FREE eBook

LEARNING vbscript

Free unaffiliated eBook created from
Stack Overflow contributors.

#vbscript

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: vbscript

It is an unofficial and free vbscript ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official vbscript.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with vbscript

Remarks

VBScript (VBS) is a Visual Basic-flavored scripting language for Internet Explorer and Windows. It can be used on the web in principle, like JavaScript, but does not have much support, so it's usually confined to standalone or server-side scripts in business environments that use Windows exclusively.

Versions

Version	Release Date
1.0	1996-08-13
2.0	1996-12-14
3.0	1997-10-01
4.0	1998-06-01
5.0	1999-03-01
5.1	1999-12-01
5.5	2000-07-01
5.6	2001-08-27
5.7	2006-10-18
5.8	2009-03-19

Examples

Hello World message using cscript and wscript

```
WScript.Echo "Hello world!"
```

This displays a message on the console if run with `cscript.exe` (the console host) or in a message box if run with `wscript.exe` (the GUI host).

If you're using VBScript as the server-side scripting language for a web page (for classic ASP, for example),

```
Response.Write "Hello world!"
```

puts the message into the HTML send to the client (browser).

If you want to displays a message in the message box, you can use:

```
 Msgbox "Hello World!"
```

Read Getting started with vbscript online: https://riptutorial.com/vbscript/topic/463/getting-started-with-vbscript

Chapter 2: Arrays and Loops

Examples

1. Arrays - Static

```
Dim cars(2)
cars(0) = "Ford"
cars(1) = "Audi"
cars(2) = "Prius"
```

2. Arrays - Dynamic

```
Dim cars()
Redim cars(0) 'Give it 1 entry
Dim tmp
tmp = "Ford"

'ubound(arrayvariable) is the count of array size.
'in this case, it would be 1, since there is 1 entry.
cars(ubound(cars)) = tmp 'cars(0)
Redim preserve cars(ubound(cars)+1)
```

5. Creating an array from a text file.

```
Dim cars
Dim filefullname : filefullname = "C:\testenv\test.txt"
'If you can, create an instaneous read for text file data for better memory handling.
'Unless it's a large file and that's impossible.
cars = Split(CreateObject("Scripting.FileSystemObject").OpenTextFile(filefullname, 1).ReadAll,
vbcrlf)
```

7. For Each loop.

You cannot alter the array's contents through the loop variable because it's a temporary each element is being assigned to.

```
Dim cars(2) 'collection of different cars
Dim trace 'track iteration details
cars(0) = "Ford"
cars(1) = "Audi"
cars(2) = "Prius"
For Each car in cars
    trace = trace & car & " temporarily changed to "
    car = "Jeep" 'affects car but not the cars array
    trace = trace & car & vbNewLine
Next

MsgBox trace 'show what happened during the loop
```

```
Dim jeeps : jeeps = 0
For Each car in cars
    If car = "Jeep" Then jeeps = jeeps +1
Next

MsgBox jeeps & " of the cars are Jeeps."
```

6. For Loop

```
Dim i, cars(2)
cars(0) = "Ford"
cars(1) = "Audi"
cars(2) = "Prius"
For i=0 to ubound(cars)
    If cars(i) = "Audi" Then Exit For
Next
```

8. Do While Loop

```
Dim x, cars
x = 0
cars = Split(CreateObject("Scripting.FileSystemObject").OpenTextFile("C:\testenv\example.txt",
1).ReadAll, vbcrlf)
Do While x < ubound(cars)
    If cars(x) = "Audi" Then Exit Loop
    x = x + 1
Loop
```

9. Do Until Loop

```
Dim copycars(), cars(2), x
Redim copycars(0)
x = 0
cars(0) = "Ford"
cars(1) = "Audi"
cars(2) = "Prius"
Do Until x = ubound(cars)
    copycars(ubound(copycars)) = cars(x)
    redim preserve copycars(ubound(copycars)+1)
    x = x + 1
Loop
redim preserve copycars(ubound(copycars)-1) 'trim off the empty last entry
```

3. Arrays - Multi-Dimensional

```
Dim mdArray(2,3)
mdArray(0, 0) = "test1"
mdArray(0, 1) = "test2"
mdArray(0, 2) = "test3"
mdArray(0, 3) = "test4"
mdArray(1, 0) = "test5"
mdArray(1, 1) = "test6"
mdArray(1, 2) = "test7"
mdArray(1, 3) = "test8"
```

```
mdArray(2, 0) = "test9"
mdArray(2, 1) = "test10"
mdArray(2, 2) = "test11"
mdArray(2, 3) = "test12"
```

4. Arrays - Multi-Dimensional - Dynamic

```
Dim mddArray()
ReDim mddArray(0)
Dim ti, testinc: testinc = "test": ti = 1
For i = 0 To 4
    Dim tmpArray(): ReDim tmpArray(0)
    For j = 0 To 3
        tmpArray(UBound(tmpArray)) = testinc & ti
        ti = ti + 1
        ReDim Preserve tmpArray(UBound(tmpArray) + 1)
    Next
    ReDim Preserve tmpArray(UBound(tmpArray) - 1)
    mddArray(i) = tmpArray
    ReDim Preserve mddArray(UBound(mddArray) + 1)
Next
ReDim Preserve mddArray(UBound(mddArray) - 1)
```

Read Arrays and Loops online: https://riptutorial.com/vbscript/topic/7520/arrays-and-loops

Chapter 3: Creating Your First Script

Introduction

To begin, in Windows, create a text document on your desktop (Right-click>New>Text Document.) Change the the extension from ".txt" to ".vbs". At this point it is executable by double clicking it(nothing will happen if you try, there's nothing in it yet.) To edit, right-click document and click edit. Add the example code for your first program.

Parameters

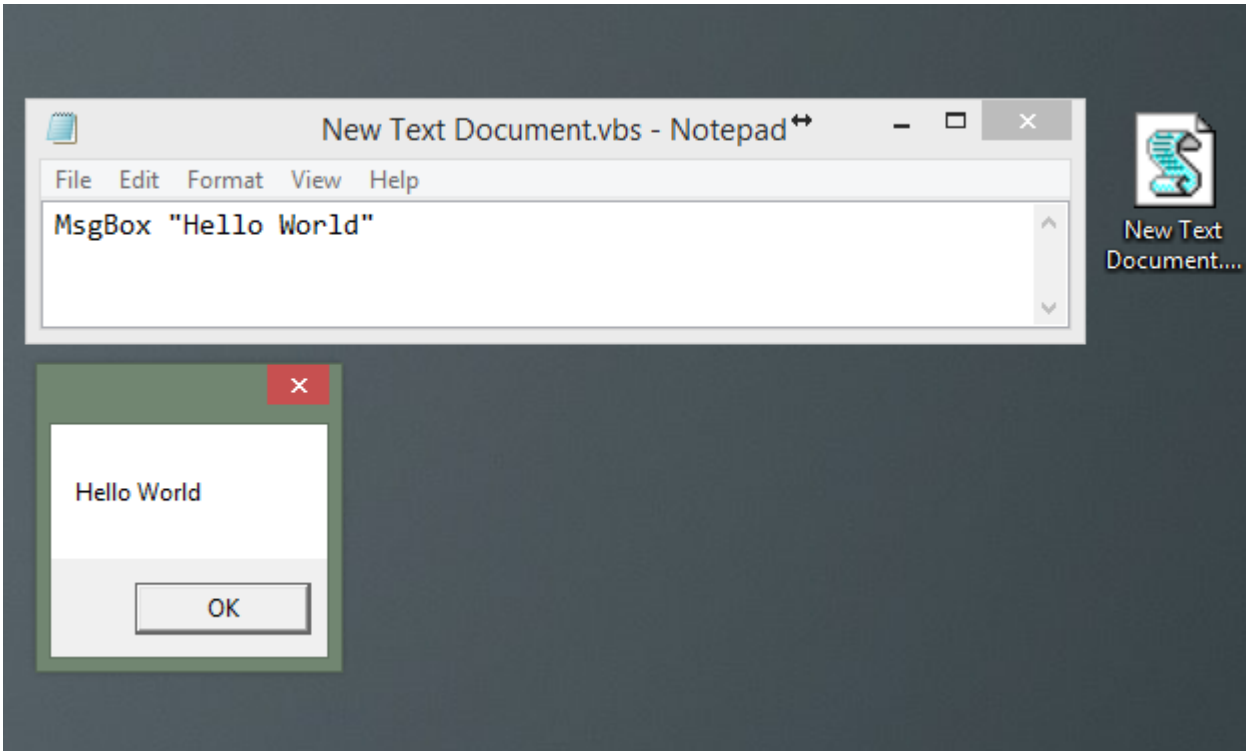
Column	Column
Cell	Cell

Examples

Hello World

Just a simple hello world to start. Copy paste the below into the document, save then double click.

```
MsgBox "Hello World"
```



Explanation

"MsgBox" displays a message in a dialog box and waits for the user to respond.
This is a good method to inform users of actions to be performed or simply the end of the script.
Such as:

Read Creating Your First Script online: https://riptutorial.com/vbscript/topic/8036/creating-your-first-script

Chapter 4: Dictionary Objects

Examples

Create dictionary and Add Items to dictionary

```
Dim oDic
Set oDic = CreateObject("Scripting.Dictionary")
oDic.Add "US", "United States of America"
oDic.Add "UK", "United Kingdom"
```

Check if key Exists in Dictionary

```
If oDic.Exists("US") Then
    msgbox "The Key US Exist. The value is " + oDic("US")
Else
    msgbox "Key Does not exist."
End If
```

Remove Item from Dictionary

```
If oDic.Exists("UK") Then
    oDic.remove("UK")
End If
```

Iterate all items in the dictionary

```
set oDic = CreateObject("Scripting.Dictionary")
oDic.add "USA", "United States of America"
oDic.add "UK", "United Kingdom"
oDic.add "CAN", "Canada"

For Each obj in oDic.Items
    Msgbox obj
Next
Set oDic = Nothing
```

***Output:**

United States of America

United Kingdom

Canada

Iterate all keys in dictionary

```
set oDic = CreateObject("Scripting.Dictionary")
```

```
oDic.add "USA", "United States of America"
oDic.add "UK", "United Kingdom"
oDic.add "CAN", "Canada"

For Each obj in oDic.keys
    Msgbox "Key: " & obj & " Value: " & oDic(obj)
Next
Set oDic = Nothing
```

Delete Key/ keys from Dictionary

```
set oDic = CreateObject("Scripting.Dictionary")
oDic.add "USA", "United States of America"
oDic.add "UK", "United Kingdom"
oDic.add "CAN", "Canada"

' Delete only if Key exists
If oDic.Exists("UK") Then
    oDic.Remove "UK"
End If

' Delete all keys from Dictionary
oDic.removeAll

Set oDic = Nothing
```

Read Dictionary Objects online: https://riptutorial.com/vbscript/topic/8232/dictionary-objects

Chapter 5: FileSystem Objects

Examples

Checking for the existence of a file/folder/drive

Methods used:

```
.DriveExists(strDrive) returns (True/False)
.FileExists(strFile) returns (True/False)
.FolderExists(strFolder) returns (True/False)
```

The following code checks for the existence of a file using the "**FileExists**" method of a file system object. For checking the existence of Folder or a drive, one can use the method "**FolderExists**" or "**DriveExists**" respectively.

Code:

```
Dim strPath, objFso
strPath = "C:\Users\GS\Desktop\tasks.txt"        'Enter the absolute path of the
File/Folder/Drive
Set objFso = CreateObject("Scripting.FileSystemObject")

'Checking for the File's existence
If objFso.FileExists(strPath) then               'returns True if the file exists, else False
    Msgbox "File Exists!"
Else
    Msgbox "File does not Exist!"
End If
Set objFso = Nothing
```

Deleting an existing folder and creating a new Folder

Methods used:

```
.DeleteFolder(FileSpec, Force (True/False))
.CreateFolder(Path)
.DeleteFile(FileSpec, Force (True/False))
```

The following example illustrates the Deletion and creation of a folder using the methods "**DeleteFolder**" and "**CreateFolder**".

Code:

```
Dim strFolderPath, objFso
strFolderPath = "C:\Users\GS\Desktop\testFolder"
Set objFso = CreateObject("Scripting.Filesystemobject")

'Checking for the folder's existence and deleting it, if found
If objFso.FolderExists(strFolderPath) then
```

```
    objFso.DeleteFolder strFolderPath, True                    'True indicates forceful
deletion
End If

'Creating a new Folder
objFso.CreateFolder strFolderPath

Set objFso = Nothing
```

Similarly, One can Delete a File using the "**DeleteFile**" method:

```
Dim strFilePath:strFilePath = "C:\Users\GS\Desktop\tasks.txt"
If objFso.FileExists(strFilePath) then
    objFso.DeleteFile strFilePath, True                      'true indicates forceful deletion
End If
```

Copying a File/Folder

Methods Used:

```
.CopyFile(Source, Dest [,Overwrite (True/False)]
.CopyFolder(Source, Dest [,Overwrite (True/False)]
```

The following code illustrates the use of **CopyFile** method to copy a file to a new location. The same thing can be achieved for the folders by using the **CopyFolder** method.

Code:

```
Dim objFso, strSourcePath, strDestPath
strSourcePath = "C:\Users\GS\Desktop\Source.txt"
strDestPath = "C:\Users\GS\Desktop\Dest.txt"
Set objFso = CreateObject("Scripting.FileSystemObject")
If objFso.FileExists(strSourcePath) then
    objFso.CopyFile strSourcePath, strDestPath, True              'True indicates the
overwritting of the file at the destination path i.e, if the file already exists, it will be
overwritten
End If
Set objFso = Nothing
```

Moving a File/Folder

Methods Used:

```
.MoveFile(Source, Dest)
.MoveFolder(Source, Dest)
```

The following code illustrates the use of **MoveFile** method to Move a file to a new location. The same thing can be achieved for the folders by using the **MoveFolder** method.

Code:

```
Dim objFso, strSourcePath, strDestPath
strSourcePath = "C:\Users\GS\Desktop\Source.txt"
strDestPath = "C:\Users\GS\Desktop\Folder\Dest.txt"
Set objFso = CreateObject("Scripting.FileSystemObject")
If objFso.FileExists(strSourcePath) then
    objFso.MoveFile strSourcePath, strDestPath
End If
Set objFso = Nothing
```

NOTE: We do not have any method of a filesystem object which allows us to rename a file. However, this can be achieved by **MoveFile** method by moving the file to the same location with a different name as shown below:

```
Dim objFso, strSourcePath, strDestPath
strSourcePath = "C:\Users\GS\Desktop\OldName.txt"
strDestPath = "C:\Users\GS\Desktop\NewName.txt"       'Location is same but the name is
different
Set objFso = CreateObject("Scripting.FileSystemObject")
If objFso.FileExists(strSourcePath) then
    objFso.MoveFile strSourcePath, strDestPath
End If
Set objFso = Nothing
```

Object reference to a folder

Methods used:

```
.GetFolder(strPath) - Returns an object referring to the path
```

We can set an object reference to a folder using the **getFolder** method and perform different operations on them.

Code:

```
Dim strFolderPath, objFso, objFolder
strFolderPath = "C:\Users\GS\Desktop\LogsFolder"
Set objFso = CreateObject("Scripting.FileSystemObject")
Set objFolder = objFso.getFolder(strFolderPath)

'Accessing the Folder's Properties
Msgbox objFolder.Name                           'Returns the Folder's Name
Msgbox objFolder.Size                           'Returns the Folder's size in Bytes
Msgbox objFolder.DateCreated                    'Returns the Folder's creation date
Msgbox objFolder.DateLastModified               'Returns the Folder's last modified date
Msgbox objFolder.Path                           'Returns the Folder's Absolute Path

Dim objChildFolders
Set objChildFolders = objFolder.SubFolders      'Returns the collection of all subfolder

Dim objChildFiles
Set objChildFiles = objFolder.Files             'Returns the collection of all files
contained in the folder

'Using the Folder's methods
objFolder.Copy strDestPAth, True                'Copies the folder to path contained in
```

```
strDestPath and overwrite Flag=True
objFolder.Delete True                              'Deletes the Folder; True indicates forceful
Deletion
objFolder.Move strDestPath                         'Moves the Folder to the path contained in
strDestPath variable
objFolder.CreateTextFile strFileName, True         'Created a new text file inside the folder
and overwrites the existing file(if it exists)
Set objChildFiles = Nothing
Set objChildFolders = Nothing
Set objFolder = Nothing
Set objFso = Nothing
```

Object reference to a File

Methods Used:

```
.GetFile(strPath) - Returns an object referring to a file.
```

We can set an object reference to a file using the **getFile** method and perform different operations on them.

Code:

```
Dim strFilePath, objFso, objFile
strFilePath = "C:\Users\GS\Desktop\LogsFolder\file.txt"
Set objFso = CreateObject("Scripting.FileSystemObject")
Set objFile = objFso.getFile(strFilePath)

'Accessing the File's Properties
Msgbox objFile.Name                                'Returns the File's Name
Msgbox objFile.Size                                'Returns the File's size in Bytes
Msgbox objFile.DateCreated                         'Returns the File's creation date
Msgbox objFile.DateLastModified                    'Returns the File's last modified date
Msgbox objFile.Path                                'Returns the File's absolute path

'Using the File's Methods
objFile.Delete True                                'Forcefully deletes the File
objFile.Copy strDestPath, True                     'Copies the file to path contained in variable
strDestPath
objFile.Move strDestPath                           'Moves the file to the path contained in the
variable strDestPath
objFile.OpenAsTextStream mode                      'Opens the file as a text stream in either Read
mode(mode=1), write mode(mode=2) or Append mode(mode=8)
Set objFile = Nothing
Set objFso = Nothing
```

Read FileSystem Objects online: https://riptutorial.com/vbscript/topic/10062/filesystem-objects

Chapter 6: Include files

Introduction

When running VbScript in Windows shell, there is no built in function to include a file, therefore, to organize your code in different files you'll need to create a method to do that.

Remarks

A few things to keep in mind when using the `IncludeFile(p_Path)` method :

- There is no limitation of file type that can be included but the included files content must be VbScript.
- If there is a syntax error in the included file, you will not get the line/column of the error.
- You must define and initialize `std_internal_LibFiles` before the first call to `IncludeFile(p_Path)`
- You can use `IncludeFile(p_Path)` anywhere in your code, including other methods.

Examples

Creating an "include file" method

So the main goal of this function is to :

- Be standalone because it needs to be written in the main VbScript file and cannot be in an included file (because it defines the include function)
- Provide enough information if something goes wrong (ie. the file that was being included, the error that occurred, ...)
- Include a file once and only once to avoid include loops.

```
'
*************************************************************************************************

'! Includes a VbScript file
'! @param p_Path    The path of the file to include
'
*************************************************************************************************

Sub IncludeFile(p_Path)
    ' only loads the file once
    If std_internal_LibFiles.Exists(p_Path) Then
        Exit Sub
    End If

    ' registers the file as loaded to avoid to load it multiple times
    std_internal_LibFiles.Add p_Path, p_Path

    Dim objFso, objFile, strFileContent, strErrorMessage
    Set objFso = CreateObject("Scripting.FileSystemObject")
```

```
    ' opens the file for reading
    On Error Resume Next
    Set objFile = objFso.OpenTextFile(p_Path)
    If Err.Number <> 0 Then
        ' saves the error before reseting it
        strErrorMessage = Err.Description & " (" &  Err.Source & " " & Err.Number & ")"
        On Error Goto 0
        Err.Raise -1, "ERR_OpenFile", "Cannot read '" & p_Path & "' : " & strErrorMessage
    End If

    ' reads all the content of the file
    strFileContent = objFile.ReadAll
    If Err.Number <> 0 Then
        ' saves the error before reseting it
        strErrorMessage = Err.Description & " (" &  Err.Source & " " & Err.Number & ")"
        On Error Goto 0
        Err.Raise -1, "ERR_ReadFile", "Cannot read '" & p_Path & "' : " & strErrorMessage
    End If

    ' this allows to run vbscript contained in a string
    ExecuteGlobal strFileContent
    If Err.Number <> 0 Then
        ' saves the error before reseting it
        strErrorMessage = Err.Description & " (" &  Err.Source & " " & Err.Number & ")"
        On Error Goto 0
        Err.Raise -1, "ERR_Include", "An error occurred while including '" & p_Path & "' : " &
vbCrlf & strErrorMessage
    End If
End Sub
```

Including files

To include a file in another file, just use the one liner :

```
IncludeFile "myOtherFile.vbs"
```

Global initialization

Before we use the IncludeFile method, we need to :

- Declare `std_internal_LibFiles` globally
- Initialize it with a new dictionary

```
Dim std_internal_LibFiles
Set std_internal_LibFiles = CreateObject("Scripting.Dictionary")
```

Read Include files online: https://riptutorial.com/vbscript/topic/8345/include-files

Chapter 7: InputBox

Syntax

- InputBox(prompt[, title][, default][, xpos][, ypos][, helpfile, context])

Parameters

Argument	Detail
`prompt`	Text to display above the input field (usually an instruction as to what is required form the user).
`title`	Caption displayed in the titlebar of the input box.
`default`	A placeholder for the text field, used as the return value if the user doesn't overwrite.
`xpos`	Horizontal distance in twips to display input box from the left edge of the screen.
`ypos`	Vertical distance in twips to display input box from the top edge of the screen.
`helpfile`	A string to determine the help file to be used in order to provide contextual help with the input box.
`context`	If the helpfile argument is supplied, a context number must also be supplied to identify the appropriate help topic in the help file.

Remarks

[1]All arguments except `prompt` are optional.
[2]`helpfile` and `context` are coupled - if one is supplied, the other must also be supplied.

Examples

Use InputBox to assign user input to a string

```
' Omitting the 4th and 5th argument ("xpos" and "ypos") will result in the prompt
' being display center of the parent screen

exampleString = InputBox("What is your name?", "Name Check", "Jon Skeet", 2500, 2000)

WScript.Echo "Your name is " & exampleString
```

Read InputBox online: https://riptutorial.com/vbscript/topic/634/inputbox

Chapter 8: Strings

Remarks

MSDN Date/Time, String and Numeric Functions

https://msdn.microsoft.com/en-us/library/3ca8tfek(v=vs.84).aspx

Examples

1. Standard String

In vbscript, an object doesn't necessarily need a designated type. Similar to C#'s var variable.

```
Dim ExampleString1 As String
Dim ExampleString2
```

2. String Manipulation Basics

```
'Base string
Dim exStr : exStr = " <Head>data</Head> "

'Left
Dim res: res = Left(exStr,6) 'res now equals " <Head"
'Right
Dim res: res = Right(exStr,6) 'res now equals "Head> "
'Mid
Dim res: res = Mid(exStr,8,4) 'res now equals "data"
'Replace
Dim res: res = Replace("variable", "var", "") 'res now equals "riable"
'LCase
Dim res: res = Lcase(exStr) 'res now equals " <head>data</head> "
'UCase
Dim res: res = UCase(exStr) 'res now equals " <HEAD>DATA</HEAD> "
'LTrim
Dim res: res = LTrim(exStr) 'res now equals "<Head>data</Head> " notice no space on left side
'RTrim
Dim res: res = RTrim(exStr) 'res now equals "<Head>data</Head> " notice no space on right side
'Trim
Dim res: res = Trim(exStr) 'res now equals "<Head>data</Head>"
'StrReverse
Dim res: res = StrReverse(exStr) 'res now equals " >daeH/<atad>daeH< "
'String
Dim res: res = String(4,"c") 'res now equals "cccc"


'StrComp - String Compare, by default, compares the binary of 2 strings.
'The third parameter allows text comparison, but does not compare case(capitalization).
'Binary
'-1 = if Binary structure of "cars" < "CARS"
' 0 = if Binary structure of "cars" = "cars"
' 1 = if Binary structure of "CARS" > "cars"
```

```
Dim res: res = StrComp("cars", "CARS") 'res now equals -1
Dim res: res = StrComp("cars", "cars") 'res now equals 0
Dim res: res = StrComp("CARS", "cars") 'res now equals 1

'Text
'-1 = if Text structure of "cars" < "CARSSS"
' 0 = if Text structure of "cars" = "cars"
' 1 = if Text structure of "CARSSS" > "cars"
Dim res: res = StrComp("cars", "CARSSS", 1) 'res now equals -1
Dim res: res = StrComp("cars", "cars", 1) 'res now equals 0
Dim res: res = StrComp("CARSSS", "cars", 1) 'res now equals 1

'Space
Dim res: res = "I" & Space(1) & "Enjoy" & Space(1) & "Waffles" 'res now equals "I Enjoy
Waffles"

'Instr - Returns position of character or string in the variable.
Dim res: res = Instr(exStr, ">") ' res now equals 6
'InstrRev - Returns position of character or string in the variable from right to left.
Dim res: res = Instr(exStr, ">") ' res now equals 2

'Split and Join
'These are methods that can be used with strings to convert a string to an array
'or combine an array into a string
Dim res1 : res1 = Split(exStr, ">")
'res1(0) = " <Head"
'res1(1) = "data</Head"
'res1(2) = " "
Dim res2 : res2 = Join(res1, ">")
'res2 now equals " <Head>data</Head> "
```

3. Searching a String

Given a text file `test.txt`:

```
Ford
Jeep
Honda
```

The following script is processing this text file:

```
'Read in File Data to an array, separate by newline vb equivalent (vbcrlf)
Dim car, cars
Dim filefullname : filefullname = "C:\testenv\test.txt"
cars = Split(CreateObject("Scripting.FileSystemObject").OpenTextFile(filefullname, 1).ReadAll,
vbcrlf)

'Exact Match search.
Dim searchstring : searchstring = "Jeep"
For Each car In cars
    If Car = searchstring Then Exit For
Next

'Partial Match search
'Instr returns >0 if any result is found, if none, InStr returns -1
'The If statement will use -1 = false, >0 = true
Dim searchstring : searchstring = "Jee"
Dim position
```

```
For car = 0 To ubound(cars)
    If InStr(cars(car), searchstring) Then
        position = car
        Exit For
    End If
Next
```

5. Populating array with specific text from string via start and end characters.

```
'Note: I use this method to extract non-html data in extracted GET telnet results
'This example effectively grabs every other vehicle that have start and finish
'characters, which in this case is "/".
'Resulting in an array like this:
'extractedData(0) = "/Jeep"
'extractedData(1) = "/Ford"
'extractedData(2) = "/Honda"


Dim combined : combined = Join(cars, "/") & "/" & Join(cars, "/")
'combined now equals Ford/Jeep/Honda/Ford/Jeep/Honda

Dim record, trigger : record = false : trigger = false
Dim extractedData() : ReDim extractedData(0)
For I = 1 to len(combined) 'searching the string one character at a time
    If trigger Then 'if I've already started recording values
        If Mid(combined, I, 1) = "/" Then 'End Character is found, stop recording
            record = false
            trigger = false
            ReDim Preserve extractedData(ubound(extractedData)+1) 'Prep next Array Entry
        End If
    Else
        If Mid(combined, I, 1) = "/" Then record = true 'Start recording on start character
    End If
    If record Then
        'Increment text on array entry until end variable is found.
        extractedData(ubound(extractedData)) = extractedData(ubound(extractedData)) &
Mid(combined, I, 1)
        trigger = true
    End If
Next
```

4. Chaining string manipulation methods together.

```
Dim exStr : exStr = " <Head>data</Head> "

Dim res
res = Ucase(Replace(Mid(exStr, instr(exStr, ">")+1,4), "ata", "ark"))
'res now equals DARK
'instr(exStr, ">") returns 7
'Mid(" <Head>data</Head> ", 7+1, 4) returns "data"
'Replace("data",  "ata", "ark") returns "dark"
'Ucase("dark") returns "DARK"
```

Read Strings online: https://riptutorial.com/vbscript/topic/7540/strings

Chapter 9: Using Classes

Examples

Creating a Class

```
Class Car

    Private wheels_
    Private distances_

    ' Property getter
    Public Property Get Wheels()
        Wheels = wheels_
    End Property

    ' Property setter
    Public Property Let Wheels(v)
        wheels_ = v
    End Property

    ' Parameterless Constructor
    Public Sub Class_Initialize()
        distances_ = Array(0)
    End Sub

    ' Method
    Public Function GetTotalDistance()
        dim d
        'GetTotalDistance = 0
        For Each d in distances_
            GetTotalDistance = GetTotalDistance + d
        Next
    End Function

    ' Void Method
    Public Sub Drive(distance)
        distances_(ubound(distances_)) = distance
        Redim Preserve distances_(ubound(distances_)+1)
    End Sub

End Class
```

Using a Class Instance

```
' Initialize the object
Dim myCar
Set myCar = new Car

' Setting a property
myCar.Wheels = 4

' Getting a property value
wscript.echo myCar.Wheels
```

```
' Using a subroutine in a class
myCar.Drive 10
myCar.Drive 12

' Using a function in a class
wscript.echo myCar.GetTotalDistance()    ' returns 22
```

Global Factory Function to Emulate a Parameterized Constructor

```
' Making a factory with parameter to the class
Public Function new_Car(wheels)
    Set new_Car = New Car
    new_Car.Wheels = wheels
End Function

' Creating a car through a factory
Dim semiTrailer
Set semiTrailer = new_Car(18)
```

Init Method to Emulate a Parameterized Constructor

```
Class Car
    ...
    ' Parameterless Constructor
    Public Sub Class_Initialize()
        distances_ = Array(0)
    End Sub

    ' Default initialization method that can be invoked without
    ' explicitly using the method name.
    Public Default Function Init(wheels)
        wheels_ = wheels
        Set Init = Me
    End Function
    ...
End Class

Set car1 = (New Car)(18)       ' implicit invocation
Set car2 = (New Car).Init(8)   ' explicit invocation
```

Loading external Class files into script.

```
Dim classFile : classFile = "carClass.vbs"
Dim fsObj : Set fsObj = CreateObject("Scripting.FileSystemObject")
Dim vbsFile : Set vbsFile = fsObj.OpenTextFile(classFile, 1, False)
Dim myFunctionsStr : myFunctionsStr = vbsFile.ReadAll
vbsFile.Close
Set vbsFile = Nothing
Set fsObj = Nothing
ExecuteGlobal myFunctionsStr

Dim car1 : Set car1 = (New Car)(18)
```

Read Using Classes online: https://riptutorial.com/vbscript/topic/3911/using-classes

Chapter 10: WMI queries

Introduction

VBScript can query Windows Management Instrumentation (WMI) for various vital info related to local and remote PC . We can use WMI queries to perform various tasks such as extracting PC's name , getting screen resolution , getting info about user and username , extracting vital info about any process , modifying core system settings,etc .

Below are some examples which use WMI queries to carry out specific tasks .

Examples

Extracting Local PC's name

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" _& "{impersonationLevel=impersonate}!\\" &
strComputer & "\root\cimv2")

Set colSettings = objWMIService.ExecQuery _("Select * from Win32_ComputerSystem")

For Each objComputer in colSettings
wscript.echo objComputer.Name
Next
```

This code will echo PC's name in which it's executed .

Getting number of instances of any process

```
strComputer = "."
instances = 0
processName = "chrome.exe"

Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
Set colProcess = objWMIService.ExecQuery("Select * from Win32_Process")

For Each objProcess in colProcess
If objProcess.Name = processName Then instances = instances + 1
Next

wscript.echo "Process - "&processName&" has "&instances&" instances running."
```

Getting Active Monitor's Screen Resolution

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
Set colItems = objWMIService.ExecQuery("Select * from Win32_DesktopMonitor",,48)

For Each objItem in colItems
```

```
WScript.Echo "ScreenHeight: " & objItem.ScreenHeight
WScript.Echo "ScreenWidth: " & objItem.ScreenWidth
Next
```

Read WMI queries online: https://riptutorial.com/vbscript/topic/9572/wmi-queries

Credits

S. No	Chapters	Contributors
1	Getting started with vbscript	Community, Derpcode, Ekkehard.Horner, Lankymart, Martha, Nathan Tuggy, Purendra Agrawal
2	Arrays and Loops	Rich, Wolf
3	Creating Your First Script	jondanson
4	Dictionary Objects	Barney
5	FileSystem Objects	Gurman
6	Include files	Foxtrot Romeo
7	InputBox	Ansgar Wiechers, Macro Man, Shawn V. Wilson
8	Strings	Foxtrot Romeo, Rich, Wolf
9	Using Classes	Ansgar Wiechers, AutomatedChaos, Rich
10	WMI queries	Abhishek