

VBScript

Lesson 1: Introduction to
VBScript

Lesson Objectives

- What is VBScript?
- Understand purpose of VBScript
- Advantages and disadvantages
- Sample Code
- Basics of VBScript



Introduction to VBScript

- What is VB Script
- VBScript in Client Side and Server Side
- Features of VBScript
- Advantages of VBScript
- Disadvantages of VBScript
- A Sample Code
- Basics of VBScript
 - Where to Insert Scripts in an HTML page?
 - Formatting



Copyright © Capgemini 2015. All Rights Reserved 3

Add the notes here.

What is VBScript?

- VBScript is short for Visual Basic Script.
- Visual Basic, as you may know, is a powerful programming language developed by Microsoft. VBScript is a scripting language, a subset of Visual Basic, by Microsoft, intended to be used with Microsoft products.
- VBScript is an interpreted programming language that can be embedded into an HTML web page or used in server side scripting.



Copyright © Capgemini 2015. All Rights Reserved. 4

Interpreted programming languages tend to be simpler to program but slower to execute in general. Each time a program is run, it has to be interpreted (interrogated) line by line, based on the flow of execution (you will see later how branches and loops affect the flow of execution).

Compiled programming languages have a more complex syntax, and require more strict programming practices. With a compiled programming language, you first write the source code, then you feed it to a compiler (a special computer program), which produces an executable binary program. On the Windows platform, the output of the compiler usually ends in the ".exe" file extension. The program that comes out of the compilation process tends to be platform (operating system) specific. The key benefit for the programmer is that no other programmer can look at the source code once it is compiled. The other key factor is that the language used to write the source code becomes irrelevant once it has been compiled. Visual Basic is a compiled language, whereas VBScript is an interpreted language.

VBScript in Client Side and Server Side

Client Side Scripting

- VBScript code is executed/interpreted when an event is triggered. When the code is executed it is interpreted one line at a time. There are a number of events that will trigger the execution of a VBScript, like clicking on a form button, or the completion of a web page loading.

Server Side Scripting

- Server-side scripts are completely processed by the servers instead of clients. When clients request a page containing server-side scripts, the applicable server processes the scripts and returns an HTML page to the client
- VBScript is the default scripting language for ASP(Application Service Provider). When the web server loads an .asp page from the disk into memory, it automatically knows to interpret the code in this document. Once the code has been interpreted, the resulting HTML page is sent to the browser (client) making the request.



Copyright © Capgemini 2015. All Rights Reserved 5

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).



Copyright © Capgemini 2015. All Rights Reserved.

6

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

Advantages of VBScript

- VBScript has a good platform coverage. It can be run in many environments. Currently there are VBScript script engines for the 32-bit Windows API, 16-bit Windows API, and the Macintosh.
- VBScript is used as a scripting language in one of the popular Automation testing tools – Quick Test Professional abbreviated as QTP.
- VBScript is used for Client side scripting in Microsoft Internet Explorer.
- The ability to implement VBScript in our own applications. The VBScript source implementation can be licensed from Microsoft, completely free of charge, for use in any products and applications.



Copyright © Capgemini 2015. All Rights Reserved

7

Disadvantages of VBScript

- VBScript is supported 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.



Copyright © Capgemini 2015. All Rights Reserved 8

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

A Sample Code

- A VBScript to print out "Hello World!"

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



Copyright © Capgemini 2015. All Rights Reserved 9

We write the VBScript code in an HTML file and save the same with .htm or .html extension.

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, above code will display following result:

Hello World!

Basics of VBScript - Where to Insert Scripts in an HTML page?

- Where to Insert Scripts in an HTML page?
We have the liberty to insert Script code, in an HTML page, at any / all of the following locations:
 - Between the header tags i.e. <head> and </head>.
 - Within the HTML document's body i.e. between the <body> and </body>
 - In both header and the body
- You can have as many scripts on a webpage as you want, this includes both scripts in the head section and the body section of a webpage.



Copyright © Capgemini 2015. All Rights Reserved 10

Example for Script Within the header tags :

A script placed in the head section of a webpage will be executed when it is called, or when certain events are triggered such as the clicking of a button or when a form is submitted

```
<html>
<head>
<script type="text/vbscript">
    document.write("This is a script")
</script>
<title>VBScript script in the head section</title>
</head>
<body>
</body>
</html>
```

Example for Script Within the body tags :

A script placed in the body section of a webpage will be executed when the page loads.
A script placed in the body section of a webpage generates the content of the page.



Copyright © Capgemini 2015. All Rights Reserved 11

```
<html>
<head>
<title>VBScript script in the body section</title>
</head>
<body>
<script type="text/vbscript">
    document.write("This is a script")
</script>
</body>
</html>
```

Example for Script in both head and body :

```
<html>
<head>
<script type="text/vbscript">
    document.write("This is a script")
</script>
<title>VBScript script in the head and body sections</title>
</head>
<body>
<script type="text/vbscript">
    document.write("This is a script")
</script>
</body>
</html>
```

Basics of VBScript-Formatting

- White space ,tabs and newlines
 - VBScript ignores spaces, tabs and newlines that appear within VBScript programs.
- 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 .
- Multiple Line Syntax
 - If a statement must span more than one physical line, place an underscore at the end of the line. The underscore acts as a *line continuation character*.
- Comment statements
 - In VBScript, comment statements are also called remark statements. There are two ways to indicate a remark statement.
 - The most common is to use a single quotation mark (').
 - You can also use the Rem keyword.



Copyright © Capgemini 2015. All Rights Reserved. 12

Examples:

1. Single Line Syntax

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

2. Multiple Line Syntax

```
<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>
```

3. Comments by use a single quotation mark

```
<script type="text/vbscript">
'this is a single line comment
'this is another single line comment
document.write("Here is some text")
</script>
```

4. Comments By Using the Rem keyword:

```
Sub cmdButton1_Click
Rem Comments describing the event procedure should go here
Rem and probably here too
End Sub
```

Lab

- Lab 1



Copyright © Capgemini 2015. All Rights Reserved. 13

Lesson Summary

- VBScript is an interpreted programming language that can be embedded into an HTML web page or used in server side scripting.
- Advantages and disadvantages of scripting.
- Formatting and basics in VBScript.



Copyright © Capgemini 2015. All Rights Reserved. 14.

Review - Questions

- Question 1: Comments in VB Script can be created using
 - Option 1: <!-- -->
 - Option 2: "
 - Option 3: '
 - Option 4: rem
- Question 2: We can insert VBScript tags only in the header section of html page. State True or False
- Question 3: Only _____ browser supports VBScript.



VBScript

Lesson 2: Variables ,Constants
and Operators in VBScript

Lesson Objectives

- Variables
- Constants
- Operators
- Type conversion



Copyright © Capgemini 2015. All Rights Reserved | 2

Variables ,Constants and Operators in VBScript

- Variables
 - Scope of variables
- Constants
 - Intrinsic constants
- Operators
 - Operator Precedence
- Type Conversions



Copyright © Capgemini 2015. All Rights Reserved | 3

Add the notes here.

Variables

- In VBScript all variables are of type variant, they can store any type of value
- Variable names in VBScript must follow these rules:
 - Must begin with an alphabetic character
 - Include any combination of letters, numbers, and underscores
 - Cannot contain an embedded period(.)
 - Must not exceed 255 characters
 - Must be unique in the scope in which it is declared
- VBScript is not case-sensitive



Copyright © Capgemini 2015. All Rights Reserved. 4

Variables

Variables in VBScript hold information (values). Whenever you use a variable, VBScript sets up an area in the computer's memory to store the information.

To VBScript, MyName and myname are the same variable name.

Variables names

- Below shows a list of acceptable and not acceptable variables.

Examples	Acceptability
OnFirst	Acceptable
1stOn	Not acceptable (first character is not a letter)
First.1	Not acceptable (uses a period)
ThisIsKindOfLongButIsTheoreticallyOK	Acceptable (less than 255 characters but probably too cumbersome for a realistic program)



Copyright © Capgemini 2015. All Rights Reserved 5

All characters in a variable name are significant. Base is a different variable from Base1, and both are different from Base_1.

You can't use names reserved by VBScript for variable names; for example, Sub is not acceptable as a variable name.

However, you can embed reserved words within a variable's name. For example, Substitute is a perfectly acceptable variable name.

Declaring Variables

- Declaring of variables can also be considered as creating them.
- Variables in VBScript can be declared anywhere in the script, generally done using the keyword 'Dim'.
- They can be declared with or without the 'Dim'.
- VBScript has only ONE fundamental data type, Variant
- 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.



Copyright © Capgemini 2015. All Rights Reserved

6

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

empname="Sandra"

Now you have also created a variable. The name of the variable is "empname". 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 "empname" variable to "empnime", the script will automatically create a new variable called "empnime".

To prevent your script from doing this, you can use the Option Explicit statement. This statement forces you to declare all your variables.

Option Explicit

- To ensure that, your VBScript programs declare its variables before you can use them, use option explicit directive

```
<SCRIPT LANGUAGE="VBSCRIPT">
    Option Explicit
<SCRIPT>
```

For e.g.:

```
'Since Option Explicit is mentioned the variable needs to be declared
Option Explicit
Dim empname
empname="Sandra"
```



Copyright © Capgemini 2015. All Rights Reserved 7

Explicit

After VBScript processes an Option Explicit statement, it will no longer allow you to use a variable unless you declare it first. If you try to use a variable without declaring it, an error message will appear. Finally, the first time you use a variable, VBScript temporarily assigns it the default value "empty." This basically means that the variable doesn't contain a value. The "empty" value disappears the moment you assign a value to the variable.

You shouldn't hesitate to assign initial values to your variables. Otherwise, you risk creating a breeding ground for hard-to-find bugs. It is therefore quite common to use the first few statements in an event procedure to initialize the variables.

Declaration

Assigning Values to the Variables

- While assigning the values to the variables following rules to be kept in mind:
 - The numeric values should be enclosed in single quotes(')
 - The String values should be enclosed within double quotes("")
 - Date and Time variables should be enclosed within hash symbol(#)



Copyright © Capgemini 2015. All Rights Reserved



Examples :

```
' Below Example, The value 20 is assigned to the variable.  
Number1 = 20 '
```

```
A String Value 'Come' is assigned to the variable StrValue.  
StrValue = "Come"
```

```
' The date 01/01/2020 is assigned to the variable DTToday.  
DToday = #01/03/2010#
```

```
' A Specific Time Stamp is assigned to a variable Time1  
Time1 = #12:20:24 PM#
```

Scope of Variables-Local

- A variable's scope is determined by where you declare it
- Variable declared within a procedure is accessible only within that procedure
- It has local scope and is called a procedure - level variable



Copyright © Capgemini 2015. All Rights Reserved. 9

Sharing values within the procedure

When programmers discuss the availability of a variable used in one part of the program to the other parts of the program, they refer to the scope of variables. An event procedure will not normally have access to the value of a variable changed in another event procedure. As always, it is not a good programming practice to rely on defaults. If you want to be sure that a variable is local within an event procedure, use the Dim statement inside the event procedure to declare all its variables.

Scope of Variables –Global

- Variable declared outside a procedure, will be recognizable to all the procedures in your script.
- This is a script - level variable, and it has script-level scope.



Copyright © Capgemini 2015. All Rights Reserved 10

Sharing values across procedures

Occasionally you will want the value of a variable to be available to all the VBScript code for your Web page. For example, if an application is designed to perform a calculation involving a single interest rate at a time, that rate should be available to all the procedures for the page. Variables that are shared in this way are called script-level variables. Just as with the Option Explicit statement, you put the declaration statements for script-level variables outside any event procedures.

Example

- Script and Procedure level variables

```
<html>
<body>
<script language="vbscript" type="text/vbscript">
Dim Num1
Dim Num2
Call prod()
Function prod()
    Num1 = 2
    Num2 = 5
    Dim Num3
    Num3 = Num1*Num2
    MsgBox Num3      'Displays 10, the product of two values.
End Function
MsgBox Num1      ' Displays 2 as Num1 is declared at Script level
MsgBox Num2      ' Displays 5 as Num2 is declared at Script level
MsgBox Num3      ' Num3 has No Scope outside the procedure. Prints Empty
</script> </body> </html>
```



Copyright © Capgemini 2015. All Rights Reserved 11

In the above example:

the values of Num1 and Num2 are declared at script level while Num3 is declared at procedure level.

Variables declared using different Keywords

- Scope of the variables differ based on the keyword used while declaring it.
- Variables declared using "Public" Keyword are available to all the procedures across all the associated scripts.
- Variables that are declared as "Private" have scope only within that script in which they are declared.
- 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.



Copyright © Capgemini 2015. All Rights Reserved 12

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".

Demo

- Using different declaration of Variables
 - vardeclar_private
 - vardeclar_public



Copyright © Capgemini 2015. All Rights Reserved 13

Add the notes here.

Constants

- Constants are declared the same way the variables are declared
- Use the convention that the names of the constants are always uppercase
- If a user tries to change a Constant Value, the Script execution ends up with an error
- A constant inside an event procedure, is visible only in that event procedure
- A constant right after the HTML comment tag all your VBScript code can see it and use it

```
Const PI = 3.14159  
Const USER_NAME = "B. Smith"  
Const SCRIPT_LANGUAGE = "VBScript"  
Const CUTOFFDATE = #6-1-97#
```



Copyright © Capgemini 2015. All Rights Reserved 14

Constants

The keyword for telling VBScript that something is a constant is simply const, as shown in the following example:

```
Const PI = 3.14159
```

You might also set up string constants using the same keyword:

```
Const USER_NAME = "B. Smith"  
Const SCRIPT_LANGUAGE = "VBScript"
```

Intrinsic constants

- VBScript comes with lots of useful intrinsic constants
- Some of the intrinsic constants:

Constant	Value	Description
vbSunday	1	Sunday
vbMonday	2	Monday
vbYellow	&hFFFF	Yellow
vbRed	&hFF	Red
vbGreen	&hFF00	Green
vbCrLf	Chr(13) & Chr(10)	Carriage return–linefeed combination



Copyright © Capgemini 2015. All Rights Reserved 15

VBScript comes with lots of useful built-in constants. These are constants such as vbRed and vbYellow for colors, and vbSunday and vbMonday for date functions. VBScript does not follow the convention that built-in constants are uppercase.

Operators

- VBScript supports below operators:
 - Arithmetic Operators
 - Comparison Operators
 - Logical (or Relational) Operators
 - Concatenation Operators



Copyright © Capgemini 2015. All Rights Reserved 16

Arithmetic Operators

- If A =5 and B=10.

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



Copyright © Capgemini 2015. All Rights Reserved 17

The ordinary division symbol (/) gives you a value that has a decimal point. The integer division symbol (\) throws away the remainder in order to give you an integer.

For example, $7\backslash 3 = 2$.

Since ordinary division always produces a number with a decimal point, use integer division or the Mod operator if you really want to work with integers. The Mod operator gives you the remainder after integer division.

For example, $7 \text{ Mod } 3 = 1$.

When one integer perfectly divides another, there is no remainder, so the Mod operator gives 0: $8 \text{ Mod } 4 = 0$.

The term for a combination of numbers, variables, and operators from which VBScript can extract a value is numeric expression

Comparison Operators

- A=10 and B=20

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



Copyright © Capgemini 2015. All Rights Reserved 18

As an example, suppose you want to prevent a "divide-by-zero" error when a user enters data in a text box. Use a fragment like this:

```
Do  
Number = InputBox("Please enter a nonzero number.")  
Loop Until Number <> 0
```

Logical Operators:

- A=10 and B=0

Operator	Description	Example
AND	Called Logical AND operator. If both the conditions are True then Expression becomes true.	$a <> 0 \text{ AND } b <> 0$ is False.
OR	Called Logical OR Operator. If any of the two conditions are True then condition becomes true.	$a <> 0 \text{ OR } b <> 0$ 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.	$\text{NOT}(a <> 0 \text{ OR } b <> 0)$ 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.	$(a <> 0 \text{ XOR } b <> 0)$ is false.



Concatenation Operators

- A=5 and B=10 (for 1st 2 rows) and A="Microsoft" and B="VBScript" (for last 2 rows)

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



Copyright © Capgemini 2015. All Rights Reserved 20

Eqv operator

- The Eqv operator is used to perform a logical comparison on two expressions (i.e., are the two expressions identical),
 - For example:

```
If (X=True And Y=True) Or (X=False And Y=False)
```

- is the same as

```
If (X Eqv Y) Then
```



Copyright © Capgemini 2015. All Rights Reserved 21

Eqv Operator –Examples

- The truth table for Eqv is as follows:

Expr1	Expr2	Expr1 EQV Expr2
True	True	True
True	False	False
False	True	False
False	False	True



Copyright © Capgemini 2015. All Rights Reserved 22

Operator Precedence

- When several operators occur in an expression, each part is evaluated in a predetermined order called operator precedence. When expressions contain operators from more than one category-
 - Arithmetic operators are evaluated first
 - Comparison operators are evaluated next
 - Logical operators are evaluated last
- Comparison operators all have equal precedence; that is, they are evaluated in the left-to-right order in which they appear
- Arithmetic operators are evaluated in the following order:
 - Exponentiation
 - Multiplication
 - Division
 - Modulus
 - Addition and subtraction
 - and finally concatenation



Copyright © Capgemini 2015. All Rights Reserved 23

For example, if you try to calculate the expression $a = 5-4*3/5^6$, what you expect as the result?

The result will be 4.999232

How? The exponentiation comes first, then come multiplication and division and finally comes subtraction.

So the above expression gets calculated like this: $5-4*3/(5^6) \rightarrow 5-(4*3)/30 \rightarrow 5-(12/30) \rightarrow 5-.04 \rightarrow 4.999232$

Operator Precedence (continued)

- Logical operators are evaluated in the following order:
 - NOT
 - AND
 - OR
 - XOR
- Parentheses are used to change the normal order of precedence to the way it is needed. Within parentheses normal operator precedence is maintained.



Copyright © Capgemini 2015. All Rights Reserved 24

In the earlier example Suppose, you want to calculate $5-4$ first, then you should write the expression as $a = (5-4)*3/5^6$.

Now you get the value of a as $1*3/5^6 \rightarrow 1*3/30 \rightarrow 3/30 \rightarrow 0.1$

Arithmetic on date variables

- When an integer is being added or subtracted to/from data variables it is actually the no of days that gets added or subtracted
- Adding a fraction changes the time within a day
- Function Now is used to retrieve the current date and time
- The following example displays today's date and time and then displays the date and time 10,000 days ago:

```
Dim Today, LongTimeAgo  
Today = Now  
MsgBox(Today)  
LongTimeAgo = Today - 10000  
MsgBox(LongTimeAgo)
```



Copyright © Capgemini 2015. All Rights Reserved 25

Arithmetic on date variables

VBScript makes it easy to do calculations with date variables. If you add or subtract an integer, you add or subtract that many days.

Adding a fraction changes the time within a day. You can use the function Now to retrieve the current date and time.

Demo

- Operator precedence
 - operator_precedence
- Arithmetic on date variable
 - datevariable_arithmetic



Copyright © Capgemini 2015. All Rights Reserved 26

Add the notes here.

Type conversions

- The Syntax used to carry on type conversions in VBScript is:
`Variable_name = Conversion_function_name(expression)`
- Some of the Conversion function names are:
 1. Cint : Converts an expression to an Integer, rounding if necessary
 2. CLng : Converts an expression to a long integer, rounding if necessary
 3. CSng : Converts an expression to a single-precision number
 4. CDbl : Converts an expression to a double-precision number
 5. Ccur : Converts an expression to a variable of type Currency
 6. CStr : Converts an expression to a string
 7. Cbool : Converts an expression to a Boolean value (True or False)
 8. Cbyte : Converts an expression to a variable of type Byte
 9. Cdate : Converts a string to a variable of type Date



Copyright © Capgemini 2015. All Rights Reserved 27

For example, the following line ensures that VBScript regards the zip code as a text string and not as a number:

```
ZipCode = CStr(22203)
```

Demo

- Type conversions
 - typeconversion



Copyright © Capgemini 2015. All Rights Reserved 28

Add the notes here.

Lab

▪ Lab 2



Copyright © Capgemini 2015. All Rights Reserved 29

Lesson Summary

- Variables can be declared as DIM or PUBLIC or PRIVATE
- Option Explicit ensures that the variable is always declared before it is used
- Type conversion functions help in conversion from one data type to other



Copyright © Capgemini 2015. All Rights Reserved 30

Review - Questions

- Question1: Intrinsic Constants should always be in upper case. State True or False.
- Question 2: _____ Operators are operated at the end.
- Question 3: _____ is used to retrieve the current date and time.



Copyright © Capgemini 2015. All Rights Reserved 31

VBScript

Lesson 3: Using Conditional
Statements and Loops in
VBScript

Lesson Objectives

- Conditional Statements
- Loops



Copyright © Capgemini 2015. All Rights Reserved | 2

Using Conditional Statements and Loops in VBScript

- Conditional Statements:

- If..Then ,
- If ...Then.. Else..,
- If ...Then.. Elseif..,
- Select Case

- Loops

- Do While Loop
- Do Until Loop
- For Next
- For-Step-Next
- For Each Next



Copyright © Capgemini 2015. All Rights Reserved | 3

Add the notes here.

Conditional Statements

- In VBScript there are 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 Statements

- An If statement consists of a Boolean expression followed by one or more statements. If the condition is said to be True, the statements under If condition(s) are Executed. If the Condition is said to be False, the statements after the If loop are executed.
- The syntax of an If statement in VBScript is:

```
If(boolean_expression) Then  
Statement 1 ..... ..... Statement n  
End If
```



Copyright © Capgemini 2015. All Rights Reserved 5

If Else Statements

- An If statement consists of a boolean expression followed by one or more statements. If the condition is said to be True, the statements under If condition(s) are Executed. If the Condition is said to be False, the statements under Else Part would be executed.
- The syntax of an if statement in VBScript is:

```
If(boolean_expression) Then  
    Statement 1  
    .....  
    ....  
    Statement n  
Else  
    Statement 1  
    .....  
    ....  
    Statement n  
End If
```



Copyright © Capgemini 2015. All Rights Reserved

6

If..ElseIf..Else Statements

- An If statement followed by one or more ElseIf Statements that consists of boolean expressions and then followed by a default else statement, which executes when all the condition becomes false.
- The syntax of an If-ElseIf-Else statement in VBScript is:

```
If(boolean_expression) Then  
    Statement 1  
    ....  
    Statement n  
    ElseIf (boolean_expression) Then  
        Statement 1  
        ....  
        Statement n  
    ElseIf (boolean_expression) Then  
        Statement 1  
        ....  
        Statement n  
    Else  
        Statement 1  
        ....  
        Statement n  
End If
```



Copyright © Capgemini 2015. All Rights Reserved 7

Nested If Statement

- An If or Elself statement inside another If or Elself statement(s).
- The Inner If statements are executed based on the Outermost If statements. This enables VBScript to handle complex conditions with ease
- The syntax of a Nested if statement in VBScript is:

```
If(boolean_expression) Then  
    Statement 1  
    Statement n  
    If(boolean_expression) Then  
        Statement 1  
        Statement n  
        Elseif (boolean_expression) Then  
            Statement 1  
            Statement n  
        Else  
            Statement 1  
            Statement n  
        End If  
    Else  
        Statement 1  
        Statement n  
    End If
```



Copyright © Capgemini 2015. All Rights Reserved

8

Select Case Statements

- When a User want to execute a group of statements depending upon a value of an Expression, then Switch Case is used.
- Each value is called a Case, and the variable being switched ON based on each case.
- Case Else statement is executed if test expression doesn't match any of the Case specified by the user.
- Case Else is an optional statement within Select Case, however, it is a good programming practice to always have a Case Else statement
- The syntax of a Switch Statement in VBScript is:

```
Select Case expression
    Case expressionlist1
        statement1
        Statementn
    Case expressionlistn
        statement1
        ...
    Case Else
        elstatement1
        elstatement2
    End Select
```



Copyright © Capgemini 2015. All Rights Reserved 9

The Select-Case Statement

Suppose you were designing a program to decide whether to hire people for programming jobs based on their grades on the final exam in their VBScript programming course. You'll make an offer to those who received an A; you'll bring those who received a B in for an interview; you won't consider anybody else. For example, you could write this:

```
If Grade = "A" Then
    MsgBox "We would like to hire you!"
If Grade = "B" Then
    MsgBox "Please schedule an interview."
If (Grade <> "A") And (Grade <> "B") Then
    MsgBox "I am sorry, we won't be able to consider your application."
```

Using the Select-Case statement, however, you can write this:

```
Select Case Grade
    Case "A"
        MsgBox "We would like to hire you!"
    Case "B"
        MsgBox "Please schedule an interview."
    Case "C"
        MsgBox "I am sorry, we won't be able to consider your application."
End Select
```

The Select-Case statement makes it clear that a program has reached a point with many branches; multiple If-Then statement's do not.
(And the clearer a program is, the easier it is to debug.)

The elimination of all cases that require special testing is so common that VBScript has a way of lumping all the remaining cases into one. This is represented (naturally enough) by the keywords Case Else

Demo

- Conditional statements demo
 - ifstatement
 - Ifelsestatement
 - Ifelseifelsestatement
 - nestedifstatement
 - selectcasestatement



Copyright © Capgemini 2015. All Rights Reserved 10

Add the notes here.

Loops

- A loop statement allows us to execute a statement or group of statements multiple times.
- VBScript provides the following types of loops to handle looping requirements:
 - Do While Loop
 - Do Until Loop
 - While...Wend Loop
 - For Next
 - For-Step-Next
 - For Each Next



Copyright © Capgemini 2015. All Rights Reserved 11

Do..While Loop and Do...Until Loop

- A Do..While and Do...Until loop is used when one wants to repeat a set of statements as long as the condition is true. The Condition may be checked at the beginning of the loop or at the end of the loop.
- The syntax of a Do..While loop in VBScript is:

```
Do While/Until condition  
[statement 1]  
[statement 2]  
...  
[statement n]  
[Exit Do]  
[statement 1]  
[statement 2]  
[statement n]  
Loop
```



Copyright © Capgemini 2015. All Rights Reserved 12

While...Wend Loop

- In a While...Wend loop, if the condition is True, all statements are executed until Wend keyword is encountered.
- If the condition is false, the loop is exited and the control jumps to very next statement after Wend keyword
- The syntax of a While...Wend loop in VBScript is:

```
While condition(s)
    [statements 1]
    [statements 2]
    ...
    [statements n]
Wend
```



Copyright © Capgemini 2015. All Rights Reserved 13

For....Next Loop

- The For...Next statement is used to run a block of code a specified number of times.
- The For statement specifies the counter variable, and its start and end values. The Next statement increases the counter variable by one.



Copyright © Capgemini 2015. All Rights Reserved 5.4

For-Step-Next Loop

- With the Step keyword, you can increase or decrease the counter variable by the value you specify.
- In the example below, the counter variable (m) is Increased by 3, each time the loop repeats
 - For m=1 To 10 Step 3
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 m is decreased by 3, each time the loop repeats.
 - For m=10 To 3 Step -3
some code
Next



Copyright © Capgemini 2015. All Rights Reserved 15

For...Each Loops

- A For Each loop is used when we want to execute a statement or a group of statements for each element in an array or collection.
- The syntax of a For Each loop in VBScript is:

For Each element In Group

[statement 1]

[statement 2]

....

[statement n]

[Exit For]

[statement 11]

[statement 22]

Next



Copyright © Capgemini 2015. All Rights Reserved 16

A **For Each** loop is similar to For Loop; however, the loop is executed for each element in an array or group. Hence, the step counter won't exist in this type of loop and it is mostly used with arrays or used in context of File system objects in order to operate recursively

Demo

- Loops demo

- Doloop
- do_untilloop
- while_wend_loop
- for_next
- for_step_next
- for_each_loop



Copyright © Capgemini 2015. All Rights Reserved 17

Add the notes here.

Lab

▪ Lab 3



Copyright © Capgemini 2015. All Rights Reserved 18

Lesson Summary

- Decision making allows programmers to control the execution flow of a script or one of its sections. The execution is governed by one or more conditional statements.
- When you need to execute a block of code several number of times we use loops.



Copyright © Capgemini 2015. All Rights Reserved 19

Review - Questions

- Question 1: The optional statement in Select Case is _____. (Case or Select Case)
- Question 2: In For-step-Next loop , you must specify an end value that is less than the start value.
 - True or False
- Question 3: In a While..Wend loop, if the condition is True, all statements are executed until _____ keyword is encountered.



VBScript

Lesson 4: Using Functions
and Arrays in VBScript

Lesson Objectives

- Functions
- Arrays



Copyright © Capgemini 2015. All Rights Reserved. 2

Functions and Arrays

- Built In Functions
 - Date and Time Functions
 - Math Functions
 - String Functions
 - Format Functions
 - Other Functions
- Arrays
 - Array Functions
 - Dynamic Arrays
 - Multidimensional arrays



Copyright © Capgemini 2015. All Rights Reserved. 3

Add the notes here.

Built In Functions: Date/Time Functions

Function	Description	Syntax
Date	Returns the current system date	Date
DateAdd	Returns a date to which a specified time interval has been added	DateAdd(interval,number,date)
DateDiff	Returns the number of intervals between two dates	DateDiff(interval,date1,date2)
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)	Day(date)
FormatDateTime	Returns an expression formatted as a date or time	FormatDateTime(date,format)
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)	Hour(time)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date	IsDate(expression)
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)	Minute(time)



Copyright © Capgemini 2015. All Rights Reserved 4

Please note only those parameters mentioned in square brackets are optional.

Explanation of some of the parameters:

- **DateAdd function**

➤ the 'interval' parameter 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

➤ The 'number' parameter is the number of interval you want to add. Can either be positive, for dates in the future, or negative, for dates in the past

➤ The 'date' parameter is the Variant or literal representing the date to which interval is added



Copyright © Capgemini 2015. All Rights Reserved 5

- **DateDiff Function:**

- The 'interval' parameter is the interval you want to use to calculate the differences between date1 and date2. It can take the same values as that of DateAdd function.
- The 'date1', 'date2' parameters are two dates you want to use in the calculation.

Built In Functions: Date/Time Functions

Function	Description	Syntax
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)	Month(date)
MonthName	Returns the name of a specified month	MonthName(month[,abbreviate])
Now	Returns the current system date and time	Now
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)	Second(time)
Time	Returns the current system time	Time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)	Weekday(date[,firstdayofweek])
WeekdayName	Returns the weekday name of a specified day abbreviate[,firstdayofweek]]	WeekdayName(weekday[,abbreviate[,firstdayofweek]])
Year	Returns a number that represents the year	Year(date)



Copyright © Capgemini 2015. All Rights Reserved 6

- **Weekday and WeekdayName Function:**

- The parameter 'firstdayofweek' is Optional. It specifies the first day of the week. It 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

- **MonthName and weekdayName Function:**

- The 'abbreviate' parameter is optional. It can take a boolean value that indicates if the weekday name is to be abbreviated

Demo

- Date functions
 - datefunctions_1
 - datefunctions_2
 - datefunctions_3
 - datefunctions_4



Copyright © Capgemini 2015. All Rights Reserved. T

Add the notes here.

Built In Functions: Math Functions

Function	Description	Syntax
Abs	Returns the absolute value of a specified number	Abs(number) Exp(number)
Exp	Returns e raised to a power	
Int	Returns the integer part of a specified number	Int(number)
Log	Returns the natural logarithm of a specified number	Log(number)
Rnd	Returns a random number less than 1 but greater or equal to 0	Rnd[(number)]
Sgn	Returns an integer that indicates the sign of a specified number	Sgn(number)
Sqr	Returns the square root of a specified number	Sqr(number)
Abs	Returns the absolute value of a specified number	Abs(number)



Copyright © Capgemini 2015. All Rights Reserved

8

Demo

- Math functions
- Mathfunctions



Copyright © Capgemini 2015. All Rights Reserved 0

Add the notes here.

Built In Functions: String Functions

Function	Description	Syntax
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string	InStr([start,]string1,string2[,compare])
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string	InStrRev(string1,string2[,start[,compare]])
LCase	Converts a specified string to lowercase	LCase(string)
Left	Returns a specified number of characters from the left Left(string,length) side of a string	



Copyright © Capgemini 2015. All Rights Reserved 10

Explanation of some of the parameters:

➤ **For the Functions InStr and InStrRev :**

- The parameter 'start' is Optional. It 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.
- The parameter 'string1' is required. This is the string to be searched.
- The parameter 'string2' is required. This is the string expression to search for .
- The 'compare' parameter is optional. It specifies the string comparison to use. Default is 0.

It can have one of the following values:

- 0 = vbBinaryCompare - Perform a binary comparison
- 1 = vbTextCompare - Perform a textual comparison

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

Built In Functions: String Functions

Function	Description	Syntax
Len	Returns the number of characters in a string	Len(string)
LTrim	Removes spaces on the left side of a string	LTrim(string)
RTrim	Removes spaces on the right side of a string	RTrim(string)
Trim	Removes spaces on both the left and the right side of a string	Trim(string)
Mid	Returns a specified number of characters from a string	Mid(string,start[,length])
Replace	Replaces a specified part of a string with another string a specified number of times	Replace(string,find,replacewith[,start[,count[,compare]]])



Copyright © Capgemini 2015. All Rights Reserved 11

➤ For the Replace Function:

- The 'string' parameter is required. It is the string to be searched.
- The 'find' parameter is required. This is the part of the string that will be replaced.
- The 'replacewith' parameter is required. It is the replacement substring.
- The 'start' parameter is optional. It specifies the start position. Default is 1. All characters before the start position will be removed.
- The 'count' parameter is also optional. It specifies the number of substitutions to perform. Default value is -1, which means make all possible substitutions
- The 'compare' parameter is optional. It specifies the string comparison to use. Default is 0

It can have one of the following values:

- 0 = vbBinaryCompare - Perform a binary comparison
- 1 = vbTextCompare - Perform a textual comparison

Built In Functions: String Functions

Function	Description	Syntax
Right	Returns a specified number of characters from the right side of a string	Right(string,length)
Space	Returns a string that consists of a specified number of spaces	Space(number)
StrComp	Compares two strings and returns a value that represents the result of the comparison	StrComp(string1,string2[,compare])
String	Returns a string that contains a repeating character of a specified length	String(number,character)
StrReverse	Reverses a string	StrReverse(string)
UCase	Converts a specified string to uppercase	UCase(string)



Copyright © Capgemini 2015. All Rights Reserved 12

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)

Demo

- String functions
 - stringfunctions_1
 - stringfunctions_2
 - stringfunctions_3



Copyright © Capgemini 2015. All Rights Reserved 13

Add the notes here.

Built In Functions: Format Functions

Function	Description	Syntax
FormatCurrency	Returns an expression formatted as a currency value	FormatCurrency(Expression[, NumDigAfterDec[, InclLeadingDig[, UseParForNegNum[, GroupDig]]]])
FormatDateTime	Returns an expression formatted as a date or time	FormatDateTime(date,format)
FormatNumber	Returns an expression formatted as a number	FormatNumber(Expression[, NumDigAfterDec[, InclLeadingDig[, UseParForNegNum[, GroupDig]]]])
FormatPercent	Returns an expression formatted as a percentage	FormatPercent(Expression[, NumDigAfterDec[, InclLeadingDig[, UseParForNegNum[, GroupDig]]]])



Copyright © Capgemini 2015. All Rights Reserved 5.4

Vbscript FormatCurrency function provides the functionality to format the simple number expression into currency formatted numbers with grouping, specified number of digits after decimal point, leading zero for fractional numbers before decimal point etc. You can also format the negative currency value numbers using vbscript FormatCurrency function.

Syntax:

FormatCurrency (expression, NumDigitsAfterDecimal,IncludeLeadingDigit, UseParensForNegativeNumbers, GroupDigits)

FormatCurrency accepts 5 types of parameters as shown in the above syntax:

1. expression: first parameter accepts number expression that is to be formatted into currency format.
2. NumDigitsAfterDecimal: accepts the number value to specify the number of digits after decimal points.
3. IncludeLeadingDigit: accepts 3 types of values to specify whether to display leading zero for fractional decimal numbers. e.g.: 0.9 or .9
IncludeLeadingDigit accepts following values:
 - 2 : Use the computer default regional settings.
 - 1 : True
 - 0 : False
4. UseParensForNegativeNumbers: accepts 3 types of values to specify whether to display parenthesis () for negative numbers or not. E.g.: -9 or (9)
UseParensForNegativeNumbers accept following values:
 - 2 : Use the computer default regional settings.
 - 1 : True
 - 0 : False



Copyright © Capgemini 2015. All Rights Reserved 15

5. GroupDigits: also accepts 3 types of values to specify whether to group the numbers using default delimiter symbol specified in computer regional settings or not.

GroupDigits accepts the following values:

- 2 : Use the computer default regional settings.
- 1 : True
- 0 : False

All parameters of FormatCurrency function are optional except first parameter i.e. number expression that accepts the number value which is to be formatted.

FormatDateTime Function:

➤ The 'date' parameter is required. Any valid date expression (like Date() or Now()) format Optional. A value that specifies the date/time format to use It 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



Copyright © Capgemini 2015. All Rights Reserved 16

Vbscript FormatNumber function provides the functionality to format the simple number expression into numbers with grouping, specified number of digits after decimal point, leading zero for fractional numbers before decimal point etc. You can also format the negative numbers using vbscript FormatNumber function.

FormatNumber (expression, NumDigitsAfterDecimal, IncludeLeadingDigit, UseParensForNegativeNumbers, GroupDigits)

FormatNumber accepts 5 types of parameters as shown in the above syntax:

1. expression: first parameter accepts number expression that is to formatted.
2. NumDigitsAfterDecimal: accepts the number value to specify the number of digits after decimal points.
3. IncludeLeadingDigit: accepts 3 types of values to specify whether to display leading zero for fractional decimal numbers. e.g.: 0.9 or .9

IncludeLeadingDigit accepts following values:

- 2 : Use the computer default regional settings.
- 1 : True
- 0 : False



Copyright © Capgemini 2015. All Rights Reserved 17

4. **UseParensForNegativeNumbers:** accepts 3 types of values to specify whether to display parenthesis () for negative numbers or not. E.g.: -9 or (9)

UseParensForNegativeNumbers accept following values:

- 2 : Use the computer default regional settings.
- 1 : True
- 0 : False

5. **GroupDigits:** also accepts 3 types of values to specify whether to group the numbers using default delimiter symbol specified in computer regional settings or not.

GroupDigits accepts the following values:

- 2 : Use the computer default regional settings.
- 1 : True
- 0 : False

All parameters of FormatNumber function are optional except first parameter i.e. number expression that accepts the number value which is to be formatted.

Vbscript FormatPercent function provides the functionality to format the simple number expression multiplying it with 100 and display the number values with grouping of digits, specified number of digits after decimal point, leading zero for fractional numbers before decimal point etc. You can also format the negative numbers as percentage using vbscript FormatPercent function.

Syntax:

FormatPercent (expression, NumDigitsAfterDecimal, IncludeLeadingDigit, UseParensForNegativeNumbers, GroupDigits)



FormatPercent accepts 5 types of parameters as shown in the above syntax:

1. expression: first parameter accepts number expression that is to be formatted as percentage with % as trailing character.
2. NumDigitsAfterDecimal: accepts the number value to specify the number of digits after decimal points.
3. IncludeLeadingDigit: accepts 3 types of values to specify whether to display leading zero for fractional decimal numbers. e.g.: 0.9 or .9

IncludeLeadingDigit accepts following values:

- 2 : Use the computer default regional settings.
- 1 : True
- 0 : False

4. UseParensForNegativeNumbers: accepts 3 types of values to specify whether to display parenthesis () for negative numbers or not. E.g.: -9 or (9)

UseParensForNegativeNumbers accept following values:

- 2 : Use the computer default regional settings.
- 1 : True
- 0 : False

5. GroupDigits: also accepts 3 types of values to specify whether to group the numbers using default delimiter symbol specified in computer regional settings or not.

GroupDigits accepts the following values:

- 2 : Use the computer default regional settings.
- 1 : True
- 0 : False

All parameters of FormatPercent function are optional except first parameter i.e. number expression that accepts the number value which is to be formatted as percentage value i.e. number multiplied by 100 with % sign at the end.

Demo

- Format functions
 - FormatCurrencyDemo
 - FormatDateTimeDemo
 - FormatNumberDemo
 - FormatPercentDemo



Copyright © Capgemini 2015. All Rights Reserved 19

Add the notes here.

Built In Functions: Other Functions

Function	Description	Syntax
Eval	Evaluates an expression and returns the result	Eval(expression)
IsEmpty	Returns a Boolean value that indicates whether a specified variable has been initialized or not	IsEmpty(expression)
IsNull	Returns a Boolean value that indicates whether a specified expression contains no valid data (Null)	IsNull(expression)
IsNumeric	Returns a Boolean value that indicates whether a specified expression can be evaluated as a number	IsNumeric(expression)



Copyright © Capgemini 2015. All Rights Reserved 20

Demo

- Other functions
 - otherfunctions



Copyright © Capgemini 2015. All Rights Reserved 21

Add the notes here.

Arrays

- Dim SalesInMonth(11)
- This Dim statement would tell VBScript to set aside 12 slots in memory
- Those slots would be identified as:

`SalesInMonth(0) = some number 'for sales in January`
`SalesInMonth(1) = some number 'for sales in February`

The SalesInMonth array is an example of a *fixed array*



Copyright © Capgemini 2015. All Rights Reserved. 22

In general, a list (or one-dimensional array) is a way to group items so that you can refer to each item individually by means of its index. The index is simply the number you place inside the parentheses; it is the position of the item in the array.

The name of the list has to follow VBScript's rules for variable names. You tell VBScript that you will be working with a list by using a variation of the Dim statement that you have already seen. The only difference is that you need to add the parentheses with the maximum index. For example, the following statement could be used to identify the sales for a year: `Dim SalesInMonth(11)`

In general, a list (or one-dimensional array) is a way to group items so that you can refer to each item individually by means of its index. The index is simply the number you place inside the parentheses; it is the position of the item in the array. The name of the list has to follow VBScript's rules for variable names.

You tell VBScript that you will be working with a list by using a variation of the Dim statement that you have already seen. The only difference is that you need to add the parentheses with the maximum index. For example, the following statement could be used to identify the sales for a year:

`Dim SalesInMonth(11)`

This Dim statement would tell VBScript to set aside 12 slots in memory. Those slots would be identified as:

`SalesInMonth(0) = some number 'for sales in January`
`SalesInMonth(1) = some number 'for sales in February`

`SalesInMonth(11) = some number 'for sales in December`

Dynamic Arrays

- Dynamic array whose size can change inside any event procedure
`Dim gThingsToDo()`
- To change the size of the array each time the user adds or removes an item. This can be done with the ReDim statement:
`ReDim gThingsToDo(.listBox1.ListCount - 1)`
- To know the current upper bound, if array size keeps changing. It is done with the UBound function
`For I = 0 To UBound(AnArray)`



Copyright © Capgemini 2015. All Rights Reserved. 23

Dynamic Versus Fixed Arrays

The SalesInMonth array is an example of a fixed array. That's because we fixed its size in the Dim statement. However, you might need an array whose size can change while the program is running. This is called a dynamic array. To create a dynamic array whose size you can change inside any event procedure, simply use parentheses without the maximum index and make sure that the array is a script-level variable `Dim gThingsToDo()`

Now suppose, for example, we want to activate the Web page and we want to store the contents of the text-box in an array. We need to change the size of the array each time the user adds or removes an item. This can be done with the

ReDim statement:

`ReDim gThingsToDo(listBox1.ListCount - 1)`

When you start enlarging or shrinking the size of an array, it becomes vital (in For-Next loops, for example) to have a way of knowing the current upper bound. This is done with the UBound function. A For-Next loop whose code starts like this will go through all the elements in the array:

`For I = 0 To UBound(AnArray)`

Array Functions

Function	Description	Syntax
Array	Returns a variant containing an array	Array(arglist)
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria	Filter(inputstrings,value[,include[,compare]])
IsArray	Returns a Boolean value that indicates whether a specified variable is an array	IsArray(variable)
Join	Returns a string that consists of a number of substrings in an array	Join(list[,delimiter])



Copyright © Capgemini 2015. All Rights Reserved 24

➤ For the Filter Function:

- The 'inputstrings' parameter is required. It is one-dimensional array of strings to be searched
- The 'value' parameter is required. It is the string to search for
- The 'include' parameter is 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.
- The 'compare' parameter is optional. Specifies the string comparison to use.
It can have one of the following values:
 - 0 = vbBinaryCompare - Perform a binary comparison
 - 1 = vbTextCompare - Perform a textual comparison

➤ For The Join Function:

- The 'list' parameter is required. A one-dimensional array that contains the substrings to be joined
- The 'delimiter' parameter is optional. The character(s) used to separate the substrings in the returned string. Default is the space character

➤ For the LBound and UBound Functions:

- The LBound for any dimension is ALWAYS 0.
- We can use the LBound function with the UBound function to determine the size of an array

Array Functions

Function	Description	Syntax
LBound	Returns the smallest subscript for the indicated dimension of an array	LBound(arrayname[,dimension])
UBound	Returns the largest subscript for the indicated dimension of an array	UBound(arrayname[,dimension])
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings	Split(expression[,delimiter[,count[,compare]]])



Copyright © Capgemini 2015. All Rights Reserved 25

➤ **For the Spilt Function:**

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

Demo

- Array functions
- arrayfunctions



Copyright © Capgemini 2015. All Rights Reserved 26

Add the notes here.

Multidimensional Arrays

- Two-dimensional array for the multiplication tables:
`Dim MultiplicationTable(9,9)`
- This sets aside 10 rows and 10 columns for a total of 100 slots



Copyright © Capgemini 2015. All Rights Reserved. 27

Multidimensional Arrays

Just as lists lead to one-dimensional arrays, tables lead to multidimensional arrays. For example, to make a two-dimensional array for the multiplication tables, you would write this: `Dim MultiplicationTable(9,9)`

This sets aside 10 rows and 10 columns for a total of 100 slots. (Remember that arrays start with a slot at position 0.).

To actually fill the table, use a nested For-Next loop:

```
For I = 0 To 9
    For J = 0 To 9
        MultiplicationTable(I,J) = (I+1) * (J+1)
    Next
Next
```

Converting Variables to Arrays

- We can assign an array to a variable without parentheses
- Then treat the variable just like an ordinary array

```
Dim A  
A = Array(1, 2, 3, 4)
```

Now A(0) gives you 1, A(1) gives you 2



Copyright © Capgemini 2015. All Rights Reserved 20

Converting Variables to Arrays

One of VBScript's most amazing abilities regarding arrays is that you can assign an array to a variable that doesn't use parentheses in its definition, and then treat the variable just like an ordinary array. VBScript provides two ways to do this. The first is with the Array function, which simply turns a bunch of data into an array:

```
Dim A  
A = Array(1, 2, 3, 4)
```

Now A(0) gives you 1, A(1) gives you 2, and so on. You can also declare an array and then simply assign it to another variable:

```
Dim ATable, MultiplicationTable(9,9)  
ATable = MultiplicationTable
```

You can even use a variable to temporarily hold an array in order to "swap" two arrays.

Although List property of a list box is an array, this is true only in a restricted sense. It works exactly like an array as far as element access (via the index), but you cannot assign the List property of a list box to a variable.

Lab

- Lab 2
- Lab 3
- Lab 4



Copyright © Capgemini 2015. All Rights Reserved 29

Add the notes here.

Lesson Summary

- String functions: Lcase, Ucase, Lan, Mid, Replace, InStr
- Numeric functions: Int, Round, Sgn, Abs
- Date Time functions: IsDate(), IsNumeric()
- Array is a way to group items so that you can refer to each item individually by means of its index
- Functions for Parsing and Building Strings: Join, Split, Filter



Copyright © Capgemini 2015. All Rights Reserved 30

Add the notes here.

Review - Questions

- To search a string or a part of string we can use:
 - Option 1: Find
 - Option 2: InStr
 - Option 3: InStrRev
- Question 2: Sgn function returns _____ for negative number.
- Question 3: _____ is used to change the size of an array dynamically.



Copyright © Capgemini 2015. All Rights Reserved. 31

Add the notes here.

VBScript

Lesson 5: User-Defined
Functions & Sub-Procedures in
VBScript

Lesson Objectives

- User Defined Functions
- Sub-procedures
- Passing by reference and passing by value
- Script Debugger



User-defined functions & sub-procedures

- Procedures
 - Subroutines
 - Functions
- Parameter passing
 - Pass by reference
 - Pass by value
- Script debugger
 - Starting Script debugger
 - Stop statement
 - Error trapping



Copyright © Capgemini 2015. All Rights Reserved. 3

Add the notes here.

Procedure and Function

- Procedure

- Named chunk of code that does a specific task
- Can be called (executed) from other procedures or from the main part of the program

- There are two kinds of procedures in VBScript:

- Functions - perform a task and then may or may not return a value
- Sub-procedures - doesn't return a value, it simply does something
 - event procedure is executed automatically in response to an event such as a button being clicked or a key being pressed



Copyright © Capgemini 2015. All Rights Reserved. 4

You can send data to both functions and sub-procedures by passing them parameters.

Usually, functions don't change the data they are sent; they are written primarily to return a value. If you do write a function that changes the information sent to it, the value the function returns is normally used to indicate success or failure.

Sub-procedures that accept data, on the other hand, are usually written to process that data. Therefore, if you direct it to do so, a sub-procedure changes the data it is sent.

User Defined Functions

- Here's how you can write a reusable function:

```
<html>
<body>
<script language="vbscript" type="text/vbscript">
    Dim variable1
    Function Functionname(parameter-list)
        statement 1
        statement 2
        statement 3
        .....
        statement n
    End Function
    variable1=Functionname(parameter-list)
</script>
</body>
</html>
```

- Provided a function is enclosed within the pair of <SCRIPT> </SCRIPT> tags, VBScript is smart enough to find it no matter where you put it



Copyright © Capgemini 2015. All Rights Reserved. 5

Exiting a function prematurely

Occasionally, you will want to exit a function prematurely without assigning a return value. In that case, the default return value is 0 for numbers and "" for strings. The statement that lets you do this is the Exit Function statement.

Sub-procedures

- Sub Procedures are similar to functions but there are few differences.
- Like a function name, a sub Procedure name must follow the rules for a variable name.
- Sub procedures DONOT Return a value while functions may or may not return a value.
- Sub procedures Can be called with or without call keyword.
- Sub procedures are always enclosed within Sub and End Sub statements.

- Syntax for Sub-procedures

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

- Calling the Sub-Procedure

```
Call mysub(arguments)
Or just mention
mysub arguments
```



Copyright © Capgemini 2015. All Rights Reserved. 6

Sub-procedures

You write a function to automate a task and then receive a value.

Let's suppose, however, that you want to automate a task but you don't need to receive a value. In this case, you would write a sub-procedure.

For example, you might want to automate the display of text at a certain heading level. You've gotten tired of writing this:

```
Document.Write "<H" & I & ">" & TheText & "</H>"
```

Here's a sub-procedure that has two parameters, one for the heading level and one for the text to display:

```
Sub WriteInHeadingLevel(HeadingLevel, TheText)
    Document.Write "<H" & (HeadingLevel & ">" & TheText & "</H>"
End Sub
```

In general, as this example shows, the first line of a subprocedure has the keyword Sub followed by the subprocedure's name.

Next comes the parameter list, enclosed in parentheses. (If your subprocedure uses no parameters, the parentheses are optional.) After the parameter list come the statements that make up the body of the subprocedure.

Finally there are the keywords End Sub, which end the subprocedure.

Passing by Reference Versus Passing by Value

- Pass a parameter by reference: any changes to the parameter inside the function or sub procedure will change the value of the original variable.
- Pass a parameter by value: the original variable retains its value after the function or sub procedure terminates, regardless of whether the corresponding parameter was changed inside the function or sub procedure .
- Syntax
 - Sub SubName (ByVal / ByRef Parameter)
- Default is pass by reference.



Copyright © Capgemini 2015. All Rights Reserved. 7

If **ByVal** is specified, then the arguments are sent as byvalue when the function or procedure is called.

If **ByRef** is specified, then the arguments are sent as a reference when the function or procedure is called

Example for Pass ByVal

```
Function GetValue( ByVal var )
    var = var + 1
End Function

'Pass the variable x to the GetValue function ByVal
Dim x: x = 5
Call GetValue(x)

MsgBox "x = " & x
```

Output:
X=5

In other words, when the variable x is passed to GetValue, simply a copy is being passed. When GetValue executes, var stores a copy of the variable x and increments itself by 1. Because a copy of x has been passed, it cannot be modified.



Copyright © Capgemini 2015. All Rights Reserved

8

Example for Pass By Ref

```
Function GetReference( ByRef var )
    var = var + 1
End Function
```

```
'Pass the variable x to the GetReference function ByRef
Dim x: x = 5
Call GetReference(x)
```

```
MsgBox x
```

- Output: 6

Variable x was incremented by 1. Both x and var are incremented unlike the 'pass by val case).

When the function GetReference executes, var becomes a reference of x so any changes made to var would also impact x



Copyright © Capgemini 2015. All Rights Reserved 9

Demo

- Sub Procedures
 - procedure_withoutCall
- Functions
 - Function
 - function_return
- Passby Value
 - passbyvalue
- PassBy Reference
 - passbyreference



Copyright © Capgemini 2015. All Rights Reserved. 10

Add the notes here.

Dialog Boxes

- VBScript allows the developers to interact with the user effectively with the help of dialog boxes.
- Types of Dialog Boxes in VBScript :
 - MsgBox:
Is used to display a message to a user
 - InputBox :
with which user can enter the values



Copyright © Capgemini 2015. All Rights Reserved 11

Dialog Box- Message Box

- The MsgBox function displays a message box and waits for the user to click a button and then an action is performed based on the button clicked by the user.
- Syntax:
`MsgBox(prompt[,buttons][,title])`
- Example:

```
X = MsgBox("message", vbYesNo)

If X = vbYes Then
    MsgBox "The YES button was clicked"
Else
    MsgBox "The NO button was clicked"
End if
```



Copyright © Capgemini 2015. All Rights Reserved 12

Parameter explaination:

Prompt – It is a required parameter. A String that is displayed as a message in the dialog box. The maximum length of prompt is approximately 1024 characters. If the message extends to more than a line, then we can separate the lines using a carriage return character (Chr(13)) or a linefeed character (Chr(10)) between each line.

buttons – It is an optional parameter. A Numeric expression that specifies the type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If left blank, the default value for buttons is 0.

Title – It is an optional parameter. A String expression displayed in the title bar of the dialog box. If the title is left blank, the application name is placed in the title bar

Dialog Box- Message Box

- The various Buttons that can be displayed in a message box :

Constant	Value	Description
vbOKOnly	0	Displays OK button only
vbOKCancel	1	Displays OK and CANCEL buttons
vbAbortRetryIgnore	2	Displays Abort, Retry and Ignore buttons
vbYesNoCancel	3	Displays Yes, No & Cancel buttons
vbYesNo	4	Displays Yes and No buttons
vbRetryCancel	5	Displays Retry and Cancel buttons



Copyright © Capgemini 2015. All Rights Reserved 13

Dialog Box- Message Box

- Built-in constants that can be used to display icons, in a message box

Constant	Value	Description
vbCritical	16	Displays Critical Message icon.
vbQuestion	32	Displays Warning Query icon.
vbExclamation	48	Displays Warning Message icon.
vbInformation	64	Displays Information Message icon.



Copyright © Capgemini 2015. All Rights Reserved 14

Dialog Box- Message Box

- Below constants indicates which button must be the default, in a message box

Constant	Value	Description
vbDefaultButton1	0	First button is default.
vbDefaultButton2	256	Second button is default.
vbDefaultButton3	512	Third button is default.
vbDefaultButton4	768	Fourth button is default.



Copyright © Capgemini 2015. All Rights Reserved 15

Dialog Box- Message Box

- The MsgBox function can return one of the following values:

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



Copyright © Capgemini 2015. All Rights Reserved 16

Dialog Box: InputBox

- The InputBox function helps the user to get the values from the user.
- After entering the values, clicking on the OK button or pressing the ENTER on the keyboard will cause the InputBox function to return the text in the text box.
- Clicking on the Cancel button, the function will return an empty string ("").
- Syntax :
 - `InputBox(prompt[,title][,default][,xpos][,ypos])`



Copyright © Capgemini 2015. All Rights Reserved 17

Prompt - A Required Parameter. A String that is displayed as a message in the dialog box. The maximum length of prompt is approximately 1024 characters. If the message extends to more than a line, then we can separate the lines using a carriage return character (Chr(13)) or a linefeed character (Chr(10)) between each line.

Title - An Optional Parameter. A String expression displayed in the title bar of the dialog box. If the title is left blank, the application name is placed in the title bar.

Default - An Optional Parameter. A default text in the text box that the user would like to be displayed.

XPos - An Optional Parameter. The Position of X axis which represents the prompt distance from left side of the screen horizontally. If left blank, the input box is horizontally centered.

YPos - An Optional Parameter. The Position of Y axis which represents the prompt distance from left side of the screen Vertically. If left blank, the input box is Vertically centered.

Demo

- Message Box
 - messageBox
- Input box
 - InputBox



Copyright © Capgemini 2015. All Rights Reserved. 18

Add the notes here.

Script Debugger

- There are four ways to start Script Debugger:
 - Open the web page to be debugged in Internet Explorer, select View ->Script Debugger->Open
 - Use the Stop statement in your VBScript program. Script Debugger will be displayed when the execution Stop statement
 - Click the Yes button in the Error dialog box
 - In Windows Explorer, open the Script Debugger application file (Msscrdbg.exe)



Copyright © Capgemini 2015. All Rights Reserved. 19

Or you can put a shortcut to this file on your desktop, which is a lot easier to go to.

In all but the last case, you will be working with the Web page that is currently displayed in Internet Explorer. In the last case, you can open the source for an existing Web page by selecting Open from Script Debugger's File menu, or you can create a new page from scratch by selecting New from the File menu. Once Script Debugger is running, you can debug any page that is currently running in Internet Explorer by choosing Running Documents from the View menu.

Stop Statement

- There are two ways to stop a program:
 - Stop statement in the program or using a breakpoint
 - open the source in Script Debugger, move the cursor to the line where you want to set the breakpoint, and select Toggle Breakpoint from the Debug menu
- You must manually remove all Stop statements after you have finished debugging your program.



Copyright © Capgemini 2015. All Rights Reserved. 20

There are two ways to stop a program:

- placing a Stop statement in the program.
- Using a breakpoint.

When Internet Explorer reaches a Stop statement, it stops executing the script at that point. The other way is to open the source in Script Debugger, move the cursor to the line where you want to set the breakpoint, and select Toggle Breakpoint from the Debug menu.

Unfortunately, clicking the Refresh button in Internet Explorer cancels the breakpoint in Script Debugger. For this reason, It is preferred to use Stop statements a lot more than breakpoints.

You must manually remove all Stop statements after you have finished debugging your program.

Now let's assume that we used Notepad to add a Stop statement right after the statements that get the input. Here's what the code would look like:

```
Option Explicit
Dim I, A(4), Average
For I = 1 To 4
    A(I) = InputBox("What is the student's grade on exam " & I & "?") Next Stop I
    would save the file, click the Refresh button, and enter four 100s again. When
    Internet Explorer encounters the Stop statement, it automatically opens Script
    Debugger.
```

Error Trapping

- On Error statement: enables error trapping
 - On Error Resume Next
- This statement works only for the procedure that contains it
- The Err Object :When Internet Explorer encounters an error, it stores information about the error in the properties of a built-in object named the Err object
- Properties
 - Number (Err.Number)
 - Description (Err.Description)



Copyright © Capgemini 2015. All Rights Reserved. 21

Error Trapping

The On Error statement, which enables error trapping in VBScript, looks like the following: On Error Resume Next. This tells Internet Explorer that if it encounters an error to simply forget about it and move to the next statement. This statement works only for the procedure that contains it. This means that you might want to include an On Error statement in each procedure in your program. If you don't do this and an error occurs in a called procedure, Internet Explorer will move to the statement following the call (not to the next statement in the called procedure).

The Err Object

When Internet Explorer encounters an error, it stores information about the error in the properties of a built-in object named the Err object. The value of the Description property is a string describing the error, and the value of the Number property is an integer representing the error. For example, if a user attempts to divide by 0, the Description property would be set to "Division by zero" and the Number property would be set to 11.

If you do use error trapping in a procedure you always add a fragment similar to the one.

Lab

▪ Lab



Copyright © Capgemini 2015. All Rights Reserved. 22

Lesson Summary

- Functions perform a task and then return a value
- Sub-procedures simply does something, it does not return a value
- Pass a parameter by reference makes changes to the parameter inside the function
- Pass a parameter by value retains original value of the variable after the function terminates
- MsgBox helps display an alert
- Debug your script using Script debugger



Review - Questions

- Which of the following returns a value?
 - Option 1: function
 - Option 2: Sub routine
 - Option 3: procedure
- Question 2: While debugging a program the execution stops at _____ statement.
- Question 3: Error information is stored in _____.



VBScript

Lesson 6: Working with
Browser Object Model And
Form Validation

Lesson Objectives

- What is Browser Object Model?
- Browser Objects – Window, Document, Form
- Intrinsic HTML Controls
- Event Driven Programming in VBScript
- Data Validation in VBScript & DOM
- Using Regular Expressions in VBScript

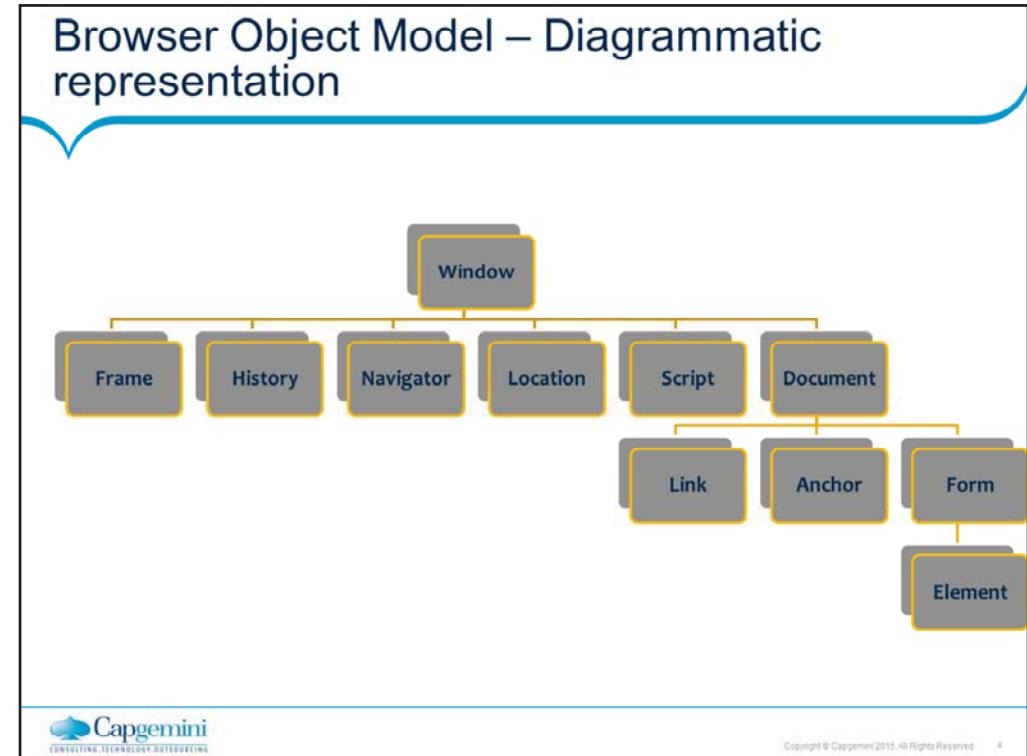


What is Browser Object Model?

- The browser object model is the interface between the code written to control or react to the browser and the internal workings of the browser itself.
- All of the interactions between the code and the browser take place via the object model.
- One can access the browser object model from Microsoft Visual Basic, Scripting Edition.
- This level of abstraction allows us to manipulate the browser without worrying about how the instructions we give are carried out.
- In Browser Object Model, the browser has one top level object called Window.
- The rest of the browser's objects are located beneath the Window object in a logical hierarchy of functionality.



Copyright © Capgemini 2015. All Rights Reserved 3



One of the main foundations of Dynamic HTML is the browser object model. Netscape 2 was actually the first browser that provided a documented object model, which allowed users to access both the browser environment and page contents with scripting code. However certain contents of the page such as text and images weren't accessible, until the advent of Dynamic HTML. If we are going to be able to program the object model using Dynamic HTML, then we need to understand the whole hierarchy of objects inside the browser first and what they all do.

What is the Browser Object Model?

In short, the browser object model is the interface between the code we write to control or react to the browser and the internal workings of the browser itself. This level of abstraction allows us to manipulate the browser without worrying about how the instructions we give are carried out. In effect, the guts of the browser become a 'black box' – we perform some action and get a result, but we don't need to know how our input is translated into output by the browser.

The Object Model Diagram

This diagram is intended to give you a feel for each object's place in the object model. The object model shows the 'hierarchical' relationships between different objects. The connections between the objects in model are arranged in a hierarchy because one object can often contain one (or more) of another object, rather like a parent having one or more children. There are also a number of important objects (like the style object that we talked about in the last chapter) that aren't an intrinsic part of the model at a high level, but are very important when we get into actually working with objects that the model provides. As you can see, the top-level object is the window object. This is the parent object of the entire model – we can think of all the other objects as being children of window. This object contains the document object – think of it in the same way as the browser window contains the HTML document. The window object also contains a collection called frames. Each object in the frames collections represents a frame currently displayed in the browser. In addition to document and frames, the window object also exposes (makes available to the programmer) five separate objects: history, navigator, location, event, and Screen.

The Window Object

- The window object represents the browser itself.
- The Window object is the top level object in VBScript, and contains in itself several other objects, such as "document", "history" etc.
- The window object has properties, methods, and events of its own as well as subsidiary objects.
- The Window Object properties, methods, and events can be called directly in VBScript without qualification, meaning that the Window object does not need to be specified.

Example :

```
<SCRIPT LANGUAGE="VBScript">
    MyVar = Confirm ("Are you sure?")
</SCRIPT>
```



Copyright © Capgemini 2015. All Rights Reserved 3

The Window Object

The window object represents the browser itself.

Since we start IE to view HTML documents and perform any other task using the browser it makes sense that the window object's properties, methods, and events concern themselves with the state and operations of the browser.

The window object has properties, methods, and events of its own as well as subsidiary objects. We'll cover the intrinsic interface elements below before moving onto the objects.

Window Object Properties

Name	Description	Usage
Name	Returns the name of the current window.	MsgBox Name
Parent	Returns a reference to the Parent Set object.	Returns a reference to the Parent Set object.
Self	Returns a reference to the Window object itself.	Returns a reference to the Window object itself.
Top	Returns a reference to the topmost Window object.	Set MyObject=Top
DefaultStatus	Sets or returns the default text in the status bar of a window	DefaultStatus="Ready"
Status	Sets the text in the status bar of a window	Status="Busy"



Copyright © Capgemini 2015. All Rights Reserved 6

Window Object Properties

Name	Description	Usage
Frames	Returns a reference to the Frames array.	Set MyObject= Frames(1)
History	Returns a reference to the History object of the current window.	Set MyObject=History
Navigator	Returns a reference to the Navigator object of the current window.	Set MyObject= Navigator
Document	Returns a reference to the Document object of the current window.	Set MyObject= Document
Location	Returns a reference to the Location object.	Set MyObject= Location



Copyright © Capgemini 2015. All Rights Reserved

- Frames** - The last major element of the window object, aside from its descendant objects, is the frames collection. This collection exists to give a developer access to all of the frames in any frameset currently displayed in the browser. The frames collection is indexed from 0 so that a three-frame frameset, has three elements in the frames collection, numbered from 0 to 2. Each frame is a window object itself, and each frame reference returns a reference to a window object. All we need is a reference to a window object to enable our use of the properties and methods. It shouldn't be surprising then that the frames collection allows us to control each frame in a frameset as if we're controlling a separate window. In fact, this is exactly what we're doing. Each element in the frames collection is a reference to a new window object.
- History** : The history object contains information about all of the different URLs that a client has visited, stored in the history list of the browser. Property length: tells you how many items are stored in the History List.

Methods :

- a. **back** - Takes an integer and moves back this many places in the history list
- b. **forward** - Takes an integer and moves forward this many places in the history list
- c. **go** - Takes a string that represents actual URL and attempts to move to this URL in the history list also takes a positive or negative integer



Copyright © Capgemini 2015. All Rights Reserved.

3. **Navigator** - Using the navigator object we can retrieve information about the capabilities and nature of the browser that is executing our code

Properties

- a. **AppName** - Microsoft Internet Explorer
- b. **AppVersion** - 4.0 (compatible; MSIE 4.0; Windows NT)
- c. **AppCodeName** - Mozilla UserAgent Mozilla/4.0 (compatible; MSIE 4.0; Windows NT)
- d. **CookieEnabled** - True

Methods - There are two methods of the navigator object.

- a. **JavaEnabled** - Returns true or false depending on whether a Java VM is installed or not
- b. **TaintEnabled** - Returns false, included for compatibility with Netscape Navigator

Collections - There are two collections of the navigator object as well

- a. **Plugins** - An alias for the collection of all the <EMBED>ded objects in the page
- b. **MimeTypes** - Collection of all the document and file types supported by the browser



Copyright © Capgemini 2015. All Rights Reserved 9

4. **Document Object** - When an HTML document is loaded into a web browser, it becomes a **document object**. The document object is the root node of the HTML document and the "owner" of all other nodes: (element nodes, text nodes, attribute nodes, and comment nodes).
The document object provides properties and methods to access all node objects, from within JavaScript. The document is a part of the window object and can be accessed as `window.document`.

Note – The Document Object would be discussed separately in the later part of this lesson.

5. **Location** - The location object contains information about the current URL. The location object is part of the window object and is accessed through the `window.location` property.

Window Object Methods

Name	Description	Usage
Alert	Displays an OK message box.	Alert "Stop that!"
Confirm	Displays an OK/Cancel message box and returns a Boolean value. True if user clicked on OK; False if not.	MyVar=Confirm ("Are you sure?")
Prompt	Displays an OK/Cancel input box that returns the text in the input box. The message and the initial text in the input box can be specified.	MyVar=Prompt ("Enter your name", "name")



Copyright © Capgemini 2015. All Rights Reserved 10

The **window** object provides three methods that can be used to display dialogs commonly used for simple messages, confirmations, or prompts

1. The **alert** method displays a dialog with a text message and a single OK button; use it to give the user a message.
2. **Confirm** is identical to alert but the dialog includes an additional Cancel button.
3. The **prompt** method displays a dialog with a single line text box that can be used for input

Window Object Methods

Name	Description	Usage
Open	Opens a new window	open("url", "windowName", "windowFeatures", width, height)
Close	Closes the window.	Close
SetTimeout	Executes the code contained in the string after msec milliseconds have elapsed.	MyID=SetTimeout ("string", msec)
ClearTimeout	Clears the timer specified.	ClearTimeout(MyID)
setInterval	Calls a function or evaluates an expression at specified intervals (in milliseconds)	t=Window.setInterval("Greet",5000)
clearInterval	The clearInterval() method clears a timer set with the setInterval() method.	Window.clearInterval(t)
Navigate	Jumps to a specified URL.	Navigate "http://www.vb.com"



Copyright © Capgemini 2015. All Rights Reserved 11

Open() Method - The Open() method opens a new browser window.

Syntax

window.open(*URL, name, specs, replace*)

Example - window.open("http://www.yahoo.com")
 window.open("", "width=200,height=100")

Close() Method - The close() method closes the current window.

Syntax

window.close()

Example – window.Close()

Timer

Using the setTimeout, clearTimeout, setInterval, and clearInterval methods of the window object we can automatically execute any code we've written after some time interval (that we set) has elapsed.

SetTimeout and ClearTimeout

setTimeout takes the name of a function, and a time value in milliseconds. After the time value has passed, the function is called automatically. For example, the following code calls a routine named TimerFunc after 5000 Milliseconds (5 seconds):

TimeoutID = Window.setTimeout ("TimerFunc",5000) Once you've started a timer with setTimeout, you may find that you want to cancel it so the function specified in the setTimeout call is not executed. This is where the

Navigate - Used to load a different page into the current browser window.

Example - navigate http://www.vbscript.com

Window Object Events

Name	Description	Usage
OnLoad	Fires when page is loaded	<BODY OnLoad="MySub">
OnUnload	Fires when page is unloaded	<BODY OnUnload="MySub">



Copyright © Capgemini 2015. All Rights Reserved 12

Window Events –

These events are very useful for executing any code that needs to coincide with the loading or unloading of the browser.

1. The **onLoad** is fired when the HTML for the current page has been downloaded, but may fire before all images, controls, or applets have finished loading.
2. The **onUnload** event is not only fired before the browser navigates to a new page, but also immediately before the browser is shut down (which is understandable, because the page is indeed being unloaded in preparation for application shutdown).

Demo

- Demo on Window Object – Properties, Methods & Events
 - Window object



Copyright © Capgemini 2015. All Rights Reserved 11

Difference between Properties and Methods

Properties	Methods
Properties relate directly to aspects of the object, or rather, of the thing the object represents	Methods are the things the object's program can do for you
Properties act like variables: You just refer to them by their name	Methods act like functions and subroutines: They can have arguments passed to them
Every property returns a value of some sort. Retrieving a property's value doesn't change anything about the object or whatever it represents	Methods don't have to return a value, but some do
Some properties let you assign new values to them. This changes the attribute of the object and the underlying thing it represents	Invoking a method can change something about the object or the real-world thing it represents



Copyright © Capgemini 2015. All Rights Reserved 14

The Document Object

- The Document object represents the current HTML page in the browser
- Document object also represents any object on the HTML page, such as a link, a form, a button, or an ActiveX object
- The document object represents all of the properties and methods associated with the actual HTML document itself
- This object allows write access to the page, let HTML *form* input areas be examined and/or changed, and so on
- The properties and methods for the Document object must be fully qualified when used
- Example –
`document.write("Good Morning!!")`



Copyright © Capgemini 2015. All Rights Reserved 15

Document Object Properties

Name	Description	Usage
BGColor	Returns or sets the background color	Document.BGColor="#FF0000" Document.BGColor="Red"
FGColor	Returns or sets the foreground color.	Document.FGColor = "#FF0000" Document.FGColor = "Red"
LinkColor	Returns or sets the link color.	Document.LinkColor= "#FF0000" Document.LinkColor= "Red"
ALinkColor	Returns or sets the anchor color.	Document.ALinkColor="#FF0000" Document.ALinkColor="Red"
VLinkColor	Returns or sets the visited link color.	Document.VLinkColor="#FF0000" Document.VLinkColor="Red"
Title	Returns the read-only title of the current HTML page.	MyVar= Document.Title
Location	Returns a string representing the complete URL for the current document.	MyVar= Document.Location



Copyright © Capgemini 2015. All Rights Reserved 16

The Objects inside Document Object

Name	Description
All	All the HTML tags and elements on the page (often too big to work with effectively)
Anchors	All the anchors in the document
Frames	All the frames in the document
Forms	All the HTML forms on the page
Images	All the images on the page
Links	Returns the read-only title of the current HTML page.



Copyright © Capgemini 2015. All Rights Reserved 17

The Form Object

- The Form object represents a form created by the <FORM></FORM> tags.
- The Form object offers properties and methods for any form on the current page.
- The Form object can be accessed as a property of the Document object.
- Any controls contained within the <FORM></FORM> tags are accessed with the *Form.Control.Property* syntax.
- Additionally, the Internet Explorer provides complete support for all of the events for a control contained in a form.
- Form object also contains an Element object that can reference any control on the form either by array or by name.



Copyright © Capgemini 2015. All Rights Reserved 18

The Form Object Properties

Name	Description	Usage
Action	Returns or sets the ACTION attribute	MyForm.Action="http://www.vb-bootcamp"
Encoding	Returns or sets the ENCODING attribute	MyForm.Encoding="text/html"
Method	Returns or sets the METHOD attribute.	MyForm.Method="POST"
Target	Returns or sets the TARGET attribute.	MyForm.Target= "NewWindow"
Elements	Returns the read-only title of the current HTML page.	MyVar= Document.Title



Copyright © Capgemini 2015. All Rights Reserved 19

The Form Object Methods And Event

Name	Description	Usage
Submit	Submits form contents.	MyForm.Submit

Name	Description	Usage
OnSubmit	Fires when form contents are submitted.	MyForm.OnSubmit= "VBScript code"



Copyright © Capgemini 2015. All Rights Reserved 20

Intrinsic HTML Controls

- The Internet Explorer supports a set of controls that are built into the browser itself.
- These controls, called *intrinsic HTML controls*, do not have to be downloaded across the Internet to be used on the client machine.
- Intrinsic controls are not defined with the <OBJECT></OBJECT> tags.
- Instead, they are defined either with the <INPUT> tag or such special tags as the <SELECT></SELECT> tags or <TEXTAREA></TEXTAREA> tags.
- The intrinsic controls are used with these tags because these tags support the original HTML specification for controls on a web page.



Copyright © Capgemini 2015. All Rights Reserved 21

Intrinsic HTML controls existed before VBScript, so the Internet Explorer supports them for backward compatibility.

For example: The OnClick event is defined in the HTML standard, not in Visual Basic for Applications.

Intrinsic HTML controls exist to support the original use of controls in HTML. Intrinsic HTML controls are part of the Internet Explorer architecture.

Although the intrinsic HTML controls support ActiveX concepts such as properties, events, and methods, they are typically defined with standard HTML syntax

Intrinsic HTML Controls

Name	Description
Button	Equivalent to a command button
Text	A one line area for user entry of text
Textarea	A multiline area for user entry of text
Password	A text control that displays asterisks instead of echoing what the user enters
Radio	A group of controls that show multiple options, allowing the user to select one of them
Checkbox	A control that lets the user select one or more possibilities from a group of options
Select	Used for a list of possibilities (resembles a list box ActiveX control)



Copyright © Capgemini 2015. All Rights Reserved 22

Intrinsic HTML Controls

Name	Description
Reset	A button that lets you restore the controls on an HTML form to their initial values
Submit	A button that sends information to the server when the user clicks it
Hidden	Used to store information that the user isn't supposed to see



Copyright © Capgemini 2015. All Rights Reserved 23

Event Driven Programming in VBScript

- Event driven programming is the predominant paradigm in today's modern programming.
- As the name implies, the event driven code gets executed when a certain event occurs.
- If an event occurs, and there is no code associated with that event, then that event gets ignored.
- When you write code for a specific event, it is called an event handler.
- VBS events can originate in the Internet Explorer, HTML Intrinsic controls, or ActiveX custom controls.



Copyright © Capgemini 2015. All Rights Reserved 24

Example – Event Handling in VBScript

```
<HTML>
<HEAD><TITLE>A Simple First Page</TITLE>
<SCRIPT LANGUAGE="VBScript">
  Sub Button1_OnClick
    MsgBox "Welcome to VBScript!!"
  End Sub
</SCRIPT>
</HEAD>
<BODY>
<H3>A Simple First Page</H3>
<FORM><INPUT NAME="Button1" TYPE="BUTTON" VALUE="Click Here"></FORM>
</BODY>
</HTML>
```



Copyright © Capgemini 2015. All Rights Reserved 25

Data Validation in VBScript & DOM

- One of the best uses of Client-Side scripting is to validate user input before processing it.
- The Document Object Model (DOM) provides a means for you to interact programmatically with the document loaded in the browser.
- Window object is the parent of all other objects in DOM.
- The Internet Explorer DOM is a rich environment which provides you great control over the document in the browser.
- The Document Object Model gives us the rich set of objects & properties which can be used to perform state of art data validate in web pages.
- We can create more robust, reliable & dynamic web pages with the help of available control level properties, events & methods.



Copyright © Capgemini 2015. All Rights Reserved 26

Intrinsic HTML Controls – Properties, Events & Methods

Name	Properties	Events	Methods
Button	Form, Enabled, Name ,Value	OnClick ,OnFocus	Click ,Focus
CheckBox	Form, Enabled Name Value Checked DefaultChecked	OnClick, OnFocus	Click ,Focus
Password	Form, Enabled, Name,Value DefaultValue	OnFocus, OnBlur	Focus ,Blur, Select
Radio	Form, Enabled ,Name ,Value, Checked	OnClick ,OnFocus	Click ,Focus
Reset	Form, Enabled, Name, Value	OnClick , OnFocus	Click ,Focus
Select	Name, Length, Options, SelectedIndex	OnFocus, OnBlur, OnChange	Focus ,Blur



Copyright © Capgemini 2015. All Rights Reserved 37

Intrinsic HTML Controls – Properties, Events & Methods

Name	Properties	Events	Methods
Submit	Form, Enabled, Name, Value	OnClick, OnFocus	Click, Focus
Text ,TextArea	Form, Enabled ,Name, Value, DefaultValue	OnFocus ,OnBlur, OnChange ,OnSelect	Focus, Blur, Select



Copyright © Capgemini 2015. All Rights Reserved 28

Demo

- Demo on Intrinsic HTML Controls
 - intrinsic_control



Copyright © Capgemini 2015. All Rights Reserved 29

Using Regular Expressions

- Regular Expressions is a sequence of characters that forms a pattern, which is mainly used for search and replace.
- The purpose of creating a pattern is to match specific strings, so that the developer can extract characters based on conditions and replace certain characters.
- RegExp Object
 - RegExp object helps the developers to match the pattern of strings and the properties and methods help us to work with Regular Expressions easily



Copyright © Capgemini 2015. All Rights Reserved

30

Regular expressions provide tools for developing complex pattern matching and textual search-and-replace algorithms. Regular expressions are one of the most powerful utilities available for manipulating text and data. By creating patterns to match specific strings, a developer has total control over searching, extracting, or replacing data. In short, to master regular expressions is to master your data.

Using Regular Expressions

- The VBScript RegExp object provides 3 properties and 3 methods to the user
- Properties
 - Pattern
 - IgnoreCase
 - Global
- Methods
 - Test (search-string)
 - Replace (search-string, replace-string)
 - Execute (search-string)



Copyright © Capgemini 2015. All Rights Reserved 33

Pattern- A string that is used to define the regular expression. This must be set before use of the regular expression object. **Patterns** are described in more detail below.

IgnoreCase - A Boolean property that indicates if the regular expression should be tested against all possible matches in a string. By default, IgnoreCase is set to False.

Global - A Boolean property that indicates if the regular expression should be tested against all possible matches in a string. By default, **Global** is set to **False**.

Test (string) - The **Test** method takes a string as its argument and returns True if the regular expression can successfully be matched against the string, otherwise False is returned.

Replace (search-string, replace-string) - The **Replace** method takes 2 strings as its arguments. If it is able to successfully match the regular expression in the search-string, then it replaces that match with the replace-string, and the new string is returned. If no matches were found, then the original search-string is returned.

Execute (search-string) - The **Execute** method works like **Replace**, except that it returns a **Matches** collection object, containing a **Match** object for each successful match. It doesn't modify the original string.

VBScript Matches Collection object

- The properties of Matches collection object are read-only, and contain information about each match
 - **Count** - A read-only value that contains the number of Match objects in the collection
 - **Item** - A read-only value that enables Match objects to be randomly accessed from the Matches collection object
- Match objects properties are also read-only
 - FirstIndex
 - Length
 - Value



Copyright © Capgemini 2015. All Rights Reserved 32

The **Matches** collection object is only returned as a result of the **Execute** method. This collection object can contain zero or more **Match** objects, and the properties of this object are read-only. Contained within each Matches collection object are zero or more Match objects. These represent each successful match when using regular expressions. The properties of these objects are also read-only, and contain information about each match.

Regular Expression Match Patterns

- Position Matching
- Literal
- Character Classes
- Repetition
- Alteration and Grouping



Copyright © Capgemini 2015. All Rights Reserved 33

Position Matching

Symbol	Function
^	Only match the beginning of a string."^A" matches first "A" in "An A+ for Anita."
\$	Only match the ending of a string."t\$" matches the last "t" in "A cat in the hat"
\b	Matches any word boundary"\b" matches "ly" in "possibly tomorrow."
\B	Matches any non-word boundary



Copyright © Capgemini 2015. All Rights Reserved 34

The significance of position matching is to ensure that we place the regular expressions at the correct places

Literal

Symbol	Function
Alphanumeric	Matches alphabetical and numerical characters literally.
\n	Matches a new line
\f	Matches a form feed
\r	Matches carriage return
\t	Matches horizontal tab
\v	Matches vertical tab
\?	Matches ?
*	Matches *
\+	Matches +
\.	Matches .
\	Matches



Copyright © Capgemini 2015. All Rights Reserved 35

Any form of characters such as alphabet, number or special character or even decimal, hexadecimal can be treated as a Literal. Since few of the characters have already got a special meaning within the context of Regular Expression, we need to escape them using escape sequences

Literal

Symbol	Function
\{	Matches {
\}	Matches }
\\\	Matches \
\[Matches [
\]	Matches]
\(Matches (
\)	Matches)
\xxx	Matches the ASCII character expressed by the octal number xxx."\\50" matches "(" or chr(40).
\xdd	Matches the ASCII character expressed by the hex number dd."\\x28" matches "(" or chr(40).
\uxxxx	Matches the ASCII character expressed by the UNICODE xxxx."\\u00A3" matches "£".



Copyright © Capgemini 2015. All Rights Reserved 36

Character Classes

Symbol	Function
[xyz]	Match any one character enclosed in the character set. "[a-e]" matches "b" in "basketball"
[^xyz]	Match any one character not enclosed in the character set. "[^a-e]" matches "s" in "basketball"
.	Match any character except \n
\w	Match any word character. Equivalent to [a-zA-Z_0-9]
\W	Match any non-word character. Equivalent to [^a-zA-Z_0-9]
\d	Match any digit. Equivalent to [0-9]
\D	Match any non-digit. Equivalent to [^0-9]
\s	Match any space character. Equivalent to [\t\r\n\f]
\S	Match any non-space character. Equivalent to [^ \t\r\n\f]



Copyright © Capgemini 2015. All Rights Reserved 37

The character classes are the Pattern formed by customized grouping and enclosed within [] braces.

If we are expecting a character class that should not be in the list, then we should ignore that particular character class using the negative symbol, which is a cap ^.

Repetition

Symbol	Function
{x}	Match exactly x occurrences of a regular expression. "\d{5}" matches 5 digits
(x,)	Match x or more occurrences of a regular expression. "\s{2,}" matches at least 2 space characters
{x,y}	Matches x to y number of occurrences of a regular expression. "\d{2,3}" matches at least 2 but no more than 3 digits
?	Match zero or one occurrences. Equivalent to {0,1}. "a\s?b" matches "ab" or "a b"
*	Match zero or more occurrences. Equivalent to {0,}
+	Match one or more occurrences. Equivalent to {1,}



Copyright © Capgemini 2015. All Rights Reserved 38

Alteration and Grouping

Symbol	Function
()	Grouping a clause to create a clause. May be nested "(ab)?(c)" matches "abc" or "c"
	Alternation combines clauses into one regular expression and then matches any of the individual clauses "(ab) (cd) (ef)" matches "ab" or "cd" or "ef"



Copyright © Capgemini 2015. All Rights Reserved 38

Demo

- Demo on Regular Expressions
 - Regular expression



Copyright © Capgemini 2015. All Rights Reserved 40

Lesson Summary

After completing this module, you now know about

- Browser Object Model
- Event Driven Programming in VBScript
- Regular Expressions



Review - Questions

- Which of the following matches the '?' (question mark) in an expression?
 - Option 1: ?
 - Option 2: ?\
 - Option 3: \?
- Question 2: When a code is written for a specific event it is called as _____.
- Question 3: _____ returns the read-only title of the current HTML page.



VB Script Lab Book Version 1.2

Document Revision History

Date	Revision No.	Author	Summary of Changes
June 2011	1.1	Vaishali Kunchur	Material Revamp
April 2015	1.2	Alphy Thomson	Material Revamp

Table of Contents

<i>Document Revision History</i>	2
<i>Table of Contents</i>	3
<i>Getting Started</i>	4
<i>Lab 1. VB Script Basics</i>	5
<i>Lab 2. Variables ,Constants and Operators in VBScript</i>	6
<i>Lab 3. Using Conditional Statements and Loops in VBScript</i>	8
<i>Lab 4. Using Functions and Arrays in VBScript</i>	9
<i>Lab 5. Using User-defined functions & sub-procedures</i>	11
<i>Lab 6. Implementing Form Validation in VBScript</i>	12
<i>Appendices</i>	16
<i>Appendix A: HTML Standards</i>	16

Getting Started

Overview

This lab book is a guided tour for learning VB Script. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for VBScript

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 512MB of RAM
- Internet Explorer 6.0 or higher
- A text editor like Notepad or Visual Studio 2005 is installed

Instructions

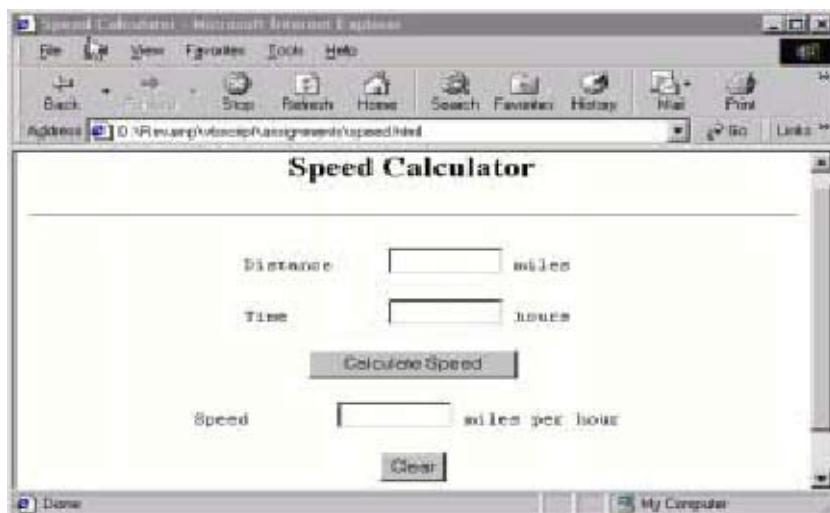
- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory VBScript_Assign. For each lab exercise create a directory as lab <lab number>.

Lab 1. VB Script Basics

Goals	<ul style="list-style-type: none">Understand the process of creating an HTML page and viewing it in a browser window.Learn to apply physical or logical character effects.Learn to manage document spacing
Time	60 minutes

Create HTML Page with VB Script that will be a Speed Calculator as shown in figure.

Solution:



Step 1: From the **Start** menu navigate to **Programs, Accessories, and Notepad**

Step 2: Write the html program in notepad.

Step 3: Mention comments wherever applicable.

Lab 2. Variables ,Constants and Operators in VBScript

Goals	Use Variables ,Constants and Operators in VBScript
Time	120 minutes

Please ensure that the variables have to be defined before it is being used.

1. Create an HTML page with a VBScript program in it to print the colors of the rainbow (use Intrinsic Constants only)
2. Create an HTML page with a VBScript program in it which would take care of the following conversions

Value	Convert to
4.5	Integer, rounding if necessary
8.9	long integer, rounding if necessary
9.6	single-precision number
89.9	double-precision number
85.8	variable of type Currency
44	string
65	Boolean value (True or False)
78	variable of type Byte
89	variable of type Date

3. Create an HTML page with a VBScript program which would return the date and time of 2 months and 15 days from today
4. Create an HTML page with a VBScript program which would declare two constants with values 52 and 54.The program should return the results of the action of the following operators on these 2 constants:

Operator
<>
<=
%
&
EQV

5. State whether the following statement is true or false. Justify your answer by writing VB Script programs.

**“When expressions contain operators from more than one category,
the
order of evaluation is as follows:
comparison operators are evaluated first
arithmetic operators are evaluated next
logical operators are evaluated last”**

Lab 3. Using Conditional Statements and Loops in VBScript

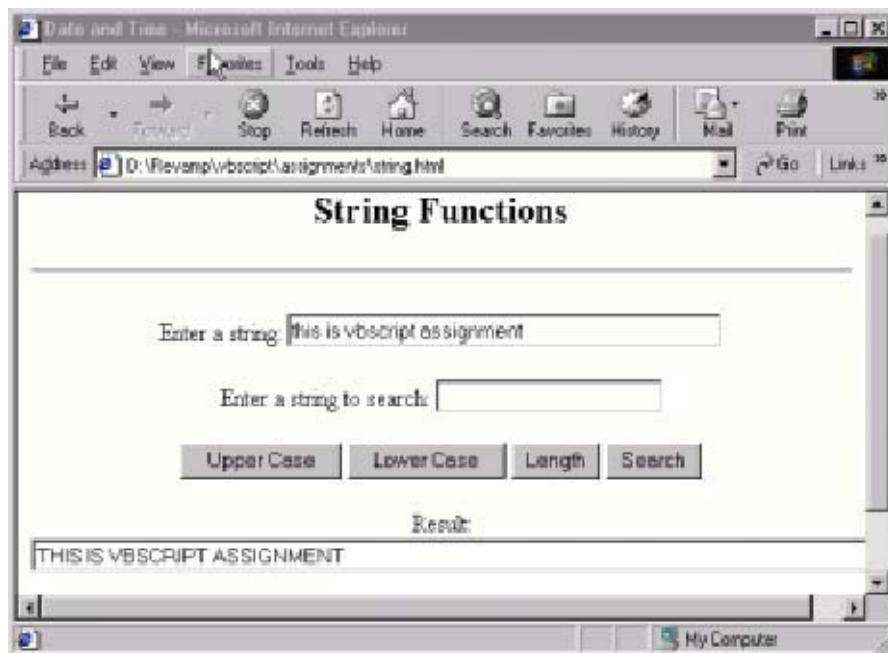
Goals	Use Conditional statements and loops in VBScript
Time	120 minutes

1. Write a VB Script Program that would print the sentence 'Programming level 1' till the 6th level.
Use the html tags to get this statement in different html heading levels.
Do this program using Do loop, Do until, for each, for next.
2. Write a VB Script program to get the above executed for alternate levels using for step next loop.
3. Write a VB Script program to print the month of the year on the choice given by the user.
Eg: if choice 1 is given print January

Lab 4. Using Functions and Arrays in VBScript

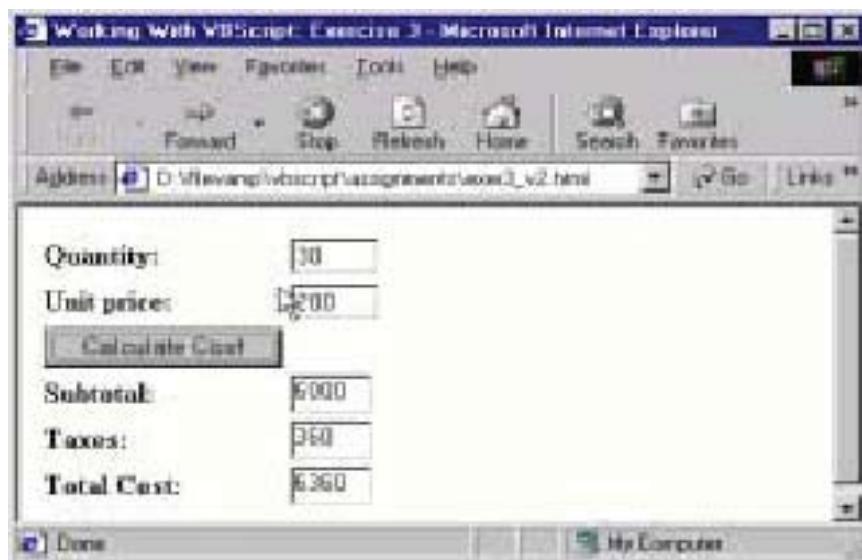
Goals	Use Functions and Arrays in VBScript
Time	120 minutes

1. Create a web page to handle string manipulation. Create a web page containing String Manipulation as shown in figure.



2. Create a web page to use Numeric function

Develop html form as shown in figure. User enters values for quantity and unit price. These fields accept numeric values only. When user clicks on calculate cost, the result of the calculation is shown in Taxes, subtotal and total cost fields. The tax is 20 percent if subtotal is more than 40000 else it is 10 percent.



3. Develop simple calculator to perform arithmetic operations

Lab 5. Using User-defined functions & sub-procedures

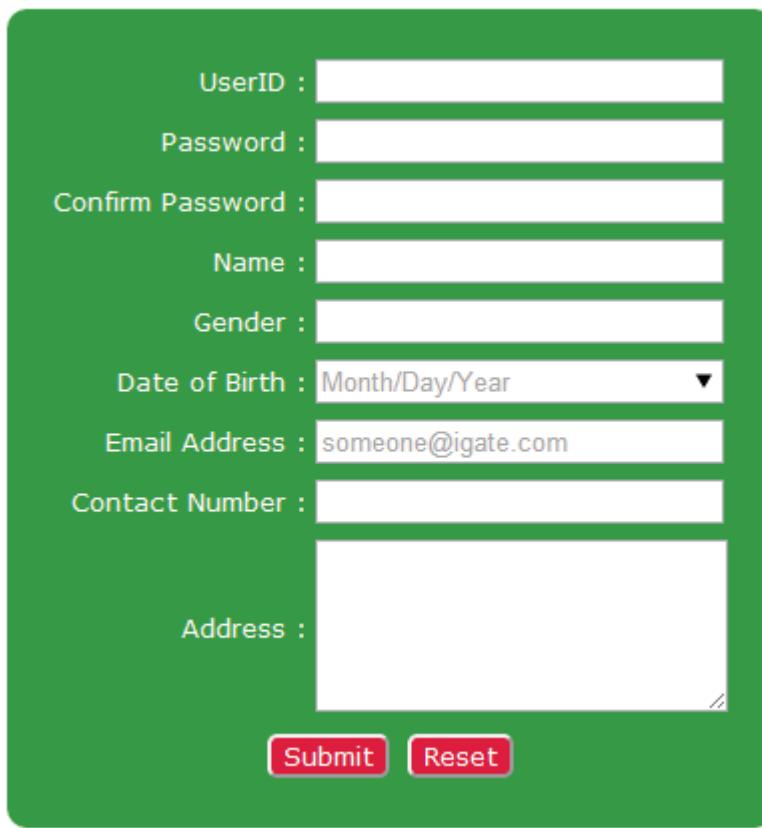
Goals	Use User-defined Functions and Sub-procedures in VBScript
Time	60 minutes

1. Create an HTML page with a VBScript program in it that would take the name of the user and wish him depending on the time of the day.
2. Create an HTML page with a VBScript program in it that would convert the feet into inches and vice versa on clicking on the appropriate button.

Lab 6. Implementing Form Validation in VBScript

Goals	<ul style="list-style-type: none"> • Create HTML Web Page • Implement Form Validation using VBScript DOM & Regular Expression
Time	120 minutes

1. Create an Email Registration Form using HTML5 & CSS3 with its new enhanced Form Elements like given below.



The form consists of the following fields:

- User ID :
- Password :
- Confirm Password :
- Name :
- Gender :
- Date of Birth : Month/Day/Year ▾
- Email Address : someone@igate.com
- Contact Number :
- Address :

At the bottom are two red buttons: **Submit** and **Reset**.

Following are the specifications needs to be implemented while designing the Registration page.

Field Name	Specification
Username	<ul style="list-style-type: none"> • Mandatory field • It should have max 10 characters
Password	<ul style="list-style-type: none"> • Mandatory field
Confirm Password	<ul style="list-style-type: none"> • Mandatory field
Name	<ul style="list-style-type: none"> • Mandatory Field • Only Alphabets allowed (3-10 characters)
Gender	<ul style="list-style-type: none"> • Mandatory field

	<ul style="list-style-type: none"> It should provide users the list of options including values like Male, Female & Transgender
Date of Birth	<ul style="list-style-type: none"> Mandatory It should allow the user to select from the calendar
Email ID	<ul style="list-style-type: none"> Mandatory field It should also check for correct format of Email id A text "someone@igate.com" in a lighter shade should be displayed in the input box till the time it's not selected by user to enter the information.
Address	<ul style="list-style-type: none"> Mandatory field Address can span up to 5 rows
Phone	<ul style="list-style-type: none"> Mandatory field It should accept only numbers in xxx-xxxx-xxxx format
Submit	<ul style="list-style-type: none"> Submit Data
Reset	<ul style="list-style-type: none"> Reset Button It should reset the form by clearing all the controls.

1. Create the below form and add the necessary validations as required.

Travel Authorization Form

Section A: Client Information

Title:
 First name:
 Middle initial:
 Last name:
 E-mail address:
 Phone number:

Section B: Travel Information

Airline name:	<input type="text" value="Select an airline"/>	Additional estimated expenses:	
Destination:	<input type="text"/>	Car rental:	<input type="text"/>
Leave date:	<input type="text"/>	Hotel:	<input type="text"/>
Return date:	<input type="text"/>	Meals:	<input type="text"/>
Estimated cost:	<input type="text"/>	Total estimate:	\$0.00
Includes customer visit?	<input type="checkbox"/>	Payment method used:	
		Cash:	<input checked="" type="radio"/>
		Credit card:	<input type="radio"/>
		Check:	<input type="radio"/>
Purpose of travel:	<input type="text"/>		

Submit

Field Name	Specification
Title	<ul style="list-style-type: none"> Mandatory field Values are Mr./Mrs./Miss
First Name	<ul style="list-style-type: none"> Mandatory field Only Alphabets allowed (3-10 characters)
Middle Initial	<ul style="list-style-type: none"> Mandatory field Only Alphabets allowed (2 characters)
Last name	<ul style="list-style-type: none"> Mandatory Field Only Alphabets allowed (1-20 characters)
Email Address	<ul style="list-style-type: none"> Mandatory field It should also check for correct format of Email id A text "someone@igate.com" in a lighter shade should be displayed in the input box till the time it's not selected by user to enter the information.
Phone Number	<ul style="list-style-type: none"> Mandatory

	<ul style="list-style-type: none"> • It should accept only numbers in xxx-xxxx-xxxx format
Airline Name	<ul style="list-style-type: none"> • Mandatory field • It should provide users the list of options including values like Indigo, Air India, Jet Airways, Spice Jet and GoAir
Designation	<ul style="list-style-type: none"> • Mandatory field • Only Alphabets allowed (1-10 characters)
Leave Date	<ul style="list-style-type: none"> • Mandatory field • It should allow the user to select from the calendar
Return Date	<ul style="list-style-type: none"> • Mandatory field • It should allow the user to select from the calendar
Estimated Cost	<ul style="list-style-type: none"> • Mandatory field • Only digits allowed
Car Rental	<ul style="list-style-type: none"> • Optional field • Only digits allowed
Hotel	<ul style="list-style-type: none"> • Optional field • Only digits allowed
Meals	<ul style="list-style-type: none"> • Optional field • Only digits allowed
Total Estimate	<ul style="list-style-type: none"> • Mandatory field • Only digits allowed
Includes Customer visit?	<ul style="list-style-type: none"> • Checkbox
Payment method used	<ul style="list-style-type: none"> • Radio buttons provided for Cash, credit card and check
Purpose of Travel:	<ul style="list-style-type: none"> • Mandatory Field • It can span up to 5 rows

Appendices

Appendix A: HTML Standards

Key points to keep in mind:

HTML standards help you reach the widest possible audience.

There are many technologies that are associated with HTML because they are used on web pages or in conjunction with HTML. Technologies that are not HTML are:

- Common Gateway Interface (CGI)
- Java
- JavaScript (JavaScript is also not Java)
- Dynamic HTML (DHTML)
- Extensible Markup Language (XML) and
- A variety of other emerging technologies.

For each of the above, please follow coding conventions specified by that technology.

Sometimes you are going to have to break rules and use non-standard syntax for good reason. Try to keep this to the minimum.

How to follow HTML standards:

Identify which version of HTML you are using in your document through the DOCTYPE line at the top of your file.

See the W3C site for more information on document types and DOCTYPE statements.

The important thing to remember is that a DOCTYPE statement is essential to assisting validation software in checking your document.

Use tools (supported by W3C) that support standards. In particular, install and use the Tidy program or Tidy GUI on your computer.

Use W3C validation markup service to check the syntax of documents you create.

Refer to W3C for technical and syntax information.

Some simple HTML standards:

Names of HTML files should always end with the extension *.html*.

Correct: foo.html

Incorrect: foo.bar

Always include a <HTML> tag.