final value of counter.

step

Amount counter is changed each time through the loop. If not specified, step defaults to one.

statements

One or more statements between For and Next that are executed the specified number of times.

VB Script Fundamentals – Loops

Loop Type	Description	Example
For Each...Next Syntax **For Each** element **In** group [statements] [**Exit For**] [statements] **Next** [element]	Repeats a group of statements for each element in an array or collection.	' is an array strNumber=Array("One","Two","Three") Dim test 'iterating using For each loop. For each item in strNumbers test = test & item & vbnewline Next msgbox test

Arguments

element
Variable used to iterate through the elements of the collection or array. For collections, element can only be a Variant variable, a generic Object variable, or any specific Automation object variable. For arrays, element can only be a Variant variable.
group
Name of an object collection or array.
statements
One or more statements that are executed on each item in group.

VB Script Fundamentals – Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all the remaining statements in the loop are NOT executed.

Control Statement	Description	Example

Exit For statement
 	
Syntax
 	
Exit For
	Terminates the **For loop** statement and transfers execution to the statement immediately following the loop	Dim a : a=10
		For i=0 to a Step 2
		msgbox("The value is i is : " & i)
		If i=4 Then
		i=i*10
		msgbox("The value is i is : " & i)
		Exit For 'Exited when i=4
		End If
		Next

Exit Do statement
 	
Syntax
 	
Exit Do
	Terminates the **Do While** statement and transfers execution to the statement immediately following the loop	i = 0
		Do While i <= 100
		If i > 10 Then
		Exit Do ' Loop Exits if i>10
		End If
		msgbox("The Value of i is : " &i)
		i = i + 2
		Loop

VB Script Fundamentals - User Defined Functions

What is a Function?

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing same code over and over again. This will enable programmers to divide a big program into a number of small and manageable functions.

Function Definition

A Function procedure is a series of VBScript statements enclosed by the **Function** and **End Function** statements.

Syntax	**For e.g. 1**	**For e.g. 2**
Function function name(parameter-list) statement 1 statement 2 ..... statement n **End Function**	**Function** sayHello() msgbox("Hello there") **End Function** **Call sayHello()**	**Function sayHello(name, age)** **Msgbox (name & " is " & age & " years old.")** **End Function** **Call sayHello("Hello", 9)**

VB Script Fundamentals - User Defined Functions

What is a Sub Procedures?

•Sub Procedures are similar to functions but there are few differences.

•Sub procedures DONOT Return a value while functions may or may not return a value.

•Sub procedures Can be called without call keyword.

•Sub procedures are always enclosed within Sub and End Sub statements.

Syntax	*For e.g. 1*	*For e.g. 2*
Sub procedure name(parameter-list) statement 1 statement 2 ..... statement n **End Sub**	**Sub sayHello()** **msgbox("Hello there")** **End Sub** **Call sayHello()**	**Sub sayHello(name, age)** **Msgbox (name & " is " & age & " years old.")** **End Sub** **Call sayHello("Hello", 9)**

VB Script Fundamentals – Built in Functions

Function Name	Example
UCase(*String*) Returns a string that has been converted to uppercase. LCase(*String*) Returns a string that has been converted to lowercase.	Dim txt txt="Have a nice day!" msgbox(UCase(txt)) msgbox(LCase(txt))
Rnd function returns a random number. The number is always less than 1 but greater or equal to 0	Dim max,min max=100 min=1 Randomize Msgbox Int((max-min+1)*Rnd+min)
Left Function Returns a specified number of characters of a given string from left side	Example: Dim val,x val="Hyderabad" x=Left(val,3) Msgbox x ' Output: Hyd
Mid function Returns a specified number of characters of a given string	Example: Dim val,x val="Hyderabad" x=Mid(Val,5,3) Msgbox x ' Output: rab

VB Script Fundamentals – Built in Functions

Function Name	Example
StrReverse function returns reverse value of a string	Example: Dim val,x val="Hyderabad" x=StrReverse(val) Msgbox x 'Output dabaredyH
StrComp function It compares two string (Binary and textual) if a) Both are equal, returns 0(zero) b) String 1 greater than string 2, returns 1(one) c) String 2 greater than string 1, returns -1	Example: Dim str1,str2,x str1="India“ str2="India" x=StrComp(str1,str2,1) Msgbox x ‘ Output 0
IsArray function returns a Boolean value that indicates whether a specified variable is an array UBound function returns the largest subscript for the indicated dimension of an array.	Example days= Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat") msgbox LBound(days) ‘ Returns 0 Msgbox UBound(days) ‘ Returns 6
Len Function Returns the number of characters in a string or the number of bytes required to store a variable. Replace It replace a sub string with given value (another sub string)	Example- Dim Mystring mystring=Len("Test Msgbox") Msgbox mystring Example- mystring=Replace("kb script", "k","v") Msgbox mystring

Thank You ☺