



Working with Excel Activity 6

- For the given Scores.xls
- How can we write a script to find average of all mp scores for all students? Do design at this point only.
- After discussing the designs, examine [my design](#) and [implementation](#).



Week 4:

- Subprograms and functions
- ByRef and ByVal subprogram calls
- Scope rules
- Working with arrays
- Problem solving with arrays
- Working with text files
- Working with Word Processors



ExampleSubAndFunc.vbs (week3)

- Modularize solution design
 - Solution reads like English statements!
- Code Re-use
- Notice, if your script has Sub-Program or Function, they are simply ignored!
 - VB Script looks for first non-Sub-Program/Function statement and starts execution from there!
- ALWAYS: sub-program and function grouped together!
 - Your "main" program **must** be together (in this class!)
 - Do not scatter your program code in between sub-program and functions
 - Main program completely before or after sub-programs/functions



Sub-Program

- Does not return anything **explicitly**
- Invoke **with** the "Call" Keyword, use ()
 - Call MySubProgram(a, b) – OK
 - Call MySubProgram a, b – Syntax error
- Invoke **without** "Call", don't use()
 - MySubProgram a, b – OK
 - MySubProgram(a, b) – Syntax error
- Use one way, and be consistent e.g.
 - I usually use the "Call" keyword
 - Call MySubProgram (a, b) – This is what I do



Function:

- Function always returns a value
- Syntax: function name returns the value

```
Function ComputeSomething(a, b)
    Dim x
    x = a + b
    ComputeSomething = x
End Function
```

 - Notice we use "**ComputeSomething**" to receive the computed value!
- Invoke with ():
 - Result = ComputeSomething(a, b)
 - () is must!



Comment Block

- Each sub-program/function should start with a comment block (beginning with a "line")
 - E.g. *****
- A Comment block should contain:
 - Function Name
 - Input: Parameters passed in
 - Output: returned parameters
 - Error: Potentially what may cause error, what errors are checked
 - Description: of what the sub-program/function does



ByValByRef.vbs

- **Activity 7, Part I**



Parameters: ByVal and ByRef

- ByVal – a copy is passed
 - Changes within the function have no effect outside of it!
- ByRef – a *reference* is passed
 - Function changes variables of caller
- ByRef and ByVal: work the same way for Function and Sub-Program



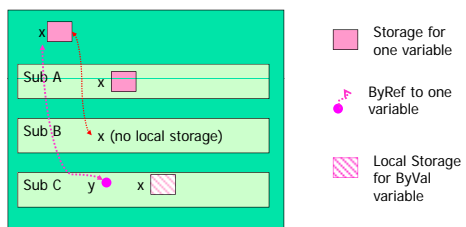
ByRef and ByVal example

- Run the script
 - See results from within both subroutines are the same
 - No resultByVal in the main! It was not passed back to the reference to this symbol.
 - "a" is changed during the last call – why?
 - Which is more secure in general? Why?

ScopeRules.xls

Activity 7: part II

Scoping diagram: ScopeRules.xls



Variable Scope Rule

- Variable search:
 - Look within current scope
 - If not found, look in parent scope
- Global vs. Local (pp. 63-65, Lomax)
 - Global: defined outside of function/sub program
 - Local: defined in a function/sub-program
 - Parameters of a function/sub-program are local variables
- Parameter association by position (NOT name)!



Variable Scope Rule

- Functions/sub-programs should **AVOID** using variables in the global scope!
- Global constants are OK – Why?
- **Exception:** On_Event subroutines do not have calling parameters – **some use of Global is needed.**



OpenExcel.html (week4)

- Use of subprograms to open and initialize the spreadsheet.
- *SubmitButton_onClick* is an event subroutine. What fires it? And what does it do?
- Is there a global variable in *OpenExcel.html*? What is it? Why?
- What is the scope of the variable *activeSheet*? Where is it stored?



Activity 8: hand tracing



Array.vbs

- Syntax:
 - Dim A(5)
 - One dimension: 6 elements, A(0) to A(5)
 - Dim A(2,3)
 - Two dimensions: 12 elements, A(0,0) to A(2,3)
 - Declare and then initialize
 - Dim A
 - A = Array(1, 2, 3, 4)
 - A(3) initialize with data 1,2,3,4



Array utilities

- Dim A(7), B
 - UBound(A) = 7
 - LBound(A) = 0
 - IsArray(A) = true
 - IsArray(B) = false



Sub PrintOneArray()

- msg = msg & "Content"
 - Concatenate msg in the for loop!
- MsgBox msg, ,label
 - Three parameters with the second parameter missing! This is valid in VBScript
 - Label is passed into the sub-program as a parameter



DynamicArray.vbs

- Dim A(4) – array with 5 elements, size cannot be changed!
- Dim A() – without any size!
 - Size can be changed with
 - ReDim A(3) – array with 4 elements
 - ReDim A(5) – array with 6 elements
 - Previous 4 elements are all *erased!*
 - ReDim Preserve A(10)
 - Previous 6 elements are *preserved!*



ReDim an Array:

- How does this work?
- Re-Create and Copy
- Cost and Efficiency?



WorkWithTextFile.vbs

- Scripting.FileSystemObject
 - Manual page:
 - A1->Script Runtime->FileSystemObject
 - ActiveX Control, for file Input/Output
 - FileExists(fileName)
 - GetFile(fileName)
 - OpenAsTextStream()
 - ReadLine
 - AtEndOfStream



Vbs code: open a file ...

```

FileName = user entered file name ... (full path)

Dim FS
Set FS = CreateObject("Scripting.FileSystemObject")
' Get reference to the file system on harddrive

If FS.FileExists(FileName) Then
Dim FileHandler
Set FileHandler = FS.GetFile(FileName)
' Now get the handler to the file based on the FileName

Dim InputTextStream
Set InputTextStream = FileHandler.OpenAsTextStream(FileForReading)
' Now open the file as an inputTextStream so we can "stream" in text data

```



Vbs code: Reading text file

```

-- continue from previous --
Dim InputTextStream
Set InputTextStream = FileHandler.OpenAsTextStream(FileForReading)
' open the file as inputTextStream to "stream" in text data

Dim LineCount
LineCount = 1

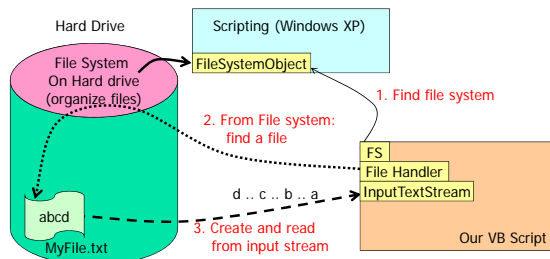
Do While InputTextStream.AtEndOfStream <> TRUE
Dim InputLine
InputLine = InputTextStream.ReadLine
MsgBox "Line " & LineCount & ": " & InputLine
LineCount = LineCount + 1
Loop

MsgBox "End of input file. That was fun, bye!"

```



Reading from a file





Simple problem:

- Continuously enter numbers from user until -1, then Print all the numbers in MsgBox
 - Since we do not know the number of numbers ... we must use ReDim
 - But, do not ReDim for every input number!
- Let's see how we can use this idea in reading a text file ...



UsingDynamicArray.vbs

- Text file
 - Open the file (same procedure as above)
 - Read in every line into an array
 - We do not know how many lines there are in the file! (use dynamic array)
 - Increase array size by 5 at a time! (why not by 1?)



ReadInputStreamToArray()

- The central do/while loop
 - If (counter > UBound (data)) then
ReDim Preserve data (UBound(data) + size)
are you sure that the If test is correct one?
- We must return the counter
 - Since the array size and the number of lines are different!

CommonDialog: open a file ...

```

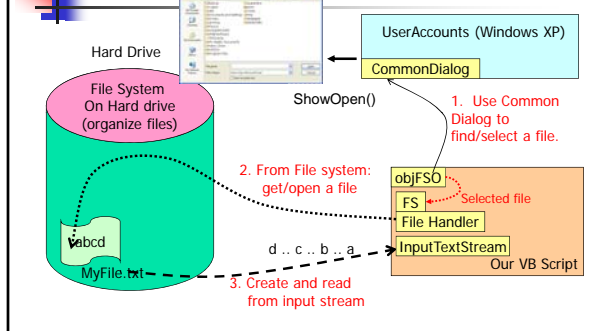
Set ObjFSO = CreateObject("UserAccounts.CommonDialog")
ObjFSO.InitialDir = CurrentFolder ' Initial Folder
SelectResult = ObjFSO.ShowOpen()
SelectedFile = ObjFSO.FileName

Set FS = CreateObject("Scripting.FileSystemObject")
Set FileHandler = FS.GetFile(SelectedFile)

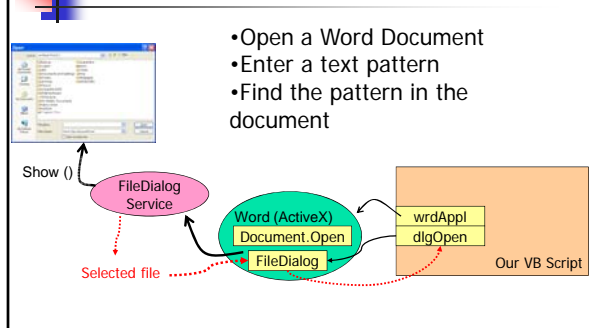
Set InputTextStream = FileHandler.OpenAsTextStream(FileForReading)
' Now open the file as an inputTextStream so we can "stream" in text data

InputLine = InputTextStream.ReadLine
    
```

Using Common Dialog



Using FileDialog within Word





Word.Application

- Active X Control for MS Word
- FileDialog(msoFileDialogOpen) identical to that of Excel (MSO utility)
- Range()
 - Function from Word Document
 - Always start from the beginning of file
 - Search forward (each Execute() call)
- The **With** statement
- WorkwithWord.vbs



NumberOfOccurrence.vbs

- Count number of occurrences of a string in a doc file
- Notice the loop around
 - Find.Execute()
 - Moving *current file location* forward!



Random Number Function

- *Rand.vbs from Week4*
- *How does it work – what is the seed?*
