

# VB Script Examples

By Dave Grund

VBScript is a scripting language developed by Microsoft, and modeled after their Visual Basic development language. It is run within a host environment. Windows Scripting Host (WSH) and Internet Explorer (IE) are the two most common hosts, both of which are supplied with Microsoft Windows.

VBScript is commonly used to perform “quick” programming tasks, like reformatting data. It can be used to interface intricately with the Microsoft Office products. Examples to read and update Excel spreadsheets, create Word tables, etc, are included in this document.

I learn by example better than any other way. I believe it was Jerry Pournelle who coined one law of documentation: *You can never have too many examples.*

This is a collection of example VBScript files. Most are developed from excerpts of code found in the Public Domain areas on the internet, and are themselves Public Domain.

Some of these examples are for demonstration of a feature or technique only, and have little or no true utility value. But most of these examples provide a strong foundation for building a useful tool for your own use.

This document contains only examples. It is a live document, and is continuously being updated.

For instructional or tutorial information, do a Google search on VBScript xxx, where xxx represents the keyword or subject. The more information you supply with keywords, the closer you will get to finding the information you are looking for.

Last update: May 14, 2012

# VBScript Examples

## Table of Contents

<i>VBScript Objects</i> .....	4
Excel .....	5
Outlook .....	6
Word .....	7
<i>VBScripts for Microsoft Excel</i> .....	8
CheckCells .....	9
Create Comma-Separated Values File .....	10
Create a Pie Chart .....	12
Populate a Spreadsheet .....	14
Spreadsheet Statistics .....	18
Copy an Excel Spreadsheet .....	20
Read a Spreadsheet; Write a Spreadsheet .....	22
Divide Big Cells .....	25
Check to See if a Worksheet Exists .....	27
Put a Border Around Cells .....	28
Align Cells at the Top .....	30
<i>VBScripts for Microsoft Word</i> .....	31
Create Word Combinations .....	31
Create Word Tables .....	33
Display Word Properties .....	34
Variable Name Summary .....	38
Copy a Text File into a Word Document .....	42
Process a Word Document .....	44
Create Days of Year .....	46
Print a Mainframe Listing .....	47
<i>VBScripts for Microsoft Outlook</i> .....	49
Display Current Appointments .....	50
Process Current Appointments .....	55
Process Outlook E-mail .....	57
<i>VBScripts for Microsoft Access</i> .....	58
Create an Access Database .....	59
<i>VBScripts for Open Office</i> .....	61
Read a Spreadsheet .....	62
<i>VBScripts for XML</i> .....	64
Create an XML File .....	65
<i>VBScripts for Data</i> .....	66
Classes .....	67
Duration .....	69
Create and Process a Record Set .....	70
Modify a File .....	73
<i>VB Scripts for Lotus Notes</i> .....	74
SendMail .....	74
<i>Snippets</i> .....	75
Change the Format of a Column .....	75

# VBScript Examples

Include Common Code .....	75
<i>Notes</i> .....	76
Converting VBA to VBS .....	76

## VBScript Examples

### ***VBScript Objects***

This section illustrates the VBScript Object Model.

# VBScript Examples

## *Excel*

A workbook is the actual spreadsheet itself. A worksheet is a sheet of that workbook.

objExcel (Set objExcel = CreateObject("Excel.Application") )

.ActiveWindow.FreezePanes = True  
.Columns(3).AutoFit()  
.Columns(1).ColumnWidth = 10  
.Columns(3).Font.ColorIndex = BLUE  
.Columns(i).Interior.ColorIndex = num  
.Rows(i).Interior.ColorIndex = num  
.visible = true            Make Excel visible  
.workbooks.add            Add a new workbook

objWorkbook (Set objWorkbook = objExcel.Workbooks.Open(wkbkname) )

objWorkBook (Set objWorkbook = objExcel.Workbooks.Add())

.sheets.count            Number of worksheets in the workbook  
.sheets(i).Name            The name of the nth worksheet  
.Worksheets(wksheetno)

objWorkSheet (Set objWorkSheet = objExcel.ActiveWorkbook.Worksheets(Sheet))

.Name = "VBS\_Excel\_Example"  
.Cells(row,col).Value = "ColA Hdr"  
.Range("A1:M1").Font.Bold = True  
.Range("A1:M1").Font.Size = 14

Set objWorksheet = objWorkbook.Worksheets(1)

objRange (Set objRange = objWorksheet.UsedRange)

objRange.select

Freeze the panes

objWorkSheet.Range("A2").Select  
objExcel.ActiveWindow.FreezePanes = True

Cleanup

ActiveWorkbook.SaveAs pathname  
ActiveWorkbook.Close  
Application.Quit            Quit Excel

## VBScript Examples

### *Outlook*

```
Const olFolderCalender = 9
Set objOutlook = CreateObject("Outlook.application")
Set objNameSpace = objOutlook.GetNameSpace("MAPI")
Set objFolder = objNameSpace.GetDefaultFolder(olFolderCalender)
Set MyItems = objFolder.Items
MyItems.Subject
MyItems.Start
MyItems.Duration
MyItems.GetRecurrencePattern 3 = weekly, 0 = daily
MyItems.BusyStatus
MyItems.Sensitivity
MyItems.End
```

## VBScript Examples

### *Word*

```
objWord (Set objWord = CreateObject("Word.Application"))  
.Visible = True
```

```
objDoc (Set objDoc = objWord.Documents.Add())  
Tables.Add objRange, NUMBER_OF_ROWS, NUMBER_OF_COLUMNS
```

```
objRange (Set objRange = objDoc.Range())
```

```
objTable (Set objTable = objDoc.Tables(1) ' Work with the first (and only) table)
```

For a complete list of the Microsoft Word objects, follow this link: [Microsoft Word Objects](#).  
Note, however, that some of the links within are *broken*.

For a list of Dialog enumerations, follow this link: [Dialog enumerations](#).

***VBScripts for Microsoft Excel***

# VBScript Examples

## CheckCells

This script will display the length of any cells over a designated number of bytes in length.

### Features:

- Examine each cell in a spreadsheet.

```
Option Explicit
Dim objExcel, objWorkbook, objWorksheet, row, col, msg, ThisTxt, ThisLen, ctrCells, objFSO
Dim ipFN
Const Threshold = 1094          \ ←-----

If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a file to the icon!"
    wscript.quit
end if

Set objFSO = CreateObject("Scripting.FileSystemObject")
IpFN = trim(wscript.arguments(0))

' Start the application
Set objExcel = CreateObject("Excel.Application")
objExcel.Application.visible = true      ' make Excel visible
Set objWorkbook = objExcel.Workbooks.Open(IpFN)
Set objWorksheet = objWorkbook.Worksheets(1)      '<---- Hardcode the worksheet number

msg = "This spreadsheet has " & objWorksheet.UsedRange.Rows.Count & " rows " & _
    objWorksheet.UsedRange.Columns.Count & " columns." & vbCrLf

ctrCells = 0
For Row = 1 to objWorksheet.UsedRange.Rows.Count
    'For col = 1 to objWorksheet.UsedRange.Columns.Count
    For col = 1 to 9
        ctrCells = ctrCells + 1
        ThisTxt = (objWorksheet.Cells(row,col).Value)
        ThisLen = len(ThisTxt)
        If ThisLen > Threshold then
            msg = msg & "Row " & Row & " col " & Col & " is " & ThisLen & " bytes." & vbCrLf
            end if
        next
    next
msg = msg & "I checked " & ctrCells & " cells." & vbCrLf
MsgBox(Msg)
' objExcel.Application.quit Leave the spreadsheet open

Set objWorksheet = nothing
Set objWorkbook = nothing
Set objExcel = nothing

Wscript.quit
```

# VBScript Examples

## *Create Comma-Separated Values File*

This script will create a comma-separated values file. This kind of thing can be done manually, however, this example tailors the process.

### Features:

- Create a text file.

```
` Create a CSV (Comma-separated values) file
Option Explicit
Dim objFSO, InitFSO, objFSO2
Dim objExcel, objWorkbook, objWorksheet, row, col, msg, ThisTxt, ThisLen, i, j
Dim wksheetno, wkbkname, intLimit, ColIDs, IDIdx, strExtra
Dim objTextFile, OpLine
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

Set ObjFSO = CreateObject("UserAccounts.CommonDialog")
ObjFSO.Filter = "Excel files|*.xls"
InitFSO = ObjFSO.ShowOpen
If InitFSO = False Then
    Wscript.Echo "You did not select a file!"
    Wscript.Quit
Else
    wkbkname = ObjFSO.FileName
End If

' Open the CSV file
Set ObjFSO2 = CreateObject("Scripting.FileSystemObject")
Set objTextFile = objFSO2.OpenTextFile _
    ("MyCSVFile.txt", ForWriting, True)

' Start the application
Set objExcel = CreateObject("Excel.Application")

'objExcel.Application.visible = true    ' make Excel visible

Set objWorkbook = objExcel.Workbooks.Open(wkbkname)

If objWorkbook.sheets.count > 1 then
    wkSheetno = cInt(inputbox("Enter the worksheet number you wish to work with",WScript.scriptname &
" needs some input from you!","1"))
else
    wkSheetNo = 1
end if

intLimit = objWorkbook.sheets.count
strExtra = ""
If intLimit > 10 then
    intLimit = 10
    strExtra = "first ten"
end if

Msg = "Workbook Name: " & objWorkbook.Name & vbcrLf

If wksheetno > objWorkbook.sheets.count then
```

## VBScript Examples

```
msgbox "You selected worksheet number " & wksheetno & ", and there are only" &
objWorkbook.sheets.count ,,"Input Error"
wscript.quit
end if

Set objWorksheet = objWorkbook.Worksheets(wksheetno)

' We already know which rows and columns are DEFINED. Now see which cell is the last to actually
' contain a value.
For i = 1 to objWorksheet.UsedRange.Rows.Count
  OpLine = ""
  For j = 1 to objWorksheet.UsedRange.Columns.Count
    OpLine = OpLine & objWorksheet.Cells(i,j).Value
    If j mod 2 = 0 then OpLine = OpLine & ","
  next
  OpLine = left(OpLine,len(OpLine) -1)      ' Drop off the last comma
  objTextFile.writeline(OpLine)
next

Msg = "Complete!"
MsgBox Msg,,WScript.Scriptname

objExcel.Application.quit          ' Do not Leave the spreadsheet open
objTextFile.Close

Set objWorksheet = nothing
Set objWorkbook = nothing
Set objExcel = nothing

Wscript.quit
```

# VBScript Examples

## Create a Pie Chart

This script will create an Excel spreadsheet, and then create a pie chart based on the worksheet it created.

```
' Chart types
Const EXPLODEDPIECHART = 70
Const EXPLODEDDONUT = 80
Const BARCHART = -4100
Const PIECHART = -4102

Const xlDataLabelsShowPercent = 3

Set objExcel = CreateObject("Excel.Application")
objExcel.Visible = True
Set objWorkbook = objExcel.Workbooks.Add()

Set objWorksheet = objWorkbook.Worksheets(1)

objWorksheet.Cells(1,1) = "Operating System"
objWorksheet.Cells(2,1) = "Windows Server 2003"
objWorksheet.Cells(3,1) = "Windows XP"
objWorksheet.Cells(4,1) = "Windows 2000"
objWorksheet.Cells(5,1) = "Windows NT 4.0"
objWorksheet.Cells(6,1) = "Other"

objWorksheet.Cells(1,2) = "Number of Computers"
objWorksheet.Cells(2,2) = 145
objWorksheet.Cells(3,2) = 487
objWorksheet.Cells(4,2) = 211
objWorksheet.Cells(5,2) = 41
objWorksheet.Cells(6,2) = 56

Set objRange = objWorksheet.UsedRange
objRange.Select

Set colCharts = objExcel.Charts
colCharts.Add()

objWorkBook.sheets(4).Delete      ' Sheet3
objWorkBook.sheets(3).Delete      ' Sheet2
objWorkBook.sheets(2).Visible = false ' Sheet 1 has the data that is charted
objWorkBook.sheets(1).Name = "Exploded Pie Chart"

Set objChart = colCharts(1)
objChart.Activate

objChart.ChartType = EXPLODEDPIECHART
objChart.Elevation = 30
objChart.Rotation = 80

objChart.ApplyDataLabels xlDataLabelsShowPercent

objChart.PlotArea.Fill.Visible = False
objChart.PlotArea.Border.LineStyle = -4142

objChart.SeriesCollection(1).DataLabels.Font.Size = 14
objChart.SeriesCollection(1).DataLabels.Font.ColorIndex = 2

objChart.ChartArea.Fill.ForeColor.SchemeColor = 49
objChart.ChartArea.Fill.BackColor.SchemeColor = 23
```

## VBScript Examples

```
objChart.ChartArea.Fill.TwoColorGradient 1,1
```

```
objChart.ChartTitle.Font.Size = 24
```

```
objChart.ChartTitle.Font.ColorIndex = 2
```

```
objChart.Legend.Shadow = True
```

# VBScript Examples

## *Populate a Spreadsheet*

This script will populate an Excel spreadsheet, and color the rows and columns.

### Features:

- Create a spreadsheet
- Populate that spreadsheet
- Delay VBScript processing (sleep).

```
' Populate a Spreadsheet
' This will demonstrate how to populate a spreadsheet, and to color cells/rows/columns

' Color enums
Const Aqua = 42
Const Black = 1
Const Blue = 5          ' Identical to 32
Const BlueGray = 47
Const BrightGreen = 4
Const Brown = 53
Const Cream = 19
Const DarkBlue = 11    ' Identical to 25
Const DarkGreen = 51
Const DarkPurple = 21
Const DarkRed = 9      ' Identical to 30
Const DarkTeal = 49
Const DarkYellow = 12
Const Gold = 44
Const Gray25 = 15
Const Gray40 = 48
Const Gray50 = 16
Const Gray80 = 56
Const Green = 10
Const Indigo = 55
Const Lavender = 39
Const LightBlue = 41
Const LightGreen = 35
Const LightLavender = 24
Const LightOrange = 45
Const LightTurquoise = 20 ' Identical to 34
Const LightYellow = 36
Const Lime = 43
Const NavyBlue = 23
Const OliveGreen = 52
Const Orange = 46
Const PaleBlue = 37
Const Pink = 7         ' Identical to 26
Const Plum = 18        ' Identical to 54
Const PowderBlue = 17
Const Red = 3
Const Rose = 38
Const Salmon = 22
Const SeaGreen = 50
Const SkyBlue = 33
Const TanColor = 40
Const Teal = 14        ' Identical to 31
Const Turquoise = 8    ' Identical to 28
Const Violet = 13     ' Identical to 29
Const White = 2
```

## VBScript Examples

```
Const Yellow = 6           ' Identical to 27

Const NUMROWS = 29
Const NUMCOLS = 13

Set objExcel = CreateObject("Excel.Application")    ' Bind to the Excel object
objExcel.Visible = True
objExcel.Workbooks.Add                               ' Create a new workbook.
Sheet = 1                                           ' Select the first sheet
Set objSheet = objExcel.ActiveWorkbook.Worksheets(Sheet) ' Bind to worksheet.
objSheet.Name = "VBS_Excel_Example"                ' Name the worksheet

strExcelPath = "U:\data\VBScript\Vbs_Excel_Example.xls" ' Set the save location
Randomize

'-----*
'Populate the worksheet with data
'-----*
'Add some titles to row 1
objSheet.Cells(1, 1).Value = "ColA Hdr"           ' Row 1 Column 1 (A)
objSheet.Cells(1, 2).Value = "ColB Hdr"           ' Row 1 Column 2 (B)
objSheet.Cells(1, 3).Value = "ColC Hdr"           ' Row 1 Column 3 (C)
objSheet.Cells(1, 4).Value = "ColD Hdr"           ' Row 1 Column 4 (D)
objSheet.Cells(1, 5).Value = "ColE Hdr"           ' Row 1 Column 5 (E)
objSheet.Cells(1, 6).Value = "ColF Hdr"           ' Row 1 Column 6 (F)
objSheet.Cells(1, 7).Value = "ColG Hdr"           ' Row 1 Column 7 (G)
objSheet.Cells(1, 8).Value = "ColH Hdr"           ' Row 1 Column 8 (H)
objSheet.Cells(1, 9).Value = "ColI Hdr"           ' Row 1 Column 9 (I)
objSheet.Cells(1, 10).Value = "ColJ Hdr"          ' Row 1 Column 10 (J)
objSheet.Cells(1, 11).Value = "ColK Hdr"          ' Row 1 Column 11 (K)
objSheet.Cells(1, 12).Value = "ColL Hdr"          ' Row 1 Column 12 (L)
objSheet.Cells(1, 13).Value = "ColM Hdr"          ' Row 1 Column 13 (M)

'Add some data using a loop
For row = 2 to NUMROWS
    objSheet.Cells(row, 1).Value = "Row " & row & " Col A"
    objSheet.Cells(row, 2).Value = "Row " & row & " Col B"
    objSheet.Cells(row, 3).Value = "Row " & row & " Col C"
    objSheet.Cells(row, 4).Value = "Row " & row & " Col D"
    objSheet.Cells(row, 5).Value = "Row " & row & " Col E"
    objSheet.Cells(row, 6).Value = "Row " & row & " Col F"
    objSheet.Cells(row, 7).Value = "Row " & row & " Col G"
    objSheet.Cells(row, 8).Value = "Row " & row & " Col H"
    objSheet.Cells(row, 9).Value = "Row " & row & " Col I"
    objSheet.Cells(row, 10).Value = "Row " & row & " Col J"
    objSheet.Cells(row, 11).Value = "Row " & row & " Col K"
    objSheet.Cells(row, 12).Value = "Row " & row & " Col L"
    objSheet.Cells(row, 13).Value = "Row " & row & " Col M"
Next

'-----*
' Format the spreadsheet
'-----*
'Put the first row in bold, font size 14
objSheet.Range("A1:M1").Font.Bold = True
objSheet.Range("A1:M1").Font.Size = 14

'Freeze the panes
objSheet.Range("A2").Select
objExcel.ActiveWindow.FreezePanes = True

'Change column A and B to use a fixed width
objExcel.Columns(1).ColumnWidth = 10
```

## VBScript Examples

```
objExcel.Columns(2).ColumnWidth = 10
objExcel.Columns(3).ColumnWidth = 10
objExcel.Columns(4).ColumnWidth = 10
objExcel.Columns(5).ColumnWidth = 10
objExcel.Columns(6).ColumnWidth = 10
objExcel.Columns(7).ColumnWidth = 10
objExcel.Columns(8).ColumnWidth = 10
objExcel.Columns(9).ColumnWidth = 10
objExcel.Columns(10).ColumnWidth = 10
objExcel.Columns(11).ColumnWidth = 10
objExcel.Columns(12).ColumnWidth = 10
objExcel.Columns(13).ColumnWidth = 10

'Change columns to autofit
objExcel.Columns(3).AutoFit()
objExcel.Columns(6).AutoFit()

'Change the background color of columns
intNext = RN
For i = 1 to NUMCOLS
    intNext = RN
    objExcel.Columns(i).Interior.ColorIndex = intNext
    wscript.sleep(400)
Next

'Change the background color of rows
For i = 1 to NUMROWS
    intNext = RN
    objExcel.Rows(i).Interior.ColorIndex = intNext
    wscript.sleep(300)
Next

'Change the font color of column C
objExcel.Columns(3).Font.ColorIndex = BLUE

' Now do some specific colorization
objSheet.Cells(1,1).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(2,2).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(3,3).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(4,4).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(5,5).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(6,6).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(7,7).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(8,8).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(9,9).Interior.ColorIndex = LightYellow
wscript.sleep(300)
objSheet.Cells(10,10).Interior.ColorIndex = LightYellow
wscript.sleep(300)

MsgBox "All done!"

'-----*
' Save the spreadsheet and close the workbook
'-----*
objExcel.ActiveWorkbook.SaveAs strExcelPath
```

## VBScript Examples

```
'objExcel.ActiveWorkbook.Close

'objExcel.Application.Quit          'Quit Excel

'Clean Up
'Set objSheet = Nothing
'Set objExcel = Nothing

Function RN()
Dim TempRN
TempRN = Int((40) * Rnd + 1)
' Get rid of colors that we do not want
do while (TempRN = BLACK) or (TempRN = DarkBlue) or (TempRN = DarkPurple) or _
    (TempRN = 25)
    TempRN = Int((40) * Rnd + 1)
Loop
RN = TempRN
end Function
```

# VBScript Examples

## *Spreadsheet Statistics*

This script will display statistics about it: the last defined cell, the last used cell, etc.

### Features:

- File open dialogue

```
' Spreadsheet Statistics
Option Explicit
Dim objFSO, InitFSO
Dim objExcel, objWorkbook, objWorksheet, row, col, msg, ThisTxt, ThisLen, i, j
Dim wksheetno, wkbkname, intLimit, ColIDs, IDIdx, strExtra

Set ObjFSO = CreateObject("UserAccounts.CommonDialog")
ObjFSO.Filter = "Excel files|*.xls"
InitFSO = ObjFSO.ShowOpen
If InitFSO = False Then
    Wscript.Echo "You did not select a file!"
    Wscript.Quit
Else
    wkbkname = ObjFSO.FileName
End If

' Start the application
Set objExcel = CreateObject("Excel.Application")

'objExcel.Application.visible = true    ' make Excel visible

Set objWorkbook = objExcel.Workbooks.Open(wkbkname)

If objWorkbook.sheets.count > 1 then
    wkSheetno = cInt(inputbox("Enter the worksheet number you wish to work with","SSStats needs some
input from you!","1"))
else
    wkSheetNo = 1
end if

intLimit = objWorkbook.sheets.count
strExtra = ""
If intLimit > 10 then
    intLimit = 10
    strExtra = "first ten"
end if

Msg = "Workbook Name: " & objWorkbook.Name & vbcrLf
Msg = Msg & "No. of worksheets in this workbook: " & objWorkbook.sheets.count & vbcrLf

If objWorkbook.sheets.count > 1 then
    Msg = Msg & "The " & strExtra & " worksheets are:" & vbcrLf
    for i = 1 to intLimit
        Msg = Msg & i & ". " & objWorkBook.sheets(i).Name & vbcrLf
    next
end if

Msg = Msg & vbcrLf    ' Blank line

' Get some stats on a specific worksheet
If wksheetno > objWorkbook.sheets.count then
```

## VBScript Examples

```
msgbox "You selected worksheet number " & wksheetno & ", and there are only" &
objWorkbook.sheets.count ,, "Input Error"
wscript.quit
end if

Set objWorksheet = objWorkbook.Worksheets(wksheetno)

ColIDs = " A B C D E F G H I J K L M N O P Q R S T U V W X Y Z"           ' A thru Z
ColIDs = ColIDs & "AAABACADAEAFAGAHATAJAKALAMANAOPAQAQARASATAUAVAWAXAYAZ" ' AA thru AZ
ColIDs = ColIDs & "BABBBCEBDEBEFBGBHBIBJBKBLMBNBOBPBQBRSBTTBUBVBWBXBYBZ" ' BA thru BZ
ColIDs = ColIDs & "CACBCCDCECFGCHCICJCKCLCMCNCOCPQCRCSCCTCUCVCWCXCXCYCZ" ' CA thru CZ
ColIDs = ColIDs & "DADBDCDDDEDFDGDHDIDJDKDLMDNDODPDQDRDSDTDUDVDWDXDYDZ" ' DA thru DZ
ColIDs = ColIDs & "EAEBECEDEEEFEGEHEIEJEKELEMENEOEPEQERESETEUEVEWEWEYEYEZ" ' EA thru EZ
ColIDs = ColIDs & "FAFBFCFDFEFFFFGFHFIFJFKFLFMFNFOFPFQFRFSFTFUFVFWFXFYFZ" ' FA thru FZ
ColIDs = ColIDs & "GAGBGC GDGEGFGGGHGIGJGKGLGMGNOGPGQGRGSGTGUGVGWGXGYGZ" ' GA thru GZ
ColIDs = ColIDs & "HAHBHCHDHEHFHGHGHHIHHJKHKLHMHNHOHPHQHRHSHTHUHVHWHXHYHZ" ' HA thru HZ
ColIDs = ColIDs & "IAIBICIDIEIFIGIHHIIJIKILIMINIOIPIQIRISITIUIV"         ' IA thru IV (230, max)

Msg = Msg & "Worksheet " & wksheetno & " is named " & objWorksheet.Name & vbcrLf
Msg = Msg & "The last defined cell is ROW " & objWorksheet.UsedRange.Rows.Count & ", COL "
IDIdx = objWorksheet.UsedRange.Columns.Count * 2 - 1
Msg = Msg & mid(ColIDs, IDIdx, 2) & vbcrLf
Msg = Msg & "and contains " & _
    ObjWorksheet.Cells(objWorksheet.UsedRange.Rows.Count, _
        objWorksheet.UsedRange.Columns.Count).Value & vbcrLf & vbcrLf

' We already know which rows and columns are DEFINED. Now see which cell is the last to actually
' contain a value.
For i = objWorksheet.UsedRange.Rows.Count to 1 step -1
    For j = objWorksheet.UsedRange.Columns.Count to 1 step -1
        if len(objWorksheet.Cells(i,j).Value) > 0 then
            Msg = Msg & "The last USED    cell is ROW " & i & ", COL "
            IDIdx = j*2-1
            Msg = Msg & mid(ColIDs, IDIdx, 2) & vbcrLf
            Msg = Msg & "and contains " & objWorksheet.Cells(i,j).Value & vbcrLf
            i = 1 : j = 1          ' Set to exit the loop
            end if
        next
    next

MsgBox Msg,,WScript.Scriptname

objExcel.Application.quit           ' Do not Leave the spreadsheet open

Set objWorksheet = nothing
Set objWorkbook = nothing
Set objExcel = nothing

Wscript.quit
```

# VBScript Examples

## *Copy an Excel Spreadsheet*

This script will copy selected data from one spreadsheet into another.

### Features:

- Delete a file
- Copy a spreadsheet

```
Option Explicit
Const OPFN = "U:\Assignments\_Open\A002\Template2.xls"
Dim objExcel, objFSO, objFileCopy, objWorkBookI, objWorkBookO, objWorksheetI, objWorksheetO

Proc01           ' Initialization
Proc02           ' Copy the output spreadsheet to a new one
Proc03           ' Populate the output spreadsheet
Proc04           ' Finalization
WScript.quit

'*-----*
'* Proc01 - Initialization
'*-----*
Sub Proc01
End sub

'*-----*
'* Proc02 - Copy the output spreadsheet to a new one
'*-----*
Sub Proc02
    Const IPFN = "U:\Assignments\_Open\A002\Template.xls"
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set objFileCopy = objFSO.GetFile(IPFN)
    if objFSO.FileExists(OPFN) then
        objFSO.DeleteFile(OPFN)
    end if
    objFileCopy.Copy (OPFN)
End sub

'*-----*
'* Proc03 - Populate the output spreadsheet
'*-----*
Sub Proc03
    Const filename = "U:\Assignments\_Open\A002\FR's from Copy of Requirements from DR1412 20100708 v
1 1.xls"
    Dim Seq, FrNo, Paragraph, Title, Description, CQI, strModule, Status, row
    Set objExcel = CreateObject("Excel.Application")           ' Start the application
    Set objWorkBookI = objExcel.Workbooks.Open(filename)
    Set objWorksheetI = objWorkBookI.Worksheets(1)
    Set objWorkBookO = objExcel.Workbooks.Open(OPFN)
    Set objWorksheetO = objWorkBookO.Worksheets(1)

    For row = 2 to objWorksheetI.UsedRange.Rows.Count
        Seq = (objWorksheetI.Cells(row,1).Value)
        FrNo = (objWorksheetI.Cells(row,2).Value)
        Paragraph = (objWorksheetI.Cells(row,3).Value)
        Title = (objWorksheetI.Cells(row,4).Value)
        Description = (objWorksheetI.Cells(row,5).Value)
        CQI = (objWorksheetI.Cells(row,6).Value)
        strModule = (objWorksheetI.Cells(row,7).Value)
```

## VBScript Examples

```
Status = (objWorksheetI.Cells(row,8).Value)

' Move the fields to the output spreadsheet
objWorksheetO.Cells(row+7,1).Value = Seq
objWorksheetO.Cells(row+7,4).Value = FRNo
objWorksheetO.Cells(row+7,5).Value = Title
objWorksheetO.Cells(row+7,6).Value = Description
Next
objWorkBookI.close
'objWorkBookO.close
objExcel.Visible = true
End sub

'*-----*
'* Proc04 - Finalization
'*-----*
Sub Proc04
' Free the memory
MsgBox "Complete",,wscript.scriptname
End sub
```

# VBScript Examples

## *Read a Spreadsheet; Write a Spreadsheet*

This script will read a spreadsheet, and based on the contents, create a different spreadsheet.

### Features:

- Process the file name from the command line (which happens when an icon is dragged and dropped)
- Delete a file
- Read a spreadsheet
- Create a spreadsheet

```
Option Explicit
Dim objExcel, objFSO, objFileCopy, objWorkBookI, objWorksheetI, objWorkBookO, objWorkSheetO
Dim IRow, ORow, Comments, Initials, objRange
Dim Msg, strUAMsg, ctrFnd, IPFN, OPFN, ActItem, strReqNo, ctrIP, ctrFailed

'*-----*
'* Open the input Excel document, and the output Excel document
'*-----*
If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a file to the icon!"
    wscript.quit
end if

Set objFSO = CreateObject("Scripting.FileSystemObject")
IpFN = trim(wscript.arguments(0))           ' Complete input file name
OpFN = IPFN & "New.xls"                     ' Complete output application file name
Set objExcel = CreateObject("Excel.Application") ' Start the application
Set objWorkBookI = objExcel.Workbooks.Open(IPFN)
Set objWorksheetI = objWorkBookI.Worksheets(1)
Set objWorkBookO = objExcel.Workbooks.Add
Set objWorksheetO = objWorkBookO.Worksheets(1)

'*-----*
'* Read the spreadsheet
'*-----*
ctrIP = 0 : ctrFnd = 0 : ctrFailed = 0 : strUAMsg = "" : oRow = 0

' The columns in the spreadsheet:
' A - Item Number
' B - Review Technique
' C - Finding
' D - Requirement Number
' G - Testability- PASS or FAIL
' H - Reviewer's Comments
' I - Reviewer's initials
For IRow = 9 to objWorksheetI.UsedRange.Rows.Count
    ' Assign field names to some cells
    strReqNo = objWorksheetI.Cells(IRow,4).Value
    ActItem = objWorksheetI.Cells(IRow,5).Value
    Comments = objWorksheetI.Cells(IRow,8).Value
    Initials = objWorksheetI.Cells(IRow,9).Value

    If (trim(objWorksheetI.Cells(IRow,1).Value) = "") then ' Item number is empty
        exit for
    end if
    If (trim(objWorksheetI.Cells(IRow,1).Value) = "TOTAL:") then ' Item number
        exit for
    end if
end for
```

## VBScript Examples

```
end if

ctrIP = ctrIP + 1          ' Count IRows processed

' Display where testability is empty
If (trim(objWorksheetI.Cells(IRow,7).Value) = "") then
    strUAMsg = strUAMsg & "Req= " & strReqNo & " (Row=" & IRow & ")" & _
        "; Testability was not completed." & vbcrLf
    ctrFnd = ctrFnd + 1
end if

' Display where Reviewer's Initials cell is empty
If (trim(objWorksheetI.Cells(IRow,9).Value) = "") then
    strUAMsg = strUAMsg & "Req= " & strReqNo & " (Row=" & IRow & ")" & _
        "; Reviewer's initials were not supplied." & vbcrLf
    ctrFnd = ctrFnd + 1
end if

' If Failed, make sure there is something in cols C, H, and I
If (trim(objWorksheetI.Cells(IRow,7).Value) = "FAIL") then
    If (trim(objWorksheetI.Cells(IRow,3).Value) = "") then
        strUAMsg = strUAMsg & "Req= " & strReqNo & " (Row=" & IRow & ")" & _
            "; Finding is missing." & vbcrLf
        ctrFnd = ctrFnd + 1
    end if
    If (trim(objWorksheetI.Cells(IRow,8).Value) = "") then
        strUAMsg = strUAMsg & "Req= " & strReqNo & " (Row=" & IRow & ")" & _
            "; Reviewer's Comments missing." & vbcrLf
        ctrFnd = ctrFnd + 1
    end if
    If (trim(objWorksheetI.Cells(IRow,9).Value) = "") then
        strUAMsg = strUAMsg & "Req= " & strReqNo & " (Row=" & IRow & ")" & _
            "; Reviewer's Initials are missing." & vbcrLf
        ctrFnd = ctrFnd + 1
    end if
end if

' If Passed, make sure there is NOTHING in col C
If (trim(objWorksheetI.Cells(IRow,7).Value) = "PASS") then
    If (trim(objWorksheetI.Cells(IRow,3).Value) <> "") then
        strUAMsg = strUAMsg & "Req= " & strReqNo & "; Finding supplied for passed requirement." &
vbcrLf
        ctrFnd = ctrFnd + 1
    end if
end if

' Now list the failed items to the output spreadsheet
If (objWorksheetI.Cells(IRow,7).Value) = "FAIL" then
    ctrFailed = ctrFailed + 1 : oRow = oRow + 1
    objWorksheetO.Cells(oRow,1).Value = strReqNo
    objExcel.Columns(2).ColumnWidth = 60
    objWorksheetO.Cells(oRow,2).Value = Comments
    set objRange = objExcel.Range("B1").EntireColumn
    objRange.WrapText = true
    objWorksheetO.Cells(oRow,3).Value = Initials
end if
Next
objWorkBookI.close
objFSO.deleteFile(OPFN)
objWorkbookO.SaveAs(OPFN)
objExcel.visible = true

Msg = "I processed " & ctrIP & " input rows." & vbcrLf
```

## VBScript Examples

```
Msg = Msg & "I found " & ctrFnd & " problems." & vbCrLf
If ctrFnd > 0 then
    Msg = Msg & "They are: " & vbCrLf & strUAMsg
end if
Msg = Msg & ctrFailed & " requirements were failed, and written to the output spreadsheet." &
vbCrLf
MsgBox Msg,,wscript.scriptname
```

# VBScript Examples

## *Divide Big Cells*

This script will check for cells that are “too large” to be displayed (or printed?).

It is non-destructive, so you can run it at will.

It checks every cell for text that is greater than 1000 bytes in total. It will divide those cells into chunks of approximately 1000 bytes, and write these to a text file. It will then open Notepad. At this point, you can cut and paste the sections of text into the source spreadsheet. All lines are written in their entirety (no lines are broken up).

Features:

- Create a text file
- Open the output text file in Notepad

```
Option Explicit
Dim objExcel, objWorkbook, objWorksheet, row, col, msg, ThisTxt, ThisLen, ctrCells, objFSO
Dim ipFN, OPFN, objTextFile, ipLen, opLen, WshShell
Const swDebug = false

If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a file to the icon!"
    wscript.quit
end if

msg = ""

ipFN = trim(wscript.arguments(0))           ' Complete input file name
Set objFSO = CreateObject("Scripting.FileSystemObject")
OPFN = ipFN & ".DivideBigCells.txt"
Set objTextFile = objFSO.CreateTextFile (OPFN, True)

' Start the application
Set objExcel = CreateObject("Excel.Application")
objExcel.Application.visible = false        ' make Excel invisible
Set objWorkbook = objExcel.Workbooks.Open(ipFN)
Set objWorksheet = objWorkbook.Worksheets(1) '<---- Hardcode the worksheet number

ctrCells = 0
Const Threshold = 1000
For Row = 1 to objWorksheet.UsedRange.Rows.Count
    For col = 1 to 9
        ctrCells = ctrCells + 1
        ThisTxt = (objWorksheet.Cells(row,col).Value)
        ThisLen = len(ThisTxt)
        If ThisLen > Threshold then
            DivideCell
        end if
    next
next
msg = msg & "I checked " & ctrCells & " cells." & vbcrLf
MsgBox (msg)
objExcel.Application.quit                 ' Close the spreadsheet

Set objWorksheet = nothing
Set objWorkbook = nothing
```

## VBScript Examples

```
Set objExcel = nothing
objTextFile.close

' Open Notepad, and write the new stuff
Set WshShell = WScript.CreateObject("WScript.Shell")
WshShell.Run ("%windir%\notepad" & " " & OPFN)

Wscript.quit

'*-----*
Sub DivideCell
'*-----*
Dim opLine, i, j, strThisLine, opPartNo
opLine = "Row=" & row & "; col=" & col & "; length=" & ThisLen
objTextFile.writeline(OpLine)
objTextFile.writeline("Part 1 " & "-----")
opLine = "" : ipLen = 0 : OpLen = 0 : opPartNo = 1
For i = 1 to len(ThisTxt)
    ipLen = ipLen + 1
    If mid(ThisTxt,i,1) = vbLF then
        ipLen = ipLen - 1
        If swDebug then
            opLine = "Line len=" & ipLen & "; line=" & " " & opline
        end if
        objTextFile.writeline(OpLine) : OpLine = ""
        opLen = opLen + ipLen
        If opLen > 1000 then
            objTextFile.writeline
            opLen = 0
            opPartNo = opPartNo + 1
            objTextFile.writeline("Part " & opPartNo & " -----")
        end if
        ipLen = 0
    else
        opLine = opLine & mid(ThisTxt,i,1)
    end if
Next
opLine = "-----"
objTextFile.writeline(OpLine)

End Sub
```

## VBScript Examples

### *Check to See if a Worksheet Exists*

```
Set objWorkbook = objExcel.Workbooks.Open(ssVF)
'* Check to see if the worksheet name you supplied is valid.
WSExists = 0
For Each objWorksheet in objWorkbook.Worksheets
  If objWorksheet.Name = wkshname Then
    WSExists = 1
    Exit For
  End If
Next
If wSExists = 0 Then 'Doesn't exist
  MsgBox wkshName & ": Worksheet does not exist in the spreadsheet", vbCritical, WScript.Scriptname
wscript.quit
End If
```

## VBScript Examples

### *Put a Border Around Cells*

```
Const xlPasteValues = -4163      '(&HFFFFFFBD)
Const xlPasteFormats = -4122
Const xlPasteSpecialOperationNone = -4142 '(&HFFFFFFD2)
With objExcel
    .Range("B2").Select
    .Selection.Copy
    .Range("C2:BB200").Select
    .Selection.PasteSpecial xlPasteFormats, xlPasteSpecialOperationNone, False, False
    .Application.CutCopyMode = False
end with
```

# VBScript Examples

## VBScript Examples

### *Align Cells at the Top*

```
Const xlTop = -4160
objSheet.Range("A1:B999").VerticalAlignment = xlTop
```

### *Highlight a Cell*

```
' 1=black 2=nothing 3=red 4=green 5=blue 6=yellow
objWorkSheetB.Cells(RowB,10).Interior.ColorIndex = 6
```

### *Create and process a RecordSet*

```
' First, add an array to a recordset
Const adVarChar = 200 'the SQL datatype is varchar
Const adOpenStatic = 3
Const adUseClient = 3
const adChar = 129

Dim rs : Set rs = CreateObject("ADODB.RECORDSET")
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.Fields.Append "ArEntry", adChar, 16 'String with 16 characters
rs.open

For I = 1 to ubound(strArray)
    rs.AddNew
    rs("ArEntry") = strArray(i)
    rs.Update
next

' Now we sort it
rs.sort = "ArEntry"

'* Delete duplicate entries from the array
Dim strLastEntry, strThisEntry
strLastEntry = ""
Do Until rs.EOF
    strThisEntry = rs.Fields.Item("ArEntry")
    If strComp(strLastEntry, strThisEntry, vbTextCompare) = 0 then
        rs.delete
    else
        strLastEntry = strThisEntry
    end if
    rs.MoveNext
Loop
```

## VBScript Examples

### ***VBScripts for Microsoft Word***

#### *Create Word Combinations*

This script will create a list of all of the 4-letter combinations within a Word document. This is strictly a learning exercise, and has no other value.

Note: There is code in the program to stop processing after a set number of words are created. If you let the entire program process, it will (try to) create almost a half million lines.

#### Features:

- Add lines to a Microsoft Word document
- Create a line set to a specific font name and font size

```
' Words.VBS - Create a list of all 4-letter words
' This program will add these words to a Microsoft Word document.
Option Explicit
Const Alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Dim objWord, ctrWord
Dim L1, L2, L3, L4

ctrWord = 0

Set objWord = CreateObject("Word.Application")    ' Launch Microsoft Word
objWord.Documents.Add                            ' Create a new document

' Making Word visible will allow you to watch as the words are added to the document, but it
' SIGNIFICANTLY slows down the processing.
objWord.Visible = True

' All two-letter combinations:  26**2 =    676
' All three-letter combinations: 26**3 =  17,576
' All four-letter combinations:  26**4 = 456,976
Do
  For L1 = 1 to 26
    For L2 = 1 to 26
      For L3 = 1 to 26
        For L4 = 1 to 26
          ctrWord = ctrWord + 1
          objword.selection.font.name = "Times New Roman"
          objword.selection.font.size = 10
          objWord.Selection.TypeText ctrWord & vbTab

          objword.selection.font.name = "Courier New"
          objword.selection.font.size = 12
          objWord.Selection.TypeText mid(Alphabet,L1,1) & _
                                     mid(Alphabet,L2,1) & _
                                     mid(Alphabet,L3,1) & _
                                     mid(Alphabet,L4,1) & _
                                     vbCrLf

          If ctrWord > 17578 then          ' DON'T PROCESS THE ENTIRE FILE
            exit do
          end if
        next
      next
    next
  next
next
Loop
```

## VBScript Examples

```
objWord.Visible = True  
  
'Wscript.quit  
'objWord.Selection.movedown  
'objWord.Selection.typeparagraph()
```

# VBScript Examples

## *Create Word Tables*

This script will programmatically create a table inside a Microsoft Word document.

### Features:

- Create a new Microsoft Word document

```
' Create Word Tables.vbs - Create a table programmatically in a NEW Word document
Dim objWord, objDoc, objRange, objTable, i, j

Const NUMBER_OF_ROWS = 7
Const NUMBER_OF_COLUMNS = 3

Set objWord = CreateObject("Word.Application")
objWord.Visible = True
Set objDoc = objWord.Documents.Add()           ' Create a NEW document

Set objRange = objDoc.Range()
objDoc.Tables.Add objRange, NUMBER_OF_ROWS, NUMBER_OF_COLUMNS

Set objTable = objDoc.Tables(1)              ' Work with the first (and only) table

For i = 1 to NUMBER_OF_ROWS
  For j = 1 to NUMBER_OF_COLUMNS
    objTable.Cell(i, j).Range.Text = "Row " & i & ", Column " & j
    objTable.Cell(i, j).Range.Text = "Row " & i & ", Column " & j
    objTable.Cell(i, j).Range.Text = "Row " & i & ", Column " & j
  next
next
'ObjTable.Rows.add()                        ' Add a blank row

objTable.AutoFormat(16)
```

# VBScript Examples

## *Display Word Properties*

This script provides an insight into some of the information that is available while inside a Word document.

```
' List Microsoft Word Properties
Option Explicit
On Error Resume Next          ' Since we cannot trap the errors that occur
Dim objWord, objAddIn, objCaption, objDictionary, objDocument, objConverter, objFont
Dim objLabel, objLanguage, objFile, objWindow, objBinding, objTask, objTemplate, objDialog
Dim ctr, svFontname, svFontSize

Set objWord = CreateObject("Word.Application")
objWord.Documents.Add          ' Create a new document
objWord.Visible = True

objWord.Selection.TypeText "Active Printer: " & objWord.ActivePrinter & vbCRLF

For Each objAddIn in objWord.AddIns
    objWord.Selection.TypeText "AddIn: " & objAddIn & vbCRLF
Next

objWord.Selection.TypeText "Application: " & objWord.Application & vbCRLF
objWord.Selection.TypeText "Assistant: " & objWord.Assistant & vbCRLF

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "AutoCaptions (" & objWord.AutoCaptions.Count & "):" & vbCRLF
ctr = 0
For Each objCaption in objWord.AutoCaptions
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objCaption & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

objWord.Selection.TypeText "Automation Security: " & objWord.AutomationSecurity & vbCRLF
objWord.Selection.TypeText "Background Printing Status: " & objWord.BackgroundPrintingStatus &
vbCRLF
objWord.Selection.TypeText "Background Saving Status: " & objWord.BackgroundSavingStatus & vbCRLF
objWord.Selection.TypeText "Browse Extra File Type: " & objWord.BrowseExtraFileTypes & vbCRLF
objWord.Selection.TypeText "Build: " & objWord.Build & vbCRLF
objWord.Selection.TypeText "Caps Lock: " & objWord.CapsLock & vbCRLF
objWord.Selection.TypeText "Caption: " & objWord.Caption & vbCRLF

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "Caption Labels (" & objWord.CaptionLabels.Count & "):" & vbCRLF
ctr = 0
For Each objLabel in objWord.CaptionLabels
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objLabel & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

objWord.Selection.TypeText "Check Language: " & objWord.CheckLanguage & vbCRLF

For Each objAddIn in objWord.COMAddIns
    objWord.Selection.TypeText "COM AddIn: " & objAddIn & vbCRLF
Next

objWord.Selection.TypeText "Creator: " & objWord.Creator & vbCRLF

For Each objDictionary in objWord.CustomDictionaries
```

## VBScript Examples

```
objWord.Selection.TypeText "Custom Dictionary: " & objDictionary & vbCRLF
Next

objWord.Selection.TypeText "Customization Context: " & objWord.CustomizationContext & vbCRLF
objWord.Selection.TypeText "Default Legal Blackline: " & objWord.DefaultLegalBlackline & vbCRLF
objWord.Selection.TypeText "Default Save Format: " & objWord.DefaultSaveFormat & vbCRLF
objWord.Selection.TypeText "Default Table Separator: " & objWord.DefaultTableSeparator & vbCRLF

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "Dialogs (" & objWord.Dialogs.Count & "), not listed." & vbCRLF
' Displaying these dialogues as above does funny things to the Word document.
' #68 (199) will wipe out the text in the document and start all over.
' Another will do something with Outlook.
'objWord.Dialogs(80).Show           ' Word help dialogue
objWord.Selection.TypeText "Display Alerts: " & objWord.DisplayAlerts & vbCRLF
objWord.Selection.TypeText "Display Recent Files: " & objWord.DisplayRecentFiles & vbCRLF
objWord.Selection.TypeText "Display Screen Tips: " & objWord.DisplayScreenTips & vbCRLF
objWord.Selection.TypeText "Display Scroll Bars: " & objWord.DisplayScrollBars & vbCRLF

For Each objDocument in objWord.Documents
    objWord.Selection.TypeText "Document: " & objDocument & vbCRLF
Next

objWord.Selection.TypeText "Email Template: " & objWord.EmailTemplate & vbCRLF
objWord.Selection.TypeText "Enable Cancel Key: " & objWord.EnableCancelKey & vbCRLF
objWord.Selection.TypeText "Feature Install: " & objWord.FeatureInstall & vbCRLF

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "File Converters (" & objWord.FileConverters.Count & "):" & vbCRLF
ctr = 0
For Each objConverter in objWord.FileConverters
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objConverter & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

objWord.Selection.TypeText "Focus In MailHeader: " & objWord.FocusInMailHeader & vbCRLF

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "Font names (" & objWord.Fontnames.Count & "):" & vbCRLF
ctr = 0
For Each objFont in objWord.FontNames
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objFont & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

objWord.Selection.TypeText "Height: " & objWord.Height & vbCRLF

' This isn't working. The system complains about objBinding.
'For Each objBinding in objWord.KeyBindings
'    objWord.Selection.TypeText "Key Binding: " & objBinding & vbCRLF
'Next

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "LandscapeFontnames (" & objWord.LandscapeFontnames.Count & "):" &
vbCRLF
ctr = 0
For Each objFont in objWord.LandscapeFontNames
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objFont & vbCRLF
Next
objWord.Selection.TypeText vbCrLf
```

## VBScript Examples

```
objWord.Selection.TypeText "Languages (" & objWord.Languages.Count & "):" & vbCRLF
ctr = 0
For Each objLanguage in objWord.Languages
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objLanguage & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

objWord.Selection.TypeText "Left" & objWord.Left & vbCRLF
objWord.Selection.TypeText "Mail System: " & objWord.MailSystem & vbCRLF
objWord.Selection.TypeText "MAPI Available: " & objWord.MAPIAvailable & vbCRLF
objWord.Selection.TypeText "Math Coprocessor Available: " & objWord.MathCoprocessorAvailable &
vbCRLF
objWord.Selection.TypeText "Mouse Available: " & objWord.MouseAvailable & vbCRLF
objWord.Selection.TypeText "Name: " & objWord.Name & vbCRLF
objWord.Selection.TypeText "Normal Template: " & objWord.NormalTemplate & vbCRLF
objWord.Selection.TypeText "Num Lock: " & objWord.NumLock & vbCRLF
objWord.Selection.TypeText "Parent: " & objWord.Parent & vbCRLF
objWord.Selection.TypeText "Path: " & objWord.Path & vbCRLF
objWord.Selection.TypeText "Path Separator: " & objWord.PathSeparator & vbCRLF
objWord.Selection.TypeText "Print Preview: " & objWord.PrintPreview & vbCRLF

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "Recent Files (" & objWord.RecentFiles.Count & "):" & vbCRLF
ctr = 0
For Each objFile in objWord.RecentFiles
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objFile & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

objWord.Selection.TypeText "Screen Updating: " & objWord.ScreenUpdating & vbCRLF
objWord.Selection.TypeText "Show Visual Basic Editor: " & objWord.ShowVisualBasicEditor & vbCRLF
objWord.Selection.TypeText "Special Mode: " & objWord.SpecialMode & vbCRLF

svFontName = objword.selection.font.name
svFontSize = objword.selection.font.size
objWord.Selection.TypeText "Startup Path: "
objword.selection.font.name = "Courier New"
objword.selection.font.size = 8
objWord.Selection.TypeText objWord.StartupPath & vbCRLF
objword.selection.font.name = svFontName
objword.selection.font.size = svFontSize

objWord.Selection.TypeText vbCrLf
objWord.Selection.TypeText "Tasks (" & objWord.Tasks.Count & "):" & vbCRLF
ctr = 0
For Each objTask in objWord.Tasks
    ctr = ctr + 1
    objWord.Selection.TypeText "- " & ctr & ": " & objTask & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

For Each objTemplate in objWord.Templates
    objWord.Selection.TypeText "Template: " & objTemplate & vbCRLF
Next
objWord.Selection.TypeText vbCrLf

objWord.Selection.TypeText "Top: " & objWord.Top & vbCRLF
objWord.Selection.TypeText "Usable Height: " & objWord.UsableHeight & vbCRLF
objWord.Selection.TypeText "Usable Width: " & objWord.UsableWidth & vbCRLF
objWord.Selection.TypeText "User Address: " & objWord.UserAddress & vbCRLF
```

## VBScript Examples

```
objWord.Selection.TypeText "User Control: " & objWord.UserControl & vbCRLF
objWord.Selection.TypeText "User Initials: " & objWord.UserInitials & vbCRLF
```

```
objWord.Selection.TypeText "User Name: "
svFontName = objword.selection.font.name
svFontSize = objword.selection.font.size
objword.selection.font.name = "Times New Roman"
objword.selection.font.size = 18
objWord.Selection.TypeText objWord.UserName & vbCRLF
objword.selection.font.name = svFontname
objword.selection.font.size = svFontSize
```

```
objWord.Selection.TypeText "Version: " & objWord.Version & vbCRLF
objWord.Selection.TypeText "Visible: " & objWord.Visible & vbCRLF
objWord.Selection.TypeText "Width: " & objWord.Width & vbCRLF
```

```
For Each objWindow in objWord.Windows
    objWord.Selection.TypeText "Window: " & objWindow & vbCRLF
Next
```

```
objWord.Selection.TypeText "Window State: " & objWord.WindowState & vbCRLF
objWord.Quit
objWord.Selection.TypeText "*--- END OF LIST ---*" & vbCRLF
WScript.quit
```

```
Sub CheckError(ErrSrc)
If Err.Number <> 0 then
    Msg = "Exception:" & vbCrLf & _
        "    Error number: " & Err.Number & vbCrLf & _
        "    Error description: '" & Err.Description & vbCrLf & _
        "    Source" & ErrSrc
    MsgBox Msg, "Error?"
end if
End Sub
```

# VBScript Examples

## Variable Name Summary

This program will list variable names: where and how they are defined, and the instructions that reference those variable names. The input is an Assembler Language compiled source listing.

### Features:

- Add Microsoft Word headers and footers, complete with running page numbers.
- Load lines from an input text file into a fixed array.
- Set the output page margins on the Word document.
- Input file name is supplied on the command line, or alternately, by dragging and dropping the input file name in Explorer onto the name of the script.

```
Option Explicit
Dim ipPath, objFSO, Msg, arStmts(20000), ipLine, objWord, ctrStmts
Dim objDoc
Const OPLIMIT = 999          '<-----'
Const wdSeekPrimaryFooter = 4
Const wdSeekPrimaryHeader = 1
Const wdSeekMainDocument = 0
Const wdAlignParagraphCenter = 1
Const wdFieldPage = 33
Const wdFieldNumPages = 26

ProcInit          ' Initialization
Proc01            ' Add all of the Assembler statements to an array
Proc02            ' Process the variables (via the cross-reference)

MsgBox Msg,,Wscript.Scriptname & " completed."
Set objFSO = nothing
WScript.quit

'*-----*
'* Initialization
'*-----*
Sub ProcInit
If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a file to the icon!"
    wscript.quit
end if

IpPath = wscript.arguments(0)          ' Complete input file name
Msg = "IpFile: " & IPPath & vbCRLF
Set objFSO = CreateObject("Scripting.FileSystemObject")

Set objWord = CreateObject("Word.Application")          ' Launch Microsoft Word
Set objDoc = objWord.Documents.Add                    ' Create a new document
objWord.Visible = True
objword.ActiveDocument.pagesetup.leftmargin = 50      ' Pixels???
objword.ActiveDocument.pagesetup.rightmargin = 50     ' Pixels???

' Header
objWord.ActiveWindow.ActivePane.View.SeekView = wdSeekPrimaryHeader
objword.selection.font.name = "Times New Roman"
objword.selection.font.size = 14
objWord.selection.Paragraphs.Alignment = wdAlignParagraphCenter
objWord.Selection.TypeText IPPath
```

## VBScript Examples

```
ProcFooter
End Sub
```

```
'*-----*
'* Add all of the Assembler statements to an array
'*-----*
Sub Proc01
Dim objIpFile, ipLine, swInStmts, ctrIP, stmtno
Set objIpFile = objFSO.OpenTextFile(IPPath)
ctrIp = 0 : ctrStmts = 0
swInStmts = "N"
Do Until objIPFile.AtEndOfStream
    ipLine = RTrim(objIPFile.ReadLine) : CtrIP = CtrIP + 1

    if swInStmts = "N" then
        If mid(ipLine,37,4) = "STMT" then
            swInStmts = "Y"
        end if
    end if
    if swInStmts = "Y" then
        If mid(ipLine,51,21) = "RELOCATION DICTIONARY" then
            swInStmts = "X"           ' We are done
        end if
    end if
    if swInStmts = "Y" then
        ' Lines to skip
        if (mid(ipLine,42,1) = "*") or _
            (left(ipLine,1) = "1") or _
            (mid(ipLine,4,14) = "ACTIVE USINGS:") then
        else
            stmtno = mid(ipLine,36,5)
            If IsNumeric(stmtno) then
                ctrStmts = ctrStmts + 1
                arStmts(stmtno) = mid(ipLine,36,78) ' Add it to the nth entry
            end if
        end if
    end if
loop
objIPfile.Close()
Set objIPFile = nothing
Msg = "Records read:  " & ctrIP & vbcrLf
Msg = Msg & "arStmts added: " & ctrStmts & vbcrLf
End Sub
```

```
'*-----*
'* Process the variables (via the Cross-Reference)
'*-----*
Sub Proc02
Dim objIpFile, swInXref, ctrIP, ctrXRef
Set objIpFile = objFSO.OpenTextFile(IPPath)
ctrIp = 0
swInXRef = "N"

ctrXref = 0

Do Until objIPFile.AtEndOfStream
    ipLine = RTrim(objIPFile.ReadLine) : CtrIP = CtrIP + 1
    if swInXref = "N" then
        If mid(ipLine,33,43) = "ORDINARY SYMBOL AND LITERAL CROSS REFERENCE" then
            swInXref = "Y"
        end if
    end if
    if swInXref = "Y" then
```

## VBScript Examples

```
If mid(IpLine,2,1) = "=" then                                ' First literal. We are done.
  'wscript.echo "I found the END of the cross-ref section (the first literal)"
  swInXref = "X"
  end if
end if

if swInXref = "Y" then
  ctrXRef = ctrXRef + 1
  ' Lines to skip
  if (left(IpLine,8) = "1      ") or _
    (left(IpLine,7) = "--SYMBOL") then
    else
      Proc021                                ' Process this variable
    end if
  end if
end if
if ctrXref > OPLIMIT then
  exit do
end if
loop
objIPfile.Close()
Set objIPFile = nothing
Msg = "XRef recs read:  " & ctrXRef & vbcrLf
End Sub

'*-----*
'* Process one variable
'*-----*
Sub Proc021
Dim ipLabel, ipLength, ipAddress, ipType, ipDefn, ipRefs, Msg2, i, strWk
Dim stmtNo, stmtbody, arWord, refNo
if mid(IpLine,2,8) <> space(8) then
  ipLabel = mid(IpLine,2,8)      ' Assembler label
end if
ipLength = mid(IpLine,12,5)     ' Length
ipAddress = mid(IpLine,18,8)    ' Address
ipType = mid(IpLine,40,1)      ' Type
ipDefn = mid(IpLine,56,6)      ' Defined at statement no.
if mid(IpLine,2,8) = space(8) then
  ipRefs = ipRefs & mid(IpLine,63,55)  ' References
else
  ipRefs = mid(IpLine,63,55)      ' References
end if

' Process the definition statement
If trim(ipDefn) <> "" then
  objword.selection.font.name = "Courier New"
  objword.selection.font.size = 10
  Msg2 = "Variable=" & trim(ipLabel) & "; "
  Msg2 = Msg2 & "length=" & ipLength & "; "
  Msg2 = Msg2 & "Address=" & ipAddress & "; "
  Msg2 = Msg2 & "type=" & ipType
  objWord.Selection.TypeText Msg2 & vbcrLf

  objword.selection.font.size = 9
  Msg2 = " DEFN (" & trim(ipdefn) & "):" & vbTab
  strWk = arStmts(ipdefn) & space(7)
  stmtbody = Right(strWk, len(strWk)-6)
  Msg2 = Msg2 & stmtbody
  objWord.Selection.TypeText Msg2 & vbcrLf
end if

' Process the references now.
IpRefs = trim(IpRefs)
```

## VBScript Examples

```
arWord = Split(IpRefs)

for each RefNo in arWord
  Refno = trim(Refno)
  If Refno <> "" then
    If right(RefNo,1) = "B" then Refno = left(RefNo,len(RefNo) - 1)
    If right(RefNo,1) = "D" then Refno = left(RefNo,len(RefNo) - 1)
    If right(RefNo,1) = "M" then Refno = left(RefNo,len(RefNo) - 1)
    If right(RefNo,1) = "U" then Refno = left(RefNo,len(RefNo) - 1)
    objword.selection.font.size = 9
    Msg2 = " REF (" & RefNo & "):" & vbTab
    strWk = arStmts(RefNo) & space(7)
    stmtbody = Right(strWk,len(strWk)-6)
    Msg2 = Msg2 & stmtbody
    objWord.Selection.TypeText Msg2 & vbcrLf
  end if
next
objWord.Selection.TypeText vbcrLf
End Sub

'*-----*
'* Process the Footer
'*-----*
Sub ProcFooter
Dim strWk
With objWord
  .ActiveWindow.ActivePane.View.SeekView = wdSeekPrimaryFooter
  .selection.font.size = 08
  .selection.Paragraphs.Alignment = wdAlignParagraphCenter
  .Selection.TypeText Now() & " - Page "
  .Selection.Fields.Add .Selection.Range,wdFieldPage
  .Selection.TypeText " of "
  .Selection.Fields.Add .Selection.Range,wdFieldNumPages
end with

'objWord.Selection.TypeText strwk

objWord.ActiveWindow.ActivePane.View.Seekview = wdSeekMainDocument
End Sub
```

## VBScript Examples

### *Copy a Text File into a Word Document*

This program will read a text file and copy it into a Word document. One particular feature of this program is that it inserts the file name, dynamically, into the footer.

Features:

- Read a text file.
- Write a Microsoft Word document.
- Use command-line path names (allowing drag-and-drop invocation).
- Add the document's file name (including pathname), last edited date, page number, and number of pages to the footer.

```
Option Explicit
Dim objFSO, objIPFile, FN, ipPath
Dim CtrIP, ctrOP, ipLine, OPLine, objWord, objDoc
Dim Msg, i
Const OpLimit = 1000 ' <-----
Const wdSeekPrimaryFooter = 4
Const wdSeekPrimaryHeader = 1
Const wdSeekMainDocument = 0
Const wdAlignParagraphCenter = 1
Const wdFieldPage = 33
Const wdFieldNumPages = 26
Const wdFieldEmpty = -1
Const wdFieldFileName = 29

If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a file on top of the icon!"
    wscript.quit
end if

ipPath = wscript.arguments(0) ' Complete input file name

Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objIpFile = objFSO.OpenTextFile(IPPath)

Set objWord = CreateObject("Word.Application") ' Launch Microsoft Word
Set objDoc = objWord.Documents.Add ' Create a new document
objWord.Visible = True
objword.ActiveDocument.pagesetup.leftmargin = 50 ' Pixels???
objword.ActiveDocument.pagesetup.rightmargin = 50 ' Pixels???

ProcHeader
ProcFooter

CtrIP = 0 : CtrOP = 0

Do Until objIPFile.AtEndOfStream
    ipLine = RTrim(objIPFile.ReadLine) : CtrIP = CtrIP + 1
    If (left(ipLine,1) = "<") or _
        (instr(1,ipLine,"<html>",vbTextCompare) > 0) or _
        (instr(1,ipLine,"<tt>",vbTextCompare) > 0) or _
        (instr(1,ipLine,"<td w",vbTextCompare) > 0) then
        ctrSkipped = ctrSkipped + 1
    else
        opLine = ipLine
        objword.selection.font.size = 12
        objWord.selection.TypeText opLine & vbcrLf
    end if
end do
```

## VBScript Examples

```
        CtrOP = CtrOP + 1
    if ctrOP > opLimit then
        wscript.quit
    end if
end if
loop

objIPfile.Close()

Msg = ""
Msg = Msg & "IpFile: " & IPPath & vbCRLF
Msg = Msg & CtrIP & " records read. " & vbCRLF
MsgBox Msg,,Wscript.Scriptname

Set objFSO    = nothing
Set objIPFile = nothing
Set objOPFile = nothing
wscript.quit

'*-----*
'* Process the Header
'*-----*
Sub ProcHeader
With objWord
    .ActiveWindow.ActivePane.View.SeekView = wdSeekPrimaryHeader
    .selection.font.name = "Times New Roman"
    .selection.Paragraphs.Alignment = wdAlignParagraphCenter
    .selection.font.size = 14
    .Selection.TypeText "Module Linkage Report" & vbcrLf
    .selection.font.size = 12
    .Selection.TypeText "Input: " & IPPath
end with
End sub

'*-----*
'* Process the Footer
'*-----*
Sub ProcFooter
With objWord
    .ActiveWindow.ActivePane.View.SeekView = wdSeekPrimaryFooter
    .selection.font.size = 08
    .selection.Paragraphs.Alignment = wdAlignParagraphCenter
    .Selection.Fields.Add .Selection.Range,wdFieldFileName,"\p",false
    .Selection.TypeText " - " & Now() & " - Page "
    .Selection.Fields.Add .Selection.Range,wdFieldPage
    .Selection.TypeText " of "
    .Selection.Fields.Add .Selection.Range,wdFieldNumPages
end with
objWord.ActiveWindow.ActivePane.View.Seekview = wdSeekMainDocument
End Sub
```

# VBScript Examples

## *Process a Word Document*

This program will read a Microsoft Word document, clean up the special characters, and load it into a string array. It will add linefeeds where necessary. Finally, it will write the file to a text file.

### Features:

- Read a Microsoft Word document.
- Use command-line path names (allowing drag-and-drop invocation).
- Write a text file.

```
Option Explicit
Dim objFSO, objOPFile, objWord, objDoc, objSelection, objRange
Dim strDocument, strContents, strContents2
Dim FN, opPath, ThisChar
Dim Msg, i
Const wdDoNotSaveChanges = 0

If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a file on top of the icon!"
    wscript.quit
end if

strDocument = wscript.arguments(0)                ' Complete input file name
Set objWord = CreateObject("Word.Application")
objWord.Visible = False
objWord.DisplayAlerts = False                    ' Do not display file type warnings for older
versions
objWord.Documents.Open strDocument,,True        ' Open document in Read-only mode
Set objDoc = objWord.ActiveDocument            ' Create a reference to the current document

'Remove any Word-specific formatting
Set objSelection = objWord.Selection
objSelection.ClearFormatting

Set objRange = objDoc.Content                    ' Create a Range object of the file's contents
strContents = objRange.Text                     ' Grab the text within that range
strContents = objWord.CleanString(strContents)  ' Remove any non-printing characters such as
bullets
objWord.Quit wdDoNotSaveChanges                 'Close Word without saving changes to the document

' Word has created strContents, and separated each text line by ONLY a carriage return.
' We need to add line feeds.
strContents2 = ""
For i = 1 to len(strContents)
    ThisChar = mid(strContents,i,1)
    If ThisChar = vbCR then
        strContents2 = strContents2 & vbCrLf
    else
        strContents2 = strContents2 & ThisChar
    end if
Next

OpPath = "\\Dmvfsp08\mwdxg12$\Assignments\Misc\document.txt"
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objOPFile = objFSO.CreateTextFile(OPPath)
objOPFile.write(strContents2)
objOPFile.Close()
```

## VBScript Examples

```
Msg = "Command complete!"  
MsgBox Msg,,Wscript.Scriptname
```

```
Set objOPFile = nothing  
Set objFSO     = nothing  
Set objOPFile = nothing  
Set objSelection = nothing  
Set objDoc = nothing  
Set objWord = nothing
```

# VBScript Examples

## *Create Days of Year*

This program will generate a series of dates in the long date format.

Features:

- Add text to a Word document: bold, unbold
- Format a date

```
Option Explicit
Dim objWord, objDoc
Dim Msg, i, dtdate

Set objWord = CreateObject("Word.Application")           ' Launch Microsoft Word
Set objDoc = objWord.Documents.Add                       ' Create a new document
objWord.Visible = True

dtDate = #01/01/2011#
objWord.selection.font.size = 12
objWord.selection.font.name = "Times New Roman"
For i = 1 to 16
    objWord.selection.font.bold = true
    objWord.Selection.TypeText FormatDateTime(dtDate,vbLongDate) & vbcrLf
    objWord.selection.font.bold = false
    objWord.Selection.TypeText vbcrLf
    dtDate = dtDate + 1
Next

Msg = "Done"
MsgBox Msg,,Wscript.Scriptname

wscript.quit
```

# VBScript Examples

## *Print a Mainframe Listing*

This program will copy a mainframe listing to a Word document. It will process the ASA control character '1' by issuing a page break in Word. It will also process the '0' control character.

This script demonstrates some of the esoteric formatting of Word documents, like margins, orientation, etc.

```
Option Explicit
Const wdSeekPrimaryFooter = 4
Const wdSeekPrimaryHeader = 1
Const wdSeekMainDocument = 0
Const wdAlignParagraphCenter = 1
Const wdFieldPage = 33
Const wdFieldNumPages = 26
Const wdFieldEmpty = -1
Const wdFieldFileName = 29
Const wdPageBreak = 7
Const wdOrientPortrait = 0
Const wdOrientLandscape = 1
Const wdPrintView = 3

Dim objFSO, objIPFile, FN, ipPath, initFSO
Dim CtrIP, ipLine, OPLine, objWord, objDoc, ctrOP
Dim Msg, i, ctlchar

' Allow the user to choose an input file
Set ObjFSO = CreateObject("UserAccounts.CommonDialog")
ObjFSO.Filter = "An Assembler listing text file|*.txt"
objFSO.InitialDir = "C:\My Documents\source listings\"
InitFSO = ObjFSO.ShowOpen
If InitFSO = True Then
    IpPath = ObjFSO.FileName
else
    MsgBox "You did not select a file!",,wscript.Scriptname
    Wscript.Quit
End If

' Establish access to the input file
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objIpFile = objFSO.OpenTextFile(IPPath)

' Establish access to the output file (Word document)
Set objWord = CreateObject("Word.Application")           ' Launch Microsoft Word
Set objDoc = objWord.Documents.Add                       ' Create a new document
objWord.Visible = True

' Change to Print Layout View
objWord.ActiveWindow.Activepane.View.Type = wdPrintView

With objword.ActiveDocument.pagesetup
    .Orientation = wdOrientLandscape
    .LeftMargin = objword.InchesToPoints(0.17)
    .RightMargin = objword.InchesToPoints(0.17)
    .TopMargin = objword.InchesToPoints(0.25)
end with

CtrIP = 0 : ctrOP = 0

ProcHeader
ProcFooter
Do Until objIPFile.AtEndOfStream
    Objword.selection.font.name = "Courier New"
```

## VBScript Examples

```
objword.selection.font.size = 9
ipLine = RTrim(objIPfile.ReadLine) : CtrIP = CtrIP + 1
ctlchar = left(ipLine,1)

if ctlchar = "1" then
    if ctrIP > 1 then objword.Selection.InsertBreak(wdPageBreak)
end if
If ctlchar = "0" then
    objWord.selection.TypeText " " & vbcrLf          ' Blank line
    CtrOP = CtrOP + 1
end if
If len(ipLine) > 0 then
    opLine = right(ipLine,len(ipLine)-1)
else
    opLine = ""
end if
objWord.selection.TypeText opLine & vbcrLf
CtrOP = CtrOP + 1
loop

objIPfile.Close()

Msg = ""
Msg = Msg & "IpFile: " & IPPath & vbCRLF
Msg = Msg & "- " & CtrIP & " records read. " & vbCRLF
MsgBox Msg,,Wscript.Scriptname

Set objFSO = nothing
Set objIPfile = nothing
wscript.quit

'*-----*
'* Process the Header
'*-----*
Sub ProcHeader
exit sub          ' If you want a heading later, comment this statement
With objWord
    .ActiveWindow.ActivePane.View.SeekView = wdSeekPrimaryHeader
    .selection.font.name = "Times New Roman"
    .selection.Paragraphs.Alignment = wdAlignParagraphCenter
    .selection.font.size = 14
    .Selection.TypeText ipPath
    .selection.font.size = 12
end with
End sub

'*-----*
'* Process the Footer
'*-----*
Sub ProcFooter
With objWord
    .ActiveWindow.ActivePane.View.SeekView = wdSeekPrimaryFooter
    .selection.font.size = 08
    .selection.Paragraphs.Alignment = wdAlignParagraphCenter
    .Selection.Fields.Add .Selection.Range,wdFieldFileName,"\p",false
    .Selection.TypeText " - " & Now() & " - Page "
    .Selection.Fields.Add .Selection.Range,wdFieldPage
    .Selection.TypeText " of "
    .Selection.Fields.Add .Selection.Range,wdFieldNumPages
end with
objWord.ActiveWindow.ActivePane.View.Seekview = wdSeekMainDocument
End Sub
```

***VBScripts for Microsoft Outlook***

# VBScript Examples

## *Display Current Appointments*

This script will display the current month's calendar in a web page (HTML). It will then look at all of the appointments in the calendar, and display the ones that are current in the current week, the next week, and the week after that.

The program has set a switch to look at ALL occurrences of recurring tasks. This can amount to a high number, and will be an unlimited number if no end date is specified. Therefore, the program MUST provide a limit to the number of recurring items it looks at for each appointment. This one looks at the first 365, so it will see an item that started as far back as a year.

### Features:

- Create a monthly calendar
- Create a web page
- Interview all appointments in the calendar

```
Option Explicit
Const DefaultCalendarBorderColor = "#000000"
Const highlightcolor = "Yellow"

Dim objFSO, objTF
Dim weekstogo, borderSize, borderColor, monthName
Dim thisday, mydate, myday, iThisMonth, iThisYear, strFirstmdy
Dim iFirstDayofThisMonth, iDaysThisMonth, iDayToDisplay
Dim opLine, i, x, daysOffset, daysLeft, decWeekstogo, Msg
Dim iDayofYear, IE, wshShell, strOutput
Dim strWebPagename, tempfolder, iCount, oCount, MyItems, CurrAppt, dtThisMonday, dtTheSundayAfter

borderSize = 1
borderColor = DefaultCalendarBorderColor

thisday = Day(Date)
myDate = Date
myDay = DatePart("D", Date)
iThisMonth = Month(Date)
iThisYear = Year(Date)
strFirstmdy = (iThisMonth & "/1/" & iThisYear)
iFirstDayofThisMonth = DatePart("W", strFirstmdy)

' Store the month names into an array.
ReDim monthName(13)
monthName(0) = "Space"           'Set element 0 to garbage so I don't have to do math later
monthName(1) = "January"
monthName(2) = "February"
monthName(3) = "March"
monthName(4) = "April"
monthName(5) = "May"
monthName(6) = "June"
monthName(7) = "July"
monthName(8) = "August"
monthName(9) = "September"
monthName(10) = "October"
monthName(11) = "November"
monthName(12) = "December"

'Calculate number of days this month
```

## VBScript Examples

```
If iThisMonth = 12 Then
    iDaysThisMonth = DateDiff("d",strFirstmdy,("1/1/" & (iThisYear+1)))
Else
    iDaysThisMonth = DateDiff("d",strFirstmdy,((iThisMonth+1) & "/1/" & iThisYear))
End If

'*-----*&
'* Begin creation of the output file here
'*-----*&
Set ObjFSO = CreateObject("Scripting.FileSystemObject")
' We don't use "temporary" file names, because they really aren't, and they hang around forever,
' or until someone specifically deletes them.
strWebPageName = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & _
    "\VBSCalendar.html"
Set objTF = objFSO.CreateTextFile(strWebPageName)
objTF.writeline("<%@ LANGUAGE=""VBSSCRIPT"" %>")
objTF.writeline("<HTML>")
objTF.writeline("<HEAD>")
objTF.writeline("<TITLE>My Calendar</TITLE>")
objTF.writeline("</HEAD>")
objTF.writeline("<body bgcolor=""#FFFFFFD8"">")
Msg = FormatDateTime(Now(),vbLongDate)
objTF.writeline("<center><h1>" & msg & "</h1>")

OpLine = "<table cellpadding=""4"" cellspacing=""1"" border=""0"" bgcolor=""#ffffff"">"
OpLine = OpLine & "<tr><td colspan=""7""align=""center"" bgcolor=Yellow><b>"
OpLine = OpLine & (monthName(iThisMonth) & " " & iThisYear) & "</b></td></tr>"
objTF.writeLine(OpLine)

' Write the row of weekday initials
objTF.writeLine("<tr>")
objTF.writeLine("<td align=""center"" bgcolor=""#A1C6D1"">S</td>")
objTF.writeLine("<td align=""center"" bgcolor=""#A1C6D1"">M</td>")
objTF.writeLine("<td align=""center"" bgcolor=""#A1C6D1"">T</td>")
objTF.writeLine("<td align=""center"" bgcolor=""#A1C6D1"">W</td>")
objTF.writeLine("<td align=""center"" bgcolor=""#A1C6D1"">R</td>")
objTF.writeLine("<td align=""center"" bgcolor=""#A1C6D1"">F</td>")
objTF.writeLine("<td align=""center"" bgcolor=""#A1C6D1"">S</td>")
objTF.writeLine("</tr>")
objTF.writeLine("<tr>")

'Now write the first row
For i = 1 to 7
    if i = iFirstDayofThisMonth Then
        iDayToDisplay = 1
    elseif i > iFirstDayofThisMonth Then
        iDayToDisplay = iDayToDisplay + 1
    else
        iDayToDisplay=""&nbsp;";
    end if
    if iDayToDisplay = thisDay Then
        Msg = "<td align=center bgcolor=Yellow><b>" & iDayToDisplay & "</b></td>"
    else
        Msg = "<td align=center>" & iDayToDisplay & "</td>"
    end If
    objTF.writeLine(Msg)
Next

' Now, display the rest of the month.
' First figure out how many weeks are left to write
daysOffset = 8 - iFirstDayofThisMonth
daysLeft = iDaysThisMonth - daysOffset
decWeekstogo = Round((daysLeft/7),2)
```

## VBScript Examples

```
' I think this logic is screwy.
If decweekstogo > 4 Then
    weekstogo = 5
ElseIf decweekstogo > 3 and decweekstogo <= 4 Then
    weekstogo = 4
Else
    weekstogo = 3
End if

' Now write the rows and populate the data
For x = 1 To weekstogo
    objTF.WriteLine("<tr>")
    For i = 1 To 7
        If iDayToDisplay < iDaysThisMonth then
            iDayToDisplay = iDayToDisplay + 1
        else
            iDayToDisplay = "&nbsp;"
        End If
        if iDayToDisplay = thisDay then
            Msg = "<td align=center bgcolor=Yellow><b>" & iDayToDisplay & "</b></td>"
        else
            Msg = "<td align=center>" & iDayToDisplay & "</td>"
        end If
        objTF.WriteLine(Msg)
    Next
    objTF.WriteLine("</tr>")
Next
objTF.WriteLine("</table>")
objTF.WriteLine("</center>")

' The calendar is finished displaying. Now display some information that *might* be useful.
iDayofYear = DateDiff("d","01/01/" & year(now),month(now) & "/" & day(now) & "/" & year(now)) + 1
objTF.WriteLine("The Day of the Year is " & iDayofYear)

ProcessOutlook          '* Display any meetings from Outlook

objTF.WriteLine("</BODY>")
objTF.WriteLine("</HTML>")
objTF.close

'*-----*
'* Now display the web page
'*-----*
Set wshShell = WScript.CreateObject ("WScript.shell")
Set IE = CreateObject("InternetExplorer.Application")
IE.visible = 1
IE.navigate("file:" & strWebPagename)

'* We can't delete the file because we get here right after the web page is displayed, and it is
still in use.
'objFSO.DeleteFile(strWebPagename)
WScript.quit

'*-----*
'* Go to Outlook for appointments
'*-----*
Sub ProcessOutlook
Dim objOutlook, objNameSpace, objFolder
'Const olMailItem = 0
'Const olTaskItem = 3
Const olFolderTasks = 13
Const olFolderCalender = 9
```

## VBScript Examples

```
'Create Outlook, Namespace, Folder Objects and Task Item
Set objOutlook = CreateObject("Outlook.application")
Set objNameSpace = objOutlook.GetNameSpace("MAPI")
Set objFolder = objNameSpace.GetDefaultFolder(olFolderCalender)
Set MyItems = objFolder.Items
MyItems.IncludeRecurrences = True
myItems.Sort "[Start]"

' Check for meetings THIS week
dtThisMonday = dateadd("d", 1 - weekday(date), date) + 1
strOutput = strOutput & "<Center><h3> Meetings This Week </h3></center>"
DispOneWeek

' Check for meetings NEXT week
dtThisMonday = dateadd("d", +7, dtThisMonday)
strOutput = strOutput & "<Center><h3> Meetings NEXT Week </h3></center>"
DispOneWeek

' Check for meetings NEXT week
dtThisMonday = dateadd("d", +7, dtThisMonday)
strOutput = strOutput & "<Center><h3> And, the week AFTER that </h3></center>"
DispOneWeek

'Display results to user, if any.
If strOutput > "" Then
    objTF.WriteLine(strOutput)
Else
    objTF.WriteLine("Meetings for this week: NONE<br>")
End If

'Clean up
Set objFolder = Nothing
Set objNameSpace = Nothing
set objOutlook = Nothing
End sub

'*-----*
'* Sub DispOneWeek
'*-----*
' This routine will display one week's worth of Outlook reminders.
Sub DispOneWeek
oCount = 0 : iCount = 0
dtTheSundayAfter = DateAdd("d", +6, dtThisMonday)
For Each CurrAppt in MyItems
    'If CurrAppt.BusyStatus = 2 and CurrAppt.Sensitivity = 0 then '????
    iCount = iCount + 1
    If iCount > 365 then          ' Limit the number of recurring entries to look at
        exit for                ' Bail on THIS item only
    end if
    If CurrAppt.Start >= dtThisMonday And _
        CurrAppt.Start <= dttheSundayAfter Then
        CrOpLine
    End If
Next
End sub

'*-----*
'* Sub CrOpLine - Create an Output line for DispOneWeek
'*-----*
Sub CrOpLine
    oCount = oCount + 1
    strOutput = strOutput & oCount & ". " & _
```

## VBScript Examples

```
"<b>Subject:</b> " & CurrAppt.Subject & _  
" <b>Date/Time:</b> " & CurrAppt.Start & _  
" <b>Duration</b> " & CurrAppt.Duration & "<br>"  
'  
"; recurrence pattern=" & CurrAppt.GetRecurrencePattern & "<br><br>"  
End sub
```

# VBScript Examples

## *Process Current Appointments*

This script will create an e-mail (but it won't send it) listing all appointments from the calendar scheduled for the next week.

Features:

- Interview all appointments in the calendar
- Send an Outlook e-mail

```
Dim objOutlook
Dim objNameSpace
Dim objFolder
Dim MyItems
Dim CurrentAppointment
Dim strOutput

Const olMailItem = 0
Const olTaskItem = 3
Const olFolderTasks = 13
Const olFolderCalender = 9

'Create Outlook, Namespace, Folder Objects and Task Item
Set objOutlook = CreateObject("Outlook.application")
Set objNameSpace = objOutlook.GetNameSpace("MAPI")
Set objFolder = objNameSpace.GetDefaultFolder(olFolderCalender)
Set MyItems = objFolder.Items

dtLastWeek = DateAdd("d", -7, date)
dtNextWeek = DateAdd("d", +7, date)
strOutput = strOutput & "<h2> Meetings This Week </h2>"
icount = 0

For Each CurrentAppointment in MyItems
  If currentAppointment.BusyStatus = 2 and currentAppointment.Sensitivity=0 then
    If CurrentAppointment.Start >= dtLastWeek And CurrentAppointment.Start <= Date+1 Then
      icount = icount + 1
      strOutput = strOutput & icount & ". " & CurrentAppointment.Subject & vbTab & _
        " <b>Time:</b> " & CurrentAppointment.Start & _
        " <b>Duration</b> " & CurrentAppointment.Duration & vbCRLF
      txtNames = txtNames & CurrentAppointment.Subject & vbTab & _
        " <b>Time:</b> " & CurrentAppointment.Start & _
        " <b>duration</b> " & CurrentAppointment.Duration & vbCRLF
    End If
  End If
Next

' Create (but don't send) the e-mail
Set objMsg = objOutlook.CreateItem(olMailItem)
objMsg.To = "e-mail recipient address"
objMsg.Subject = "Subject of message on " & Date()
objMsg.Display
strOutput = replace(strOutput,vbCrLF,"<br>")
objMsg.HTMLBody = strOutput

'Display results to user, if any.
'If strOutput > "" Then
'  MsgBox strOutput, vbExclamation, "Meetings for this week"
'Else
```

## VBScript Examples

```
' MsgBox "No Tasks Today", vbInformation, "No Meetings for this week"  
'End If
```

```
'Clean up  
Set objFolder = Nothing  
Set objNameSpace = Nothing  
set objOutlook = Nothing  
set objMsg = Nothing
```

# VBScript Examples

## *Process Outlook E-mail*

This script will read all of the e-mail messages in the Outlook inbox, and display them. Note that Outlook, because of the way it is configured in some locations, will issue a warning message when a program, including this script, is trying to access its information.

```
' Process Outlook e-mail
Option Explicit
Dim OlApp, Inbox, InboxItems, Mailobject
Dim Msg, emSubject, emFrom, emTo, emBody, emDateSent, eCtr
Const olFolderInbox = 6

Set OlApp = CreateObject("Outlook.Application")
Set Inbox = OlApp.GetNamespace("Mapi").GetDefaultFolder(olFolderInbox)
Set InboxItems = Inbox.Items

Msg = "" : eCtr = 0
For Each Mailobject In InboxItems
  'If Mailobject.UnRead Then
  eCtr = eCtr + 1
  emSubject = Mailobject.Subject
  emfrom = Mailobject.SenderName
  emTo = Mailobject.To
  emBody = Mailobject.Body
  emDateSent = Mailobject.SentOn
  'Mailobject.UnRead = False
  Msg = Msg & "Subject= " & emSubject & ";" & vbCrLf
  Msg = Msg & "    From=" & emFrom & ";" & vbCrLf
  Msg = Msg & "    To= " & emTo & ";" & vbCrLf
  Msg = Msg & "    Sent=" & emDateSent & ";" & vbCrLf
  Msg = Msg & "    Body=" & emBody & ";" & vbCrLf
  Msg = Msg & "-----" & vbCrLf
Next

Msg = "I processed " & eCtr & " e-mail messages!" & vbCrLf & Msg
MsgBox Msg,,WScript.Scriptname

Set OlApp = Nothing
Set Inbox = Nothing
Set InboxItems = Nothing
Set Mailobject = Nothing
```

***VBScripts for Microsoft Access***

# VBScript Examples

## Create an Access Database

This script will create an Access Database. It will then create a table within, populate that table, and read it back (and display it).

It is interesting to note that you *do not* have to have Microsoft Access installed on the system for this script to be able to run!

Features:

- Create an Access database
- Create an Access database table
- Populate an Access database table
- Process an Access database/table

```
Option Explicit
Const Provider = "microsoft.jet.oledb.4.0"
const adInteger = 3                               ' Integer
const adVarChar = 202                             ' Variable Character

Dim db, ds, catalog, tblData, conn, cmd, RS, Msg, sql

db = "\\Dmvmfsp08\mwdxg12$\Data\Scripts\TestDB.mdb" ' <--- path and file name
ds = "provider=" & provider & "; data source=" & db   ' Connection string

set catalog = createobject("adox.catalog")
catalog.create ds

' Create a table in this database
set tblData = createobject("adox.table")
tblData.Name = "tblInfo"
tblData.columns.append "id", adInteger
tblData.columns.append "surname", adVarChar, 30
tblData.columns.append "Address", adVarChar, 30
tblData.keys.append "Info Key", 1, "id"           'unique id
catalog.Tables.Append tblData

' Now populate the database table
set conn = createobject("adodb.connection")
conn.open ds
sql = "insert into tblInfo (id, Surname, Address) values (1,'Grund','1234 W 5th Street')"
conn.Execute sql
sql = "insert into tblInfo (id, Surname, Address) values (2,'Smith','4523 N 6th Street')"
conn.Execute sql
sql = "insert into tblInfo (id, Surname, Address) values (3,'Jones','7890 S 1st Street')"
conn.Execute sql
sql = "insert into tblInfo (id, Surname, Address) values (4,'Blake','6543 E 2nd Street')"
conn.Execute sql
conn.close

' Now read back the data, and display it.
set cmd = createobject("adodb.command")
conn.open ds
cmd.ActiveConnection = conn
sql = "Select id, SurName, Address from tblInfo"
cmd.CommandText = sql
set RS = cmd.execute

' Enumerate each row in the result set
```

## VBScript Examples

```
Msg = ""
while rs.EOF <> true and rs.BOF <> True
    Msg = Msg & "ID=" & rs(0) & "; Surname=" & rs(1) & "; Address=" & rs(2) & vbcrLf
    rs.movenext
wend
conn.close

MsgBox Msg,,"Table contents"

' Free the memory
set tblData = nothing
set catalog = nothing
set conn = nothing

MsgBox "Database created, populated, and read!","",wscript.scriptname
```

***VBScripts for Open Office***

# VBScript Examples

## *Read a Spreadsheet*

```
' Written by Dave Grund, June 9, 2011.

' This program will read an ODS spreadsheet. This one is from Lotus Symphony
' (informational statement, in case there are problems/conflicts later).

Option Explicit
Dim objApp, objFSO, col, row
Dim IpFn, OpFn, ctrOP, OurPath, objOpFile, opLine, strwk
Dim oServiceManager, objDesktop, oDoc, oSheet, oCell, nValue
Dim mArgs(0)      ' Declare an array

OurPath = "C:\Grund\Misc\"
IpFn = OurPath & "Symphony_Spreadsheet_Example.ods"
OpFn = OurPath & "Process ODS File output.txt"

' Start the Service Manager or connect to existing one
Set oServiceManager = WScript.CreateObject("com.sun.star.ServiceManager")

' Create the Desktop - the default frame
Set objDesktop = oServiceManager.CreateInstance("com.sun.star.frame.Desktop")

' Open a Existing ODT

' TODO: Open the spreadsheet WITHOUT displaying the Symphony opening screen!
' The "MakePropertyValue" below does NOT hide the application (Lotus Symphony)
' screen.
'Set mArgs(0) = MakePropertyValue("Hidden", False)
Set mArgs(0) = nothing

Set oDoc = objDesktop.loadComponentFromURL("file:/// " & IpFn, "_blank", 0, mArgs)

strWk = "" : row = 6
For col = 1 to 12
    set oSheet = oDoc.getSheets().getByName("A")
    set oCell = oSheet.getCellByPosition(col-1, row-1)
    ' nValue = oCell.getValue()      ' Get the numeric value
    nValue = oCell.getstring()      ' get the string value

    strwk = strwk & nValue & "/"
Next
msgbox "The values in row " & row & " are " & strwk

' Open the output text file
'Set objFSO = CreateObject("Scripting.FileSystemObject")
'Set objOpFile = objFSO.CreateTextFile(OpFn)
'ctrOP = 0

' Cleanup
oDoc.Close (True)
Set oDoc = Nothing
'objOpFile.close
wscript.quit

' This is a standard function that is required in many cases.
Function MakePropertyValue(cName, uValue)
    Dim oStruct
    Set oStruct = oServiceManager.Bridge_GetStruct("com.sun.star.beans.PropertyValue")
    oStruct.Name = cName
    oStruct.Value = uValue
    Set MakePropertyValue = oStruct
End Function
```

## VBScript Examples

End Function

VBScript Examples

***VBScripts for XML***

# VBScript Examples

## Create an XML File

```
' Create an XML File
Set xmlDoc = CreateObject("Microsoft.XMLDOM")

Set objRoot = xmlDoc.createElement("Level1")
xmlDoc.appendChild objRoot

For i = 1 to 3
  Set objRecord = xmlDoc.createElement("Level2")
  objRoot.appendChild objRecord      '<--- objRecord is Level 2

  Set objName = xmlDoc.createElement("DataElement1")
  objName.Text = "Data Element 1"
  objRecord.appendChild objName

  Set objDate = xmlDoc.createElement("DataElement2")
  objDate.Text = "Data Element 2"
  objRecord.appendChild objDate
Next

Set objIntro = xmlDoc.createProcessingInstruction _
  ("xml","version='1.0'")
xmlDoc.insertBefore _
  objIntro,xmlDoc.childNodes(0)

xmlDoc.Save "xmlexample.xml"
MsgBox "Complete!",,wscript.scriptname
```

### This command creates this:

```
<?xml version="1.0"?>
<Level1>
  <Level2>
    <DataElement1>Data Element 1</DataElement1>
    <DataElement2>Data Element 2</DataElement2>
  </Level2>
  <Level2>
    <DataElement1>Data Element 1</DataElement1>
    <DataElement2>Data Element 2</DataElement2>
  </Level2>
  <Level2>
    <DataElement1>Data Element 1</DataElement1>
    <DataElement2>Data Element 2</DataElement2>
  </Level2>
</Level1>
```

VBScript Examples

***VBScripts for Data***

# VBScript Examples

## Classes

This script demonstrates the use of a Class.

```
Option Explicit
Class Driver
    Private m_DriverName
    Private m_DriverLNo
    Private m_CitationCount

    Private Sub Class_Initialize
        m_DriverName = ""
        m_DriverLNo = ""
        m_CitationCount = 0
    End Sub

    ' DriverName property.
    Public Property Get DriverName
        DriverName = m_DriverName
    End Property
    Public Property Let DriverName(p_drivername)
        m_DriverName = p_drivername
    End Property

    ' DriverLicense Number property.
    Public Property Get DriverLNo
        DriverLNo = m_DriverLNo
    End Property
    Public Property Let DriverLNo(p_driverLNo)
        m_DriverLNo = p_driverLNo
    End Property

    ' CitationCount property (read only).
    Public Property Get CitationCount
        CitationCount = m_CitationCount
    End Property

    ' Methods.
    Public Sub IncreaseCitations(p_valuetoincrease)
        m_CitationCount = m_CitationCount + p_valuetoincrease
    End Sub

    Public Sub ClearCitationCount
        m_CitationCount = 0
    End Sub
End Class

'*-----*
'* Declare our variables and constants
'*-----*
Const strFilename = "WriteRec.XML"
Dim c, Msg, intRecNum

Set c = New Driver
intRecNum = 0

c.DriverName = "Arnie Schwartz"
c.DriverLNo = "CA123456"
c.IncreaseCitations(5)
c.IncreaseCitations(3)
```

## VBScript Examples

```
DispOneRec
```

```
c.DriverName = "Maria Schriver"  
c.DriverLNo = "CA123457"  
c.IncreaseCitations(2)  
DispOneRec
```

```
MsgBox "Complete! I wrote " & intRecNum & " records.",,WScript.scriptname
```

```
Sub DispOneRec
```

```
    intRecNum = intRecNum + 1  
    Msg = "Driver name=" & c.Drivername & ";" & vbcrLf  
    Msg = Msg & "Driver License No.=" & c.DriverLNo & ";" & vbcrLf  
    Msg = Msg & "No. of citations=" & c.CitationCount & vbcrLf  
    MsgBox Msg,,WScript.Scriptname  
    c.ClearCitationCount  
End sub
```

# VBScript Examples

## *Duration*

This script asks the user for a start time and an end time, and will display the difference.

Features:

- Date and time difference calculation

```
' Duration - Calculate how long something has taken
Option Explicit
Dim tmStart, tmEnd, tmDur
tmStart = trim(InputBox("What is the start time?","Start Time",FormatDateTime(now(),4)))
tmEnd = trim(InputBox("What is the end time?","End Time"))
If (TmStart = null) or (TmStart = "") or (TmEnd = null) or (TmEnd = "") then
    MsgBox "Please enter a valid non-blank time",,"Error!"
    wscript.quit
end if
tmDur = DateDiff("n",tmStart,tmEnd)
If tmDur < 0 then
    MsgBox "The start time should be earlier than the end time." & vbcrLf & "Please try
again.",,"Error!"
    wscript.quit
end if
MsgBox "That task took " & tmDur & " minutes.",,wscript.scriptname
```

# VBScript Examples

## *Create and Process a Record Set*

```
' This program demonstrates how to use a Record Set without a database.

Option Explicit
Dim objExcel, objWorkbook, objWorksheet, row, col, msg, ThisTxt, ThisLen, ctrCells,
objFSO
Dim ipFN, j, ctrArrayElems
Dim GroupArray()

If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a spreadsheet file name to this one!"
    wscript.quit
end if

'*-----*
'* Create a recordset to contain the fields from the STIG
'*-----*
const adUseClient = 3
const adInteger = 3
const adChar = 129
const adDBTimeStamp = 135

dim MyRec : set MyRec = createobject("adodb.recordset")
MyRec.cursorLocation = adUseClient
MyRec.Fields.append "Col18",adchar,10
MyRec.Fields.append "Col19",adchar,100
MyRec.Fields.append "Col21",adchar,100
MyRec.Fields.append "Col22",adchar,200
MyRec.Fields.append "Col23",adchar,100
MyRec.Fields.append "Col28",adchar,255
MyRec.Fields.append "Col30",adchar,255
MyRec.Open

'*-----*
'* Read the Excel spreadsheet, and create a recordset of the contents
'*-----*
Set objFSO = Createobject("Scripting.FileSystemobject")
IpFN = trim(wscript.arguments(0))

' Start the Excel application
Set objExcel = Createobject("Excel.Application")
objExcel.Application.visible = false ' Do not make Excel visible
Set objWorkbook = objExcel.Workbooks.Open(IpFN)
Set objWorksheet = objWorkbook.Worksheets(1)

ctrArrayElems = 0
Dim GroupID, GroupName
For Row = 1 to objWorksheet.UsedRange.Rows.Count
    GroupID = objWorksheet.Cells(row,18).Value
    GroupName = objWorksheet.Cells(row,19).Value
    If (trim(GroupID) <> "") or (trim(GroupName) <> "") then
        ctrArrayElems = ctrArrayElems + 1
        MyRec.Addnew
        MyRec("Col18") = objWorksheet.Cells(row,18).Value
        MyRec("Col19") = objWorksheet.Cells(row,19).Value
        MyRec("Col21") = objWorksheet.Cells(row,21).Value
```

## VBScript Examples

```
MyRec("Col22") = objWorksheet.Cells(row,22).Value
MyRec("Col23") = objWorksheet.Cells(row,23).Value
MyRec("Col28") = objWorksheet.Cells(row,28).Value
MyRec("Col30") = objWorksheet.Cells(row,30).Value
end if
next

'msg = "I found " & Row & " rows in the spreadsheet and "
'msg = msg & ctrArrayElems & " GroupIDs"
'MsgBox(Msg)

objExcel.Application.quit
Set objWorksheet = nothing
Set objWorkbook = nothing
Set objExcel = nothing

'*-----*
'* Count the records in the recordset
'*-----*
'MyRec.movefirst
'i = 0
'while not MyRec.eof
' i = i + 1
' MyRec.Movenext
'wend
'MsgBox ("Before deduping, I saw " & i & " records")

'*-----*
'* Now remove duplicates from the recordset
'*-----*
Dim Lastrec, ThisRec
LastRec = ""
myrec.sort = "Col19,Col18"
MyRec.movefirst
while not MyRec.eof
  ThisRec = trim(MyRec("Col19")) & trim(MyRec("Col18"))
  If LastRec = ThisRec then
    myrec.delete
  else
    LastRec = ThisRec
  end if
  MyRec.Movenext
wend

'*-----*
'* Count the records in the recordset
'*-----*
MyRec.movefirst
Dim Numrows : numRows = 0
while not MyRec.eof
  numRows = numRows + 1
  MyRec.Movenext
wend
'MsgBox ("After deduping, I saw " & numRows & " records")

'*-----*
'* Now create a Word table with the fields
```

## VBScript Examples

```
'*-----*
Dim objWord, objDoc, objRange, objTable
Const NUMBER_OF_COLUMNS = 7
Set objWord = CreateObject("Word.Application")
objWord.Visible = True
Set objDoc = objWord.Documents.Add() ' Create a NEW document
Set objRange = objDoc.Range()
objDoc.Tables.Add objRange, numRows, NUMBER_OF_COLUMNS
Set objTable = objDoc.Tables(1) ' Work with the first (and only) table
objTable.Cell(1, 1).Range.Text = "STIG ID"
objTable.Cell(1, 2).Range.Text = "VULID"
objTable.Cell(1, 3).Range.Text = "ID10"
objTable.Cell(1, 4).Range.Text = "Severity"
objTable.Cell(1, 5).Range.Text = "Weight"
objTable.Cell(1, 6).Range.Text = "FixRef"
objTable.Cell(1, 7).Range.Text = "System"

objTable.Cell(2, 1).Range.Text = "Col S"
objTable.Cell(2, 2).Range.Text = "Col R"
objTable.Cell(2, 3).Range.Text = "Col U"
objTable.Cell(2, 4).Range.Text = "Col V"
objTable.Cell(2, 5).Range.Text = "Col W"
objTable.Cell(2, 6).Range.Text = "Col AB"
objTable.Cell(2, 7).Range.Text = "Col AD"

MyRec.movefirst
Row = 3
while not MyRec.eof
    objTable.Cell(row, 1).Range.Text = MyRec("Col19")
    objTable.Cell(row, 2).Range.Text = MyRec("Col18")
    objTable.Cell(row, 3).Range.Text = MyRec("Col21")
    objTable.Cell(row, 4).Range.Text = MyRec("Col22")
    objTable.Cell(row, 5).Range.Text = MyRec("Col23")
    objTable.Cell(row, 6).Range.Text = MyRec("Col28")
    objTable.Cell(row, 7).Range.Text = MyRec("Col30")
    MyRec.Movenext : row = row + 1
wend

objTable.AutoFormat(16)
MsgBox("Your Word table is ready!")

Wscript.quit
```

# VBScript Examples

## *Modify a File*

This script will copy a file, and skip HTML metadata. It can be modified to do any kind of file editing and manipulation that you want.

### Features:

- Process a command line parameter. If you drag an icon over the VBScript file icon, the former will become a command line parameter.
- Look for a string within another string.

```
Option Explicit
Dim objFSO, objIPFile, objOPFile
Dim FN, ipPath, opPath
Dim CtrIP, CtrOP, ipLine, OPLine, ctrSkipped
Dim Msg, i

If Wscript.Arguments.Count = 0 then
    wscript.echo "Please drag a file on top of the icon!"
    wscript.quit
end if

ipPath = wscript.arguments(0)                ' Complete input file name
OpPath = left(ipPath,len(ipPath)-5) & ".txt"

Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objIpFile = objFSO.OpenTextFile(ipPath)
Set objOPFile = objFSO.CreateTextFile(OpPath)

CtrIP = 0 : CtrOP = 0

Do Until objIPFile.AtEndOfStream
    ipLine = RTrim(objIPFile.ReadLine) : CtrIP = CtrIP + 1
    If (left(ipLine,1) = "<") or _
        (instr(1,ipLine,"<html>",vbTextCompare) > 0) or _
        (instr(1,ipLine,"<tt>",vbTextCompare) > 0) or _
        (instr(1,ipLine,"<td w",vbTextCompare) > 0) then
        ctrSkipped = ctrSkipped + 1
    else
        objOPFile.WriteLine(ipLine) : CtrOP = CtrOP + 1
    end if
loop

objIPFile.Close()
objOPFile.Close()

Msg = ""
Msg = Msg & "IpFile: " & ipPath & vbCRLF
Msg = Msg & "OpFile: " & OpPath & vbCRLF
Msg = Msg & CtrIP & " records read. " & vbCRLF
Msg = Msg & CtrSkipped & " records skipped." & vbCrLF
Msg = Msg & CtrOP & " records written." & vbCrLF
MsgBox Msg,,Wscript.ScriptName

Set objFSO = nothing
Set objIPFile = nothing
Set objOPFile = nothing
```

## ***VB Scripts for Lotus Notes***

### *SendMail*

```
' Sendmail.vbs
Option Explicit
Dim nSession, db, doc

Set nSession = CreateObject("Notes.NotesSession")
'Gets the current user's maildatabase
Set db = nSession.GETDATABASE("", "")
Call db.OPENMAIL
Set doc = db.CREATEDOCUMENT
Call doc.REPLACEITEMVALUE("SendTo", "dgrund@nospam.com")
Call doc.REPLACEITEMVALUE("Subject", "Mail sent from VBscript
SendNotesmail Using Lotus Notes")
Call doc.REPLACEITEMVALUE("Body", "Does it work?")
Call doc.SEND(False)
MsgBox ("The command completed.")
```

## VBScript Examples

### ***Snippets***

#### *Change the Format of a Column*

This snippet will set the width of a column, and change the format (of an entire column) so it wraps the text.

```
objExcel.Columns(2).ColumnWidth = 60
objWorksheet0.Cells(oRow,2).Value = (text field)
set objRange = objExcel.Range("B1").EntireColumn
objRange.Wraptext = true
```

#### *Include Common Code*

' The common VBScript statements (example)

```
' Globals.txt - Global VBScript statements that are used by all of the
'scripts.
```

```
Const pathname = "C:\My Documents\DataFile\"
```

' This code will include a common set of VBScript statements.

```
Const ForReading = 1
Dim fso: set fso = CreateObject("Scripting.FileSystemObject")
Dim f : set f = fso.OpenTextFile("globals.txt",ForReading)
Dim s: s = f.ReadAll()
ExecuteGlobal s
```

```
Dim filename : filename = pathname & "myfile.xls"
```

```
` Prove this is working
wscript.echo "The file name is " & filename
```

## VBScript Examples

### **Notes**

#### *Converting VBA to VBS*

Many times, you can find exactly what you want by searching the web. Sometimes, however, the exact combination of features that you want to use aren't readily available in an example form.

I have found that ONE way to accomplish what you want to accomplish in VBScript is to write a macro, and then convert that macro from VBA to VBS. Usually, this entails only syntactical changes.

For example, the following VBA code:

```
Selection.Fields.Add Range:=Selection.Range, Type:=wdFieldPage  
Selection.Fields.Add Range:=Selection.Range, Type:=wdFieldNumPages
```

Is converted to the following VBScript code:

```
With objWord  
    .Selection.Fields.Add .Selection.Range,wdFieldPage  
    .Selection.Fields.Add .Selection.Range,wdFieldNumPages  
end with
```