Category:	Function/Proc. name:	Purpose:

'==Entwickler-Werkzeuge:==

Class	DevelopmentTools	Richtet z.B. Ausgabemöglichkeit für Tests etc. ein (KEIN VISTA)
Sub	.output(value)	Beliebige Ausgabe in neuem Textfenster (Alternative zu WScript.Echo)
		>> Zeile wird von nachfolgenden Ausgaben immer wieder überschrieben
Sub	.listOutput(value)	Zeilenweise Ausgabe (statt auf derselben Stelle wie output())
Sub	.protocol(text)	Output als Textdatei auf dem Desktop
Sub	.errorDialog(text)	Für Fehlermeldungen, vom Entwickler für Entwickler
Sub	errorDialog(text)	Für Fehlermeldungen, vom Entwickler für Benutzer

'==Dialoge:==

Class	Dialog	Erzeugt Dialog mit frei wählbaren Buttons (noch KEIN VISTA)
		>> Bsp.: meineKonsole1 = new ieDialog
Sub	.show()	Macht mit "new Dialog" erzeugten Dialog sichtbar
Sub	.hide()	Versteckt den Dialog
Sub	.showCaption(TrueOrFalse)	Blendet Titel-Fensterbalken ein/aus
Sub	.setIcon(filename)	Weist dem Fensterbalken/Taskbarbutton bel. Piktogramm zu
Sub	.titleBarButtons(value)	Erzeugt/Entfernt Max.- & Min.- & Schließknöpfe oben rechts
Sub	.setTitle(title)	Bestimmt Fenster-/Taskbartitel
Sub	.scrollbars(value)	Erzeugt/Entfernt Scrollbalken
Sub	.border(borderStrength, borderStyle, innerBorder)	Erzeugt/Entfernt Fensterrahmen und bestimmt Rahmenstil. Erwartete Werte: >> border: dialog, none, thick, thin >> borderStyle: complex, normal, raised, static,sunken; innerBorder: yes, no
Sub	.moveTo(x,y)	Verändert Fensterposition
Sub	.resizeTo(width,height)	Verändert Fenstergröße
Sub	.addButton(name, aufschrift, x, y)	Fügt Button hinzu, mit Positionsangabe
Sub	.positionElement(name, x, y)	Positioniert im Nachhinein bel. Elemente (z.B. Buttons)
Sub	.removeElement(name)	Entfernt bel. Elemente
Function	.eventBy(elementName)	Prüft, ob an einem Element ein Mausklick o.ä. geschehen ist
Function	.javaScriptCommand(commands)	Lässt den Dialog beliebigen Javascript-code ausführen
Sub	.Quit	Beendet den Dialog
Class	ieDialog	Erzeugt Dialog mit frei wählbaren Buttons
		>> Basis: Internet-Explorer
		>> Bsp.: meineKonsole1 = new ieDialog
Sub	.display	Macht den mit new ieDialog erzeugten Dialog sichtbar
		>> Bsp.: meineKonsole1.display
Sub	.hide()	Macht den mit new ieDialog erzeugten Dialog unsichtbar
Sub	.top(wert)	Position vom oberen Rand aus
Sub	.left(wert)	Position vom linken Rand aus
Sub	.height(wert)	Fensterhöhe
Sub	.width(wert)	Fensterbreite
Sub	.addButton(aufschrift, name)	Fügt Button hinzu
		>> Bsp.: meineKonsole1.addButton("Bitte klicken", "meinKnopf")
Function	.buttonKlickedIs(name)	Prüft, ob ein bestimmter Button geklickt wurde
		>> Jeder Button wird mit zuvor selbstgewähltem Namen angesprochen
		>> Gibt Wahrheitswert zurück
		>> Sollte in eine Schleife eingebaut werden
Sub	.setBgColor(wert)	Setzt Hintergrundfarbe
		>> Bsp.: meineKonsole1.setBgColor("#FF8897")
Sub	.setBgPicture(file)	Setzt HG-Bild
Sub	.addStyle(style)	Fügt CSS-Stil hinzu
Sub	.progressbar_set(x,y,barwidth)	Setzt Fortschrittsanzeige (noch leer)
Sub	.progressbar_fill(slowness)	Füllt Fortschrittsanzeige in 'slowness' Sekunden
Sub	.progressbar_stop	Pausiert Fortschrittsanzeige
Function	.callJavaScriptFunction(fctName)	Ruft beliebige JavaScript-Funktion aus "resources\dialogJavascript.js" auf
		>> Bsp.: meineKonsole1.callJavaScriptFunction("funktionsname()")
		>> Empfängt ggf. auch Rückgabewerte von JavaScript

'==Betriebssystem:==

Function	dosCommand(command)	Führt DOS-Befehl aus und gibt Output zurück
		>> langsame Rückgabe bei großen Outputs, dafür ohne Schwarzfenster
Function	dosCommandFast(command)	Führt DOS-Befehl aus und gibt Output zurück
		>> schnelle Rückgabe, aber Schwarzfenster erscheint manchmal
Function	dosCommandLight(command)	Führt DOS-Befehl ohne Output-Rückgabe aus
		>> Schnelle Ausführung
		>> Anderer Name für independentRun()
Sub	restartSystem	System-Neustart
Function	RegValue(zeichenkette)	Gibt Wert eines Registry-Pfads zurück
		>> Gibt String "Error" zurück, falls Schlüssel/Name nicht existent
		>> Gibt bei Binärdaten Array zurück
Sub	regWriteValue(keyPath, valueName, value, dataType)	Erzeugt einen neuen Registry-Pfad mit Wert & Wertname oder ergänzt/überschreibt einen existierenden >> dataType-Möglichkeiten: "REG_SZ", "REG_BINARY", "REG_MULTI_SZ", "REG_DWORD_BIG_ENDIAN" >> ... "REG_DWORD", "REG_DWORD_LITTLE_ENDIAN", "REG_NONE", "REG_EXPAND_SZ"

```
Sub       regDeleteKey(keyPath)                 Löscht einen Registry-Schlüssel

Sub       regDeleteValueName(keyPath)           Löscht einen Wertnamen und dazgh. Wert aus einem Registry-Schlüssel
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==Prozessmanagement:==
Class     OwnProcess                            Richtet Zugriffsmöglichkeit für selbstgestartete Prozesse ein

Sub             .start(procName, procParameters)    Startet Prozess, der später mit einfacher Indexangabe gelöscht werden kann
                                                    >> - Vorher OwnProcess-Instanz erzeugen
                                                    >> - Umgebungsvariablen ggf. umwandeln statt mit %-Zeichen im String belassen
                                                    >> - Falls keine Parameter, dann Leerstring angeben
                                                    >> - s. OwnProcess.kill()
Sub             .kill()                             Beendet mit OwnProcess.start() gestartete Programme

Sub       independentRun(command)                   Programme und Skripte unterschiedslos starten, wie mit Doppelklick in Windows (evtl. kein Vista)
                                                    >> - Gut zum Runnen von VBS-Skripten ohne "WScript"-Befehl,
                                                    >>   da dieser Argumentübergaben an das Skript nicht akzeptiert.
                                                    >> - Gut zum BAT-Runnen ohne Kommandozeilen-Fenster

Sub       kill(processName)                         Beendet jeden Prozess mit dem übergebenen Namen

Sub       killPrecisely(usedCommandLine)            Beendet Prozess gezielter als kill(), nämlich je nach seiner Kommandozeile
                                                    >> (ab WinME)

Sub       killperID(procID)                         Beendet den Prozess mit der zu übergebenden Prozess-ID

Function  processIDExists(processID)                Prüft, ob ein Prozess mit der ID gerade läuft
                                                    >> Gibt True (-1) oder False (0) zurück

Function  commandLineExists(commandline)            Prüft, ob ein Prozess mit der CommandLine gerade läuft
                                                    >> Gibt True (-1) oder False (0) zurück

Sub       killOwnDuplicates                         Beendet laufende Duplikate des Skripts
                                                    (wenn Sie vor diesem Skript gestartet worden waren)

Function  CurrentLoadPercentage                     Gibt momentane CPU-Auslastung zurück (KEIN VISTA)

Function  AverageLoadPercentage(periodInSeconds)    Gibt durchschnittliche CPU-Auslastung innerhalb (KEIN VISTA)
                                                    des anzugebenden Zeitraums zurück

Sub       TypeControlled(keyCombi, processName,     Tippt und wartet, bis Zielprogramm anzugebende Speicherauslastungs-
          commandline, memConsumDiff)               Erhöhung erreicht hat, erst ab WinME sicher nutzbar (kein Win98)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==Verzeichnis- & Dateioperationen:==
Function  ScriptPath                                Gibt Pfad des aktuellen Skripts zurück (mit Querstrich am Ende)

Function  pathPartsInQuotes(path)                   Für C:\"abc"\"def" statt C:\abc\def
                                                    >> - Gibt String zurück
                                                    >> - nur für Verzeichnisspfade ohne Dateinamenerwähnung!
                                                    >> - schließt immer mit Backslash ab
                                                    >> - Wichtig für "cmd /c"-Runnings, dosCommand() etc.

Function  cdDrive                                   Gibt Laufwerksbuchstaben des CD-Laufwerks zurück
                                                    (sinnvoll nur wenn es das einzige CD-Laufwerk ist)

Sub       changeFileVisibility(filename)            Ändert Sichtbarkeit einer Datei

Sub       copyContent(path1, path2)                 Kopiert *Inhalt* eines Verzeichnisses in anderes Verzeichnis
                                                    >> Backslash am Schluss des Parameters erforderlich
                                                    >> nur Dateien werden kopiert - bitte für Ordner erweitern!

Sub       moveContent(path1, path2)                 Bewegt *Inhalt* eines Verzeichnisses in anderes Verzeichnis
                                                    >> Backslash am Schluss des Parameters erforderlich
                                                    >> nur Dateien werden bewegt - bitte für Ordner erweitern!

Sub       Rename(file, newFilename)                 Umbenennen einer Datei
                                                    >> Parameter file:        Dateiname mit Pfad
                                                    >> Parameter newFilename: Dateiname OHNE Pfad
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==I/O-Operationen:==
Sub       makeFile(content, filename)               Gut geeignet für Signaldateien
                                                    >> Zeilenumbrüche mit "\r\n"!
                                                    >> Gleichnamige Datei wird überschrieben

Sub       saveText(content, filename)               Speichert Text als Textdatei oder hängt an vorhandene an
                                                    >> Zeilenumbrüche mit "\r\n"!

Function  textfileToArray(filename)                 Liest Textdatei in Array
                                                    >> Anwendung: meinArray = textfileToArray("xyz.txt")
                                                    >> gibt "error" zurück, falls Datei nicht existent

Function  readBetweenMarks(mark1, mark2, file)      Liest durch Markierungen gekennzeichneten Teil einer Textdatei
                                                    >> Auch für Binärdaten innerhalb von Textdateien geeignet
```

```
Function   WriteBinary(filename, data)          Schreibt Binärdatei
                                                >> Anwendungsbsp.:
                                                >> WriteBinary(filename, array(&h0A, &hB1, &h39...))
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==Strings:==
Function   occurrencesIn(String, element)      Gibt zurück, wie oft eine Zeichenfolge in einem String vorkommt

Function   replaceAll(sentence, toBeReplaced, newElement)
                                                Ersetzt alle [toBeReplaced] im String [sentence] durch [newElement]


Function   stringsEqual(kette1, kette2)            Gibt Wahrheitswert zurück, ob zwei Strings gleich sind
                                                >> Gr.-Kl.-Schreibung egal
                                                >> Außerdem: Gibt z.B. 123 und "123" als gleich an

Function   stringsExactlyEqual(kette1, kette2)     Gibt Wahrheitswert zurück, ob zwei Strings gleich sind
                                                >> Gr.-Kl.-Schreibung NICHT egal
                                                >> Außerdem: Gibt z.B. 123 und "123" als gleich an
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==Interaktion (dialogunabhängig):==
Function   getMouseX                            ermittle X-Position des Mauszeigers (KEIN VISTA)

Function   getMouseY                            ermittle Y-Position des Mauszeigers (KEIN VISTA)

Function   newHotkey(hotkeysString)             Richtet Hotkey ein (KEIN VISTA)
                                                >> Formatbeispiel: var1 = newHotkey("CTRL+SHIFT+A")
                                                >> Keine Leerzeichen ins Stringargument!

Function   hotkeyPressed(hotkey)                   Prüft, ob mit newHotkey eingerichteter Hotkey gedrückt wurde (KEIN VISTA)
                                                >> - Sollte in Schleife eingebaut werden
                                                >> - Parameter ist die zuvor mit newHotkey() definierte Variable
                                                >> - Gibt Wahrheitswert zurück

Sub        removeHotkeys                        Macht die Hotkey-Setzung rückgängig
                                                >> Sollte immer noch vor Programmende genutzt werden!

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==Internet & Netzwerke:==
Function   isOnline                             Gibt Wahrheitswert darüber zurück, ob der Computer online ist

Function   currentIPs                           Gibt Array über aktuelle IP-Adressen des PCs zurück
                                                >> Anwendung: meinArray = currentIPs

Function   currentIP                            Falls online: Gibt aktuelle Internet-IP des PCs zurück
                                                >> Falls offline: Gibt IP des PCs im lokalen Netzwerk zurück

Function   urlEncode(zKette)                    Wandelt URLs mit Sonderzeichen für Parameterübergangen an PHP-Dateien usw. um

Function   SocketServer                         Richtet Empfänger für Nachrichten ein
                                                >> Gibt SocketServerObject zurück
                                                >> Anwendung: Set myServer1 = SocketServer
Class      SocketServerObject                   Richtet Empfänger für Nachrichten ein (TCP)
                                                >> Wird nur über die Function SocketServer instantiiert
               .gotNewData                      >> Gibt Wahrheitswert darüber zurück, ob neue Daten eingetroffen sind
                                                   (sollte in Schleife eingebaut werden)
               .data                            >> Gibt eingetroffene Daten zurück
               .answer(data)                    >> sendet Daten an verbundenen SocketClient
               .port                            >> Gibt derzeit benutzten Port zurück (wurde zuvor zufällig ausgewählt)

Function   SocketClient(serverIP, port)            Richtet Sender für Nachrichten an einen Remote-PC ein (TCP)
                                                >> Gibt SocketClientObject zurück
                                                >> Anwendung: Set myClient1 = SocketClient(serverIP, port)
Class      SocketClientObject                   Richtet Sender für Nachrichten an einen Remote-PC ein
                                                >> Wird nur über die Function SocketClient instantiiert
               .isConnected                     >> Gibt Wahrheitswert darüber zurück, ob Verbindung besteht
                                                   (sollte in Schleife eingebaut werden)
               .sendData(data)                  >> sendet Daten
               .gotNewAnswer                    >> Gibt Wahrheitswert darüber zurück, ob neue Antwort eingetroffen ist
                                                   (sollte in Schleife eingebaut werden)
               .data                            >> Gibt eingetroffene Antwort zurück
               .serverIP                        >> Gibt derzeit angesprochene IP zurück
               .port                            >> Gibt derzeit angesprochenen Port zurück

Class      Uploader                             Ermöglicht das Hochladen von Daten ohne FTP (evtl. kein Vista)
                                                >> Datei "Uploader.php" im Zielverzeichnis erforderlich!
Sub            .connectWith(url)                Legt Ziel-URL fest, auf die zukünftig immer wieder hochgeladen werden kann
Sub            .upload(filename)                   Lädt beliebige Datei auf zuvor festgelegte Ziel-URL hoch
Sub            .disconnect                         Trennt die Verbindung
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==Konstanten:==
Const      HOMEDRIVE                               Pfad des aktuellen Systemverzeichnisses (oft "C:\")

Const      desktopPath                          Pfad des Desktop-Verzeichnisses
```

```
Const       TEMP                                Pfad des temporären Verzeichnisses
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'==Sonstiges:==
Function    callJavaScriptFunction(functionName) Ruft dialogunabhängig eine der JavaScript-Funktionen in "javascriptInterface.js" auf (KEIN VISTA)
                                                 >> Empfängt ggf. auch Rückgabewerte von JavaScript

Sub         permitActiveContents                 Erlaubt in XP-SP2 Nutzung des IE ohne Sicherheitswarnungen
                                                 >> IE-Datei auf CD: keine Probleme
                                                 >> IE-Datei auf FP: funktioniert nicht im IE-aufrufenden VBS-Code
                                                 >>  (Extra-Code ist manuell zu starten + Sleep im aufrufenden Code)
Sub         forbidActiveContents                 Macht permitActiveContents rückgängig, sonst bleiben die Einstellungen
                                                 evtl. auch nach dem nä. Neustart

Function    screenSize(heightOrWidth)            Gibt je nach Auflösung aktuelle Bildschirmmaße zurück (KEIN VISTA)
                                                 >> Parameter: "height", "width"
                                                 Erst ab WinME sicher nutzbar (kein Win98)

Class       Screenshot                               Bereitet Screenshot vor (KEIN VISTA)
Sub             .toFile(filename)                    Speichert kompletten Bildschirm in anzugebende Datei
                                                 >> - Gewünschtes Bildformat wird automatisch an Dateiendung erkannt
                                                 >> - JPG-Qualität vorher einstellbar via Screenshot.jpgQuality (0-100)
Sub             .areaToFile(filename)            Speichert per Maus wählbaren Bildschirmausschnitt in anzugebende Datei
                                                 >> - Gewünschtes Bildformat wird automatisch an Dateiendung erkannt
                                                 >> - JPG-Qualität vorher einstellbar via Screenshot.jpgQuality (0-100)
Sub             .areaToClipboard                     Speichert per Maus wählbaren Bildschirmausschnitt ins Clipboard
                                                 >> Es wird dennoch in "C:\" eine BMP-Datei generiert (wird nicht gelöscht)

Function    elementExistsInArray(element, array) Gibt Wahrheitswert darüber zurück, ob ein Element in einem Array vorkommt


Function    pickFromCollection(collection, index) Wählt aus einer Collection ein Objekt per Index aus
                                                  >> collection-Parameter muss eine echte Collection sein

Class       Clock                                Startet Stopuhr (Set obj = new Clock)
Function        .stopResult                          Liest Stopuhr aus (erg = obj.stopResult)

Function    randomNumber(max)                    Glatte Zufallszahl zwischen 0 und <max>

Function    skypeInstalled                       Gibt Wahrheitswert darüber zuurück, ob Skype installiert ist

Sub         endAll                               Beendet Skript und beseitigt zuvor alle vbsplus-Objekte aus dem Speicher

Sub         vbsplusTest                          Prüft Aufrufbarkeit der Funktionsbibliothek




Option Explicit '#################################################################
Dim WshShell :          Set WshShell = WScript.CreateObject("WScript.Shell")
Dim javascriptInterface : Set javascriptInterface = WScript.CreateObject("InternetExplorer.Application")
Dim shellApp :          Set shellApp = WScript.CreateObject("Shell.Application")
Dim datSystem :             Set datSystem = WScript.CreateObject("Scripting.FileSystemObject")
Dim DriveList :         Set DriveList = datSystem.Drives
Dim objWMIService :     Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
Dim HOMEDRIVE :         HOMEDRIVE = WshShell.ExpandEnvironmentStrings("%HOMEDRIVE%")
Dim desktopPath :       desktopPath = WshShell.ExpandEnvironmentStrings("%HOMEDRIVE%%HOMEPATH%\Desktop\")
Dim TEMP :                  TEMP = WshShell.ExpandEnvironmentStrings("%TEMP%\")
'#############################################################################
'== Erweitere VBS um JavaScript-Möglichkeiten: ==
    permitActiveContents
    javascriptInterface.navigate(ScriptPath & "resources\javascriptInterfaceEmbedder.htm")
    javascriptInterface.visible = 0
    Do While (javascriptInterface.Busy)
        Loop
'#############################################################################



Sub vbsplusTest
    msgBox "Gottlob - das Modul ""vbsFunctions"" ist funktionsfähig eingebunden."
    End Sub


Function callJavaScriptFunction(functionName)
    javascriptInterface.Document.getElementById("caller").value = functionName
    WScript.sleep 33
    callJavaScriptFunction = javascriptInterface.Document.getElementById("returner").value
    End Function


Function getMouseX
    javascriptInterface.Refresh
```

```
	getMouseX = javascriptInterface.Document.getElementById("mouseX").value
	End Function

Function getMouseY
	javascriptInterface.Refresh
	getMouseY = javascriptInterface.Document.getElementById("mouseY").value
	End Function




Class Clock
	'||||||||||||||||||||||||||||||||||||||||||||||||||||||
	Public clocktime1
	'||||||||||||||||||||||||||||||||||||||||||||||||||||||
	Public Function stopResult
		Dim clocktime2
		clocktime2 = (Hour(Now)*3600) +  (Minute(Now) * 60) + second(Now)
		stopResult = clocktime2 - clocktime1
		End Function
	'||||||||||||||||||||||||||||||||||||||||||||||||||||||
	Public Sub Class_Initialize()
		clocktime1 = (Hour(Now)*3600) +  (Minute(Now) * 60) + second(Now)
		End Sub
	'||||||||||||||||||||||||||||||||||||||||||||||||||||||
	End Class

Dim intern_clocktime1, intern_clocktime2
'-------------------------
Sub intern_clock_start
	intern_clocktime1 = (Hour(Now)*3600) +  (Minute(Now) * 60) + second(Now)
	End Sub
'-------------------------
Function intern_clock_stop
	intern_clocktime2 = (Hour(Now)*3600) +  (Minute(Now) * 60) + second(Now)
	intern_clock_stop = intern_clocktime2 - intern_clocktime1
	End Function



Sub permitActiveContents
	On Error Resume Next
	wshShell.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE_LOCALMACHINE_LOCKDOWN\iexplore.exe",0,"REG_DWORD"
	wshShell.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE_LOCALMACHINE_LOCKDOWN\Settings\LOCALMACHINE_CD_UNLOCK",1,
"REG_DWORD"
	End Sub


Sub forbidActiveContents
	On Error Resume Next
	wshShell.RegDelete "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE_LOCALMACHINE_LOCKDOWN\iexplore.exe"
	wshShell.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE_LOCALMACHINE_LOCKDOWN\Settings\LOCALMACHINE_CD_UNLOCK",0,
"REG_DWORD"
	End Sub


Function pickFromCollection(collection, index)
	Dim element, a
	For each element in collection
		a = a + 1
		If a = index Then
			Set pickFromCollection = element
			End If
		Next
	End Function


Sub copyContent(folderPath1, folderPath2)
	Dim element, filesList : Set filesList = datSystem.GetFolder(folderPath1).Files
```

```
    For Each element in filesList
        datSystem.CopyFile element.path, folderPath2, True
        Next
    End Sub

Sub moveContent(folderPath1, folderPath2)
    Dim element, filesList : Set filesList = datSystem.GetFolder(folderPath1).Files
    For Each element in filesList
        datSystem.MoveFile element.path, folderPath2
        Next
    End Sub


Sub Rename(file, newFilename) 'Var. file mit Pfad, Var. newFilename ohne Pfad
    Dim fileObj, uhr
    ' ==Gew. Dateiname darf nicht schon vergeben sein:==
    If datSystem.FileExists(datSystem.GetParentFolderName(file) & "\" & newFilename) Then
        errorDialog "Filename already exists."
        End If
    ' ==Umbenenn-Versuch:==
    If datSystem.FileExists(file) Then                  ' Wenn umzubenennende Datei existiert,
        Set fileObj = datSystem.GetFile(file)
        On Error Resume Next
        fileObj.Name = newFilename                      ' dann benenne sie um.
        Set uhr = new Clock
        Do While Err                                ' Falls es nicht klappt,
            On Error Resume Next
            fileObj.Name = newFilename                  ' versuche es dennoch solange
            WScript.sleep 11
            if uhr.stopResult > 5 Then                  ' bis 5 Sek. überschritten sind.
                errorDialog "Could not rename """ & file & """ to """ & newFilename & """."
                Exit Do
                End If
            Loop
        End If
    End Sub


'/¯¯¯¯¯¯¯¯¯¯¯¯¯¯\/¯¯¯¯¯¯¯¯¯¯¯¯¯¯\/¯¯¯¯¯¯¯¯¯¯¯¯¯¯\/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
'      ==D E V E L O P M E N T - T O O L S - F E A T U R E==

    Class DevelopmentTools
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public IEOutput
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Sub errorDialog(text)
        Dim antw
        antw = msgBox(text & vbCr & "Abort progam?", vbYesNo + vbExclamation,"Problem")
        If antw = vbYes Then WScript.Quit
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Sub output(value)
        IEOutput.Refresh
        IEOutput.Document.write "<font face=""Arial"" size=""4"">" & value & "</font><br>"
        IEOutput.visible = 1
        Do While (IEOutput.Busy)
            Loop
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Sub listOutput(value)
        IEOutput.Document.write "<font face=""Courier New"" size=""2"">" & value & "</font><br>"
        IEOutput.visible = 1
        Do While (IEOutput.Busy)
            Loop
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Sub protocol(text)
        Dim Dateiname,Pfad,TDatei
        Pfad=wshShell.ExpandEnvironmentStrings("%HOMEDRIVE%%HOMEPATH%\Desktop")
        Dateiname=datSystem.BuildPath(Pfad,"vbsprotocol.txt")
        If NOT datSystem.FileExists(Dateiname) Then
            Set TDatei=datSystem.CreateTextFile(Dateiname,True,False)
            TDatei.close
            End If
        If datSystem.FileExists(Dateiname) Then
            Set TDatei=datSystem.OpenTextFile(Dateiname,8,False)
            Tdatei.writeline(text)
            End If
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Private Sub Class_Initialize()
        Dim hoehe, breite, i
        Set IEOutput = WScript.createObject("InternetExplorer.Application")
        IEOutput.menubar = 0
```

```
            IEOutput.toolbar = 0
            IEOutput.FullScreen = False
            IEOutput.TheaterMode = False
            IEOutput.statusbar = 0
            IEOutput.resizable = 1
            IEOutput.height = 200
            IEOutput.width = 950
            IEOutput.top = 500
            IEOutput.left = 0
            IEOutput.navigate ""
            IEOutput.visible = 0
            IEOutput.Document.writeln "<title>Visual Basic Script Output</title>"
            Do While (IEOutput.Busy)
                  Loop
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      End Class

'\_____________/\_____________/\_____________/\_____________/\_____________/

Sub errorDialog(text)
      Dim antw
      antw = msgBox(text & vbCr & "Abort progam?", vbYesNo + vbExclamation,"Problem")
      If antw = vbYes Then WScript.Quit
      End Sub
```

```
Sub TypeControlled(keyCombi, commandline, memConsumDiff)
      Dim colItems, objItem, speicherhungerMarke, spHungerDifferenz : spHungerDifferenz = 0
      WScript.sleep 500
      '-----------------------------------------------------
      '== Prüfe momentane Speicherauslastung der CPL: ==
      On Error Resume Next
      Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
            For Each objItem in colItems
                  If StrComp(objItem.CommandLine,commandline,1) = 0 Then
                        speicherhungerMarke = objItem.WorkingSetSize / 1024
                        End If
                  Next

      WshShell.sendKeys keyCombi
      '-----------------------------------------------------
      '== Erkenne z.B. Öffn. des Fensters an Erhöhung ==
      '== der Speicherauslastung um memConsumDiff KB: ==
      Do
            Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
                  For Each objItem in colItems
                        If StrComp(objItem.CommandLine,commandline,1) = 0 Then
                              spHungerDifferenz = (objItem.WorkingSetSize / 1024) - speicherhungerMarke
                              End If
                  Next
            WScript.sleep 1000
            Loop Until spHungerDifferenz > memConsumDiff
      End Sub


Function screenSize(heightOrWidth)
      ' ... benötigt die String-Parameter "height" oder "width"!
      Dim colItems, objItem, hoehe, breite
      On Error Resume Next
      Set colItems = objWMIService.ExecQuery("Select * from Win32_DisplayConfiguration",,48)
      For Each objItem in colItems
            hoehe = objItem.PelsHeight
            breite = objItem.PelsWidth
            Next
      if heightOrWidth = "height" Then
            screenSize = hoehe
      Else
            if heightOrWidth = "width" Then
                  screenSize = breite
            Else
                  msgBox "Error: Function screenSize() feeded with parameter other than ""height"" or ""width""."
                  End If
            End If
      End Function



Function ScriptPath
```

```
    '(Gibt Skriptpfad mit Querstrich am Ende zurück)
    DIM Path
    path = WScript.ScriptFullName
    ScriptPath = Left(path, InstrRev(path, "\"))
    End Function



Function RegValue(zeichenkette)
    Dim ergebnis
    On Error resume next
    ergebnis = WshShell.RegRead(zeichenkette)
    If Err Then
        ergebnis = "Error"
        RegRead = ""
        Err.clear
        End If
    On Error Goto 0
    RegValue = ergebnis
    End Function

Sub regWriteValue(keyPath, valueName, value, dataType)
    dosCommandLight    "reg add " & _
                       keyPath & " /v """ & _
                       valueName & """ /t " & _
                       dataType & " /d """ & _
                       value & """ /f"
    End Sub

Sub regDeleteKey(keyPath)
    dosCommand "reg delete " & keypath & " /f"
    End Sub

Sub regDeleteValueName(keyPath, valuename)
    dosCommand "reg delete " & keypath & " /v " & valuename & " /f"
    End Sub

Function cdDrive
    Dim i, foundLW
    For Each i in DriveList
        if 4 = i.DriveType Then
            foundLW = i.DriveLetter
            End If
        Next
        cdDrive = foundLW
    End Function



Sub killPrecisely(usedCommandLine)
    Dim colItems, objItem, processID
    '--------------------------------------------------
    '== Stelle Prozess-ID fest & beende: ==
    On Error Resume Next
    Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
        For Each objItem in colItems
            If StrComp(objItem.CommandLine,usedCommandLine,1) = 0 Then
                objItem.Terminate()
                End If
            Next
    End Sub



Sub kill(processName)
    Dim colProcessList, objProcess
    Set colProcessList = ObjWMIService.ExecQuery("Select * from Win32_Process Where Name = '"& processName &"'")
    For Each objProcess in colProcessList
        objProcess.Terminate()
        Next
    End Sub


Sub killperID(procID)
    Dim colProcessList, objProcess
    Set colProcessList = ObjWMIService.ExecQuery("Select * from Win32_Process Where ProcessID = '"& procID &"'")
    For Each objProcess in colProcessList
        objProcess.Terminate()
        Next
    End Sub
```

```
Sub killOwnDuplicates
	Dim colItems, objItem, processID, numOfCopies : numOfCopies = 0
	'-----------------------------------------------------
	'== Stelle Prozess-ID fest & beende: ==
	On Error Resume Next
	Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
		For Each objItem in colItems
			If StrComp(objItem.CommandLine,"""" & WScript.FullName & """ """ & WScript.ScriptFullName & """ ",1) = 0 OR StrComp(objItem.CommandLine,"""" & WScript.FullName & """ """ & WScript.ScriptFullName & """",1) = 0 Then
				numOfCopies = numOfCopies + 1
				If numOfCopies > 1 Then
					killperID(processID)
					Do While processIDExists(processID) = True
						WScript.sleep 33
						Loop
					numOfCopies = numOfCopies -1
					End If
				processID = objItem.ProcessID
				End If
			Next
	End Sub



Function processIDExists(processID)
	Dim colItems, objItem, prExists : prExists = False
	'-----------------------------------------------------
	'== Stelle Prozess-ID fest & vergleiche: ==
	On Error Resume Next
	Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
		For Each objItem in colItems
			If processID = objItem.ProcessID Then prExists = True
			Next
	processIDExists = prExists
	End Function



Function commandLineExists(commandline)
	Dim colItems, objItem, clExists : clExists = False
	'-----------------------------------------------------
	'== Stelle Prozess-ID fest & vergleiche: ==
	On Error Resume Next
	Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
		For Each objItem in colItems
			If commandline = objItem.CommandLine Then clExists = True
			Next
	commandLineExists = clExists
	End Function


Sub independentRun(command)
	Dim objStartup , objConfig , objProcess , intProcessID , errReturn
	Const HIDDEN_WINDOW = 12 'Verhindere DOS-Fenster
	Set objStartup = objWMIService.Get("Win32_ProcessStartup")
	Set objConfig = objStartup.SpawnInstance_
	objConfig.ShowWindow = HIDDEN_WINDOW
	Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
	errReturn = objProcess.Create("cmd /c " & command, null, objConfig, intProcessID)
	End Sub



Class OwnProcess
	Public processID
	'||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
	Sub start(processName, processParameters)
		Dim colItems, objItem, runnerProcessID, commandlineVersionA, commandlineVersionB, commandlineVersionC, commandlineVersionD, foundCommandLine, systemroot : systemroot = WshShell.ExpandEnvironmentStrings("%SYSTEMROOT%")
		processParameters = replaceAll(processParameters,"""","&quot;")' Anfü.-zeichen aus Parameter umgewandelt, um ihrer Entfernung durch independentRun() vorzubeugen
		independentRun """" & pathPartsInQuotes(ScriptPath & "resources\") & "veryOwnProcessRunner.vbs """ & processName & """ """ & processParameters & """"
		commandlineVersionA = """" & systemroot & "\System32\WScript.exe"" """ & ScriptPath & "resources\veryOwnProcessRunner.vbs"" """ & processName & """ """ & processParameters
		commandlineVersionB = """" & systemroot & "\System32\WScript.exe"" """ & ScriptPath & "resources\veryOwnProcessRunner.vbs""  """ & processName & """ """ & processParameters
		commandlineVersionC = """" & systemroot & "\System32\WScript.exe"" """ & ScriptPath & "resources\veryOwnProcessRunner.vbs"" """ & processName & """ """ & processParameters
		commandlineVersionD = """" & systemroot & "\System32\WScript.exe"" """ & ScriptPath & "resources\veryOwnProcessRunner.vbs""  """ & processName & """ """ & processParameters
		Do
			WScript.sleep 10
			Loop Until commandLineExists(commandlineVersionA) OR commandLineExists(commandlineVersionB) OR commandLineExists(commandlineVersionC) OR commandLineExists(commandlineVersionD)
		'-----------------------------------     ----------
		'== Ermittle derzeitige ID des Prozessrunners: ==
```

```
        On Error Resume Next
        Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
            For Each objItem in colItems
                If StrComp(objItem.CommandLine,commandlineVersionA,1) = 0 OR StrComp(objItem.CommandLine,commandlineVersionB,1) = 0 OR StrComp(objItem.CommandLine,
commandlineVersionC,1) = 0 OR StrComp(objItem.CommandLine,commandlineVersionD,1) = 0  Then
                    runnerProcessID = objItem.ProcessID
                    objItem.Terminate()
                    End If
                Next
        '-----------------------------------------------------------------
        '== Suche Proz., dessen ParentPrID der runnerProcessID entspricht: ==
        On Error Resume Next
        Set colItems = objWMIService.ExecQuery("Select * from Win32_Process",,48)
            For Each objItem in colItems
                If objItem.ParentProcessID = runnerProcessID Then
                    processID = objItem.ProcessID
                    prIDsCounter = prIDsCounter + 1
                    End If
                Next
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Sub kill()
        killperID(processID)
        End Sub
    End Class
```

```
Function CurrentLoadPercentage
    DIM colItems, objItem, result
    On Error Resume Next
    Set colItems = objWMIService.ExecQuery("Select * from Win32_Processor",,48)
    For Each objItem in colItems
        result = objItem.LoadPercentage
        Next
    CurrentLoadPercentage = result
    End Function



Function AverageLoadPercentage(periodInSeconds)
    Dim lpSammlung :    lpSammlung = 0
    Dim lapNumber :     lapNumber = 0
    Dim lpAverage
    intern_clock_start
    Do
        lapNumber = lapNumber + 1
        lpSammlung = lpSammlung + CurrentLoadPercentage
        Loop Until intern_clock_stop > periodInSeconds
    lpAverage = lpSammlung / lapNumber
    AverageLoadPercentage = lpAverage
    End Function
```

```
'/‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾\
'      == H O T K E Y - F E A T U R E ==


    '#################################################################
    Dim vbsIntern_kombis(100), vbsIntern_kombiNummer : vbsIntern_kombiNummer = -1
    Dim hotkeyLinkLocation : hotkeyLinkLocation  = WshShell.SpecialFolders("StartUp")
    '#################################################################


    Function newHotkey(kombi)
        '==Deklarationen:==
            Dim linkLocation, oShellLink, fileToBeDeleted
        '==Weise der Tastenkombination Nummer zu:==
            vbsIntern_kombiNummer = vbsIntern_kombiNummer + 1
        '==Lösche eventuell noch vorhandenen Signalgeber:==
            fileToBeDeleted = wshShell.ExpandEnvironmentStrings("%HOMEDRIVE%\vbsKeySignal.dat")
            If datSystem.FileExists(fileToBeDeleted) Then
                datSystem.DeleteFile fileToBeDeleted, True
                End If
        '==Erstelle geheime Verknüpfung incl. Tastenkombination:==
```

```
        Set oShellLink = WshShell.CreateShortcut(hotkeyLinkLocation & "\vbsKeyListener" & vbsIntern_kombiNummer & ".lnk")
        oShellLink.TargetPath = ScriptPath & "resources\keypressSignalizer.vbs"
        oShellLink.Arguments = kombi
        oShellLink.WindowStyle = 1
        oShellLink.Hotkey = kombi
        oShellLink.Description = "Shortcut Script"
        oShellLink.WorkingDirectory = ScriptPath
        oShellLink.Save
        vbsIntern_kombis(vbsIntern_kombiNummer) = kombi
    '==Vorsorge gegen fehlerhaftes Verbleiben der LNKs im Autostart:==
        datSystem.CopyFile ScriptPath & "resources\keyCombinationRemover.vbs", hotkeyLinkLocation & "\", True
        newHotkey = vbsIntern_kombiNummer
    End Function




Function hotkeyPressed(kombiNummer)
    '==Deklarationen:==
        Dim signaldatei, signalDateinameMitPfad, zeile, returnValue : returnValue = False
        signalDateinameMitPfad = wshShell.ExpandEnvironmentStrings("%HOMEDRIVE%\vbsKeySignal.dat")
    '==Prüfe ob ein Signal vorhanden;     ==
    '==  falls ja, prüfe ob richtige Kombi:==
        If datSystem.FileExists(signalDateinameMitPfad) Then
            WScript.Sleep 111
            Set signaldatei = datSystem.OpenTextFile(signalDateinameMitPfad,1,False)
            zeile = signaldatei.ReadLine
            signaldatei.Close
            If zeile = vbsIntern_kombis(kombiNummer) Then
                datSystem.DeleteFile signalDateinameMitPfad, True
                returnValue = True
                End If
            End If
        hotkeyPressed = returnValue
    End Function




Sub removeHotkeys
    Dim linkDateinameMitPfad, i
    '==Lösche die Hotkey-Verknüpfungen:==
        For i = 0 To vbsIntern_kombiNummer
            linkDateinameMitPfad = hotkeyLinkLocation &  "\vbsKeyListener" & i & ".lnk"
            If datSystem.FileExists(linkDateinameMitPfad) Then
                datSystem.DeleteFile linkDateinameMitPfad, True
                End If
            Next
    '==Lösche Autostart-Reiniger:==
        If datSystem.FileExists(hotkeyLinkLocation &  "\keyCombinationRemover.vbs") Then
            datSystem.DeleteFile hotkeyLinkLocation &  "\keyCombinationRemover.vbs", True
            End If
    '==Lösche eventuell noch vorhandenen Signalgeber:==
        Dim fileToBeDeleted : fileToBeDeleted = wshShell.ExpandEnvironmentStrings("%HOMEDRIVE%\vbsKeySignal.dat")
        If datSystem.FileExists(fileToBeDeleted) Then
            datSystem.DeleteFile fileToBeDeleted, True
            End If
    End Sub

'\_____________/\_____________/\_____________/\_____________/\_____________/




'/_____________\/_____________\/_____________\/_____________\/_____________\
'     ==D I A L O G - F E A T U R E S==

    Class Dialog
        '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
        Private htaDialogFilename
        Private htaProcess
        Private htaAttributes
        Private currentXPos, currentYPos
        Private commandListFileName
        '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
        public Sub show()
            javaScriptCommand   "window.moveTo(" & currentXPos & "," & currentYPos & ");"
            End Sub
```

```
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub hide()
	javaScriptCommand  "window.moveTo(" & currentXPos & "," & screenSize("height")-100 & ");"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub showCaption(value)
	Dim answer
	If value = True Then answer = "yes" Else   answer = "no"
	javaScriptCommand  "htaDialog.caption = """ & answer & """; renewHtaAttributes();"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub setIcon(filename)
	filename = replaceAll(filename, "\", "\\")
	javaScriptCommand  "htaDialog.icon = """ & filename & """; renewHtaAttributes();"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub titleBarButtons(value)
	If value=True Then
		javaScriptCommand "htaDialog.sysMenu=""yes""; renewHtaAttributes();"
	Else
		javaScriptCommand "htaDialog.sysMenu=""no""; renewHtaAttributes();"
		End If
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub setTitle(title)
	javaScriptCommand  "document.title = """ & title & """;"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub scrollbars(value)
	Dim answer
	If value = True Then answer = "yes" Else   answer = "no"
	javaScriptCommand  "htaDialog.scroll = """ & answer & """; renewHtaAttributes();"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub border(borderStrength, borderStyle, innerBorder)
	javaScriptCommand  "htaDialog.border = """ & borderStrength & """;" & _
			   "htaDialog.borderStyle = """ & borderStyle & """;" & _
			   "htaDialog.innerBorder = """ & innerBorder & """; renewHtaAttributes();"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub moveTo(x,y)
	javaScriptCommand  "window.moveTo(" & x & "," & y & ");"
	currentXPos = x : currentYPos = y
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub resizeTo(width,height)
	javaScriptCommand  "window.resizeTo(" & width & "," & height & ");"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub addButton(name, aufschrift, x, y)
	javaScriptCommand  "document.all.tags(""body"")[0].innerHTML += ""<button id=\""knopf1\"" style=\""position:absolute;top:100px;left:0px;\""
onclick=\""reportClick(this.id)\"">" & aufschrift & "</button>"";"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub positionElement(name, x, y)
	javaScriptCommand  "document.getElementById('" & name & "').style.left = """ & x & "px"";" & _
			   "document.getElementById('" & name & "').style.top = """ & y & "px"";"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Sub removeElement(name)
	javaScriptCommand  "document.getElementsByTagName(""body"")[0].removeChild(document.getElementById(""" & name & """));"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
public Function eventBy(name)
	Dim returnVal : returnVal = False
	Dim eventFileName : eventFileName = HOMEDRIVE & "\eventBy" & name
	If datSystem.FileExists(eventFileName) Then
		datSystem.deleteFile eventFileName, True
		returnVal = True
		End If
	eventBy = returnVal
	End Function
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
Function javaScriptCommand(commands)
	Dim commandList
	Do Until Not datSystem.FileExists(commandListFileName)
		WScript.sleep 33
		Loop
	Set commandList = datSystem.CreateTextFile(HOMEDRIVE & "\commandList.dat", True, False)
	commandList.WriteLine commands
	commandList.close
	End Function
'||||||||||||||||||||||||||||||||||||||||||||||||||||||
Public Sub Class_Initialize()
```

```
            'Besetzung der klassenweiten Variablen
            commandListFileName = HOMEDRIVE & "\commandList.dat"
            htaDialogFilename = ScriptPath & "resources\htaDialog.hta"
            'Eventuelle CommandList-Überreste vor Beginn löschen:
            If datSystem.FileExists(commandListFileName) Then
                  datSystem.DeleteFile commandListFileName, True
                  End If
            'HTA-Dialog-Prototyp starten:
            Set htaProcess = new OwnProcess
            htaProcess.start "mshta", """ & htaDialogFilename & """
            currentXPos = 200 : currentYPos = 200                  'Kommen beim nä. show() zum Einsatz
            'Flache Scrollbalken einrichten:
            javaScriptCommand "htaDialog.scrollFlat=""yes""; renewHtaAttributes();"
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub Quit
            htaProcess.kill
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      End Class


Class ieDialog
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      Public ieWindow
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub display()
            ieWindow.visible = 1
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub Hide()
            ieWindow.visible = 0
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub top(wert)
            ieWindow.top = wert
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub left(wert)
            ieWindow.left = wert
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub height(wert)
            ieWindow.height = wert
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub width(wert)
            ieWindow.width = wert
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub setBgColor(wert)
            Dim ausgabebereich
            Set ausgabebereich = ieWindow.document.getElementsByTagName("body")(0)
            ausgabebereich.setAttribute "bgColor",wert
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub setBgPicture(file)
            Dim ausgabebereich
            Set ausgabebereich = ieWindow.document.getElementsByTagName("body")(0)
            ausgabebereich.setAttribute "background",file
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub addStyle(style)
            Dim ausgabebereich, styleTagElement
            Set ausgabebereich = ieWindow.document.getElementsByTagName("head")(0)
            Set styleTagElement = ieWindow.document.createElement("link")
            styleTagElement.setAttribute "rel","stylesheet"
            styleTagElement.setAttribute "type","text/css"
            styleTagElement.setAttribute "href", ScriptPath & "resources\dialogStyle_" & style & ".css"
            ausgabebereich.appendChild styleTagElement
            End Sub
      '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
      public Sub addButton(aufschrift, name)
            Dim link, linkText, umbruch, ausgabebereich
            '== Button hinzufügen: ==
                  Set ausgabebereich = ieWindow.document.getElementsByTagName("body")(0)
                  Set link = ieWindow.document.createElement("a")
                  Set linkText = ieWindow.document.createTextNode(aufschrift)
                  link.setAttribute "href","javascript:registerClick(""" & name & """)"
                  link.appendChild linkText
                  ausgabebereich.appendChild link
            '== Damit Links nicht verkleben: ==
                  Set umbruch = ieWindow.document.createElement("br")
                  ausgabebereich.appendChild umbruch
```

```
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    public Function ButtonKlickedIs(name)
        Dim buttonListener, status : status = False
        If ieWindow.document.getElementById("buttonTester").value = "" & name & "" Then
        WScript.Sleep 33
            status = True
            ieWindow.document.getElementById("buttonTester").value = "none"
            End If
        ButtonKlickedIs = status
        End Function
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Sub progressbar_set(x,y,barwidth)
        callJavaScriptFunction "progressbar_set(" & x & "," & y & "," & barwidth & ")"
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Sub progressbar_fill(slowness)
        callJavaScriptFunction("progressbar_fill(" & slowness & ")")
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Sub progressbar_stop
        callJavaScriptFunction "progressbar_stop()"
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Function callJavaScriptFunction(functionName)
        Dim returnValue
        ieWindow.Document.getElementById("caller").value = functionName
        WScript.sleep 33
        returnValue = ieWindow.Document.getElementById("returner").value
        ieWindow.Document.getElementById("returner").value = "none"
        callJavaScriptFunction = returnValue
        End Function
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Sub Class_Initialize()
        Set ieWindow = WScript.createObject("InternetExplorer.Application")
        ieWindow.menubar = 0
        ieWindow.toolbar = 0
        ieWindow.FullScreen = True
        ieWindow.TheaterMode = False
        ieWindow.statusbar = 0
        ieWindow.resizable = 0
        ieWindow.height = 300
        ieWindow.width = 400
        ieWindow.top = 0
        ieWindow.left = 0
        ieWindow.navigate ScriptPath & "resources\standard.htm"
        ieWindow.visible = 0                    'bei visible-Wert 0 bleibt IE unsichtbar
        Do While (ieWindow.Busy)
            Loop
        ieWindow.document.writeln("<html id=""dokument"">")
        ieWindow.document.writeln("<head>")
        ieWindow.document.writeln("    <script language=""JavaScript"" src=""" & ScriptPath & "resources\dialogJavascript.js""></script>")
        ieWindow.document.writeln("</head>")
        ieWindow.document.writeln("<body scroll=""no"">")
        ieWindow.document.writeln("    <script language=""JavaScript"">window.setInterval(""callReceiver()"",1);</script>")   ' Funktion callReceiver() befindet sich in "dialogJavascript.js"
        ieWindow.document.writeln("")
        ieWindow.document.writeln "<input type=""hidden"" value=""none"" name=""feld1"" id=""buttonTester"">"
        ieWindow.document.writeln "<input type=""hidden"" value=""none"" name=""feld2"" id=""caller"">"
        ieWindow.document.writeln "<input type=""hidden"" value=""none"" name=""feld3"" id=""returner"">"
        ieWindow.document.writeln "</body>"
        ieWindow.document.writeln "</html>"
        ieWindow.Refresh
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    public Sub Close
        ieWindow.Quit
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    End Class


'\_____________/\_____________/\_____________/\_____________/\_____________/


'/             \/             \/             \/             \/             \
'    == SCREENSHOT-FEATURE ==

    Class Screenshot
```

```
Public grabberProzess
Public jpgQuality
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Public Sub Class_Initialize()
	Set grabberProzess = new OwnProcess
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Public Sub areaToClipboard()
	kill "PrintScreen.exe"  'Vorher alle laufenden Instanzen löschen
	grabberProzess.start  ScriptPath & "resources\PrintScreen", _
			"/justnow /notaskbar /nosplash /clipboard /dir=" & HOMEDRIVE & "\ " & _
			"/autoname=no /quiet /preview=no /showmsg=no /cptarea=3"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Public Sub toFile(filename)
	runShooter filename, ""
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Public Sub areaToFile(filename)
	runShooter filename, " /cptarea=3"
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Public Sub runShooter(filename, captureAreaArgument)
	'(In dieser Prozedur noch viele Effizienzoptimierungsmöglichkeiten!)
	Dim imgType
	Dim fExt :		fExt = datSystem.GetExtensionName(filename)
	Dim theSame :		theSame = 0
	Dim unwishedFileName :	unwishedFilename =  datSystem.GetParentFolderName(filename) & "\" & _
					datSystem.GetBaseName(filename) & "001" & "." & fExt
	If StrComp(fExt, "bmp", 1) = theSame Then
		imgType = 0
	Else If StrComp(fExt, "jpg", 1) = theSame Then
		imgType = 1
	Else If StrComp(fExt, "gif", 1) = theSame Then
		imgType = 2
	Else If StrComp(fExt, "png", 1) = theSame Then
		imgType = 3
	Else If StrComp(fExt, "tif", 1) = theSame Then
		imgType = 4
	Else If StrComp(fExt, "tga", 1) = theSame Then
		imgType = 5
	Else errorDialog "Could not identify file extension."
		End If : End If : End If : End If : End If : End If
	'==Starte Screenshot-Programm:==
	kill "PrintScreen.exe"  'Vorher alle laufenden Instanzen löschen
	grabberProzess.start  ScriptPath & "resources\PrintScreen", _
			"/justnow /notaskbar /nosplash /dir=""" & datSystem.GetParentFolderName(filename) & """ " & _
			"/file=" & datSystem.GetBaseName(filename) & " /image=" & imgType & _
			" /quiet /preview=no /showmsg=no /autoname=0" & captureAreaArgument & _
			" /jpgqual=" & jpgQuality
	End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
End Class

'\_____________/\_____________/\_____________/\_____________/\_____________/

'/_____________\/_____________\/_____________\/_____________\/_____________\
'	==STRINGS-FEATURE==

Function occurrencesIn(mainString,searchedElement)
	Dim mStrLength
	Dim occ : occ = 0
	Dim anf : anf = 1
	mStrLength = Len(mainString)
	Do Until InStr(anf,mainString,searchedElement)=0
		occ = occ +1
		anf = InStr(anf,mainString,searchedElement)+1
		Loop
	occurrencesIn = occ
	End Function
```

```
Function replaceAll(sentence, toBeReplaced, newElement)
	Dim anzahl
	anzahl = occurrencesIn(sentence, toBeReplaced)
	replaceAll = Replace(sentence, toBeReplaced, newElement, 1, anzahl)
	End Function



Function stringsEqual(kette1, kette2)
	Dim ergebnis
	Const notCASESENSITIVE = 1
	Const EQUAL = 0
	ergebnis = StrComp(kette1, kette2, notCASESENSITIVE)
	if ergebnis = EQUAL Then
		stringsEqual = True
	Else
		stringsEqual = False
	End If
	End Function


Function stringsExactlyEqual(kette1, kette2)
	Dim ergebnis
	Const CASESENSITIVE = 0
	Const EQUAL = 0
	ergebnis = StrComp(kette1, kette2, CASESENSITIVE)
		if ergebnis = EQUAL Then
			stringsExactlyEqual = True
		Else
			stringsExactlyEqual = False
		End If
		End Function


	Function stringPartBetweenMarks(mark1, mark2, strData)
		Dim pos1, pos2, file
		pos1 = Instr(1, strData, mark1)  + Len(mark1)
		pos2 = Instr(pos1, strData, mark2)
		stringPartBetweenMarks = Mid(strData, pos1, pos2-pos1)
		strData = ""  : strData = Empty  ' Speicherfreigabe, da Text sehr lang sein könnte
		End Function

'\_____________/\_____________/\_____________/\_____________/\_____________/




'_____________\/_____________\/_____________\/_____________\/_____________\
'     == I/O - FEATURES==


	Function textfileToArray(filename)
		Dim textstream
		Dim content : Dim contentList
		If datSystem.FileExists(filename) Then
			Set textstream = datSystem.OpenTextFile(filename, 1, False)
		Else
			readTextfile = "error"
			Exit Function
		End if
		Do While NOT textstream.AtEndOfStream
			If NOT content = empty Then content = content & "\r\n"
			content = content & textstream.ReadLine()
			Loop
		textstream.Close
		contentList = Split(content, "\r\n")
		textfileToArray = contentList
		End Function

Function readBetweenMarks(mark1, mark2, filename)
	Dim text, pos1, pos2, file, textstream
	Const ForReading = 1, ForWriting = 2, TristateUseDefault = -2
	Set file = datSystem.GetFile(filename)
	Set textstream = file.OpenAsTextStream(ForReading, TristateUseDefault)
	Do While Not textstream.AtEndOfStream
		text = text & textstream.read(1)
		Loop
	textstream.close
	pos1 = Instr(1, text, mark1)  + Len(mark1)
```

```
        pos2 = Instr(pos1, text, mark2)
        readBetweenMarks = Mid(text, pos1, pos2-pos1)
        text = "" : text = Empty    ' Speicherfreigabe, da Text sehr lang sein könnte
        End Function


Sub makeFile(content, filename)
        Dim textstream : Dim contentList : Dim line
        Set textstream = datSystem.CreateTextFile(filename, True, False)
        contentList = Split(content, "\r\n")
        For Each line in contentList
                textstream.WriteLine(line)
                Next
        textstream.close
        End Sub


Sub saveText(content, filename)
        Dim textstream : Dim contentList : Dim line
        Set textstream = datSystem.OpenTextFile(filename,8,True)
        contentList = Split(content, "\r\n")
        For Each line in contentList
                textstream.WriteLine(line)
                Next
        textstream.close
        End Sub


        Sub WriteBinary(filename, data)
                dim file, textstream, element
                Const ForReading = 1, ForWriting = 2
                datSystem.CreateTextfile(filename)
                Set file = datSystem.GetFile(filename)
                Set textstream = file.OpenAsTextstream(ForWriting, -2)
                For Each element In data
                        textstream.write Chr(element)
                        Next
                textstream.close
                End Sub


'\_____________/\_____________/\_____________/\_____________/\_____________/


'/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\
'     ==INTERNET/NETWORKING FEATURES==


        '/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\

        Dim socketServersAmount : socketServersAmount = 0
        '--------------------------------
        Function SocketServer
        'Diese Funktion ist eine Art Schnittstelle zwischen dem es aufrufenden Programm und der Klasse 'SocketServerObject'.
        ' Nur durch diese Funktion kann das aufrufende Programm beliebige Namen für die SocketServer-Objekte benutzen, da
        ' die Event-Methoden 'OnConnectionRequest' und 'OnDataArrival' global generiert werden und somit nur auf feste Namen
        ' zugreifen können.
        Dim socketServerName : socketServerName = "socketServer" & socketServersAmount
        ExecuteGlobal    "Dim " &  socketServerName & vbCr & _
                         "Set " & socketServerName & " = new SocketServerObject"
        socketServersAmount = socketServersAmount+ 1
        Execute "Set SocketServer = " & socketServerName
        End Function
'--------------------------------
Class SocketServerObject
        Public socket, port, newData, gotNewData
        '--------------------------------
        Private Sub Class_Initialize()
                port = randomNumber(16383)+49152      ' beliebiger Dynamic Port zwischen incl. 49152 und incl. 65535
                newData = "" : gotNewData = False
                Dim noError : noError = True
                dosCommand    "regsvr32 /s """ & ScriptPath & "resources\oswinsck.dll"""
                'Beruhige Windows-Firewall:
                regWriteValue    "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List", _
                                 WScript.FullName, _
                                 WScript.FullName & ":*:Enabled:Microsoft (r) Windows Based Script Host", _
                                 "REG_SZ"
                Set socket = CreateObject("OSWINSCK.Winsock")
                'Bereite reagierende Methoden vor:
                ExecuteGlobal "Sub socket" & socketServersAmount & "_OnConnectionRequest(requestID)"   & vbCr & _
```

```
                    "socketServer" & socketServersAmount & ".socket.CloseWinsock"              & vbCr & _
                    "socketServer" & socketServersAmount & ".socket.Accept Clng(requestID)"    & vbCr & _
                    "End Sub"
        ExecuteGlobal "Sub socket" & socketServersAmount & "_OnDataArrival(ByVal bytesTotal)"  & vbCr & _
                    "Dim newData"                                                              & vbCr & _
                    "socketServer" & socketServersAmount & ".socket.getData newData"           & vbCr & _
                    "socketServer" & socketServersAmount & ".newData = newData"                & vbCr & _
                    "socketServer" & socketServersAmount & ".gotNewData = True"                & vbCr & _
                    "End Sub"
        'Teile dem Objekt reagierende Methoden mit:
        WScript.ConnectObject socket, "socket" & socketServersAmount & "_"
        Do '(Schleife)
            On Error Resume Next
            socket.LocalPort = port
            socket.Listen                               'Versuche Port abzuhören.
            If Err Then                                     'Falls besetzt...
                port = randomNumber(16383)+49152            '...nimm anderen Port.
                noError = False
            Else noError = True
                End If
            On Error Goto 0
            Loop Until NoError                          'Wiederhole Voriges bis unbesetzten Port gefunden.
        End Sub
'------------------------------
Public Function data
        data = newData
        newData = ""
            gotNewData = False
            End Function
        '------------------------------
        public Sub answer(data)
            socket.SendData data
            End Sub
        '------------------------------
        Private Sub Class_Terminate()
            dosCommand          "regsvr32 /s /u """ & ScriptPath & "resources\oswinsck.dll"""
            regDeleteValueName  "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List", _
                                WScript.FullName
            End Sub
        End Class

'\_____________/\_____________/\_____________/




'/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\/‾‾‾‾‾‾‾‾‾‾‾‾‾\

Dim socketClientsAmount : socketClientsAmount = 0
'------------------------------
Function SocketClient(serverIP, port)
    'Diese Funktion ist eine Art Schnittstelle zwischen dem es aufrufenden Programm und der Klasse 'SocketClientObject'.
    ' Nur durch diese Funktion kann das aufrufende Programm beliebige Namen für die SocketClient-Objekte benutzen, da
    ' die Event-Methode 'OnConnect'  global generiert wird und somit nur auf feste Namen zugreifen kann.
    dosCommand "regsvr32 /s """ & ScriptPath & "resources\oswinsck.dll"""
    Dim SocketClientName : SocketClientName = "SocketClient" & socketClientsAmount
    'Bereite reagierende Methoden vor:
    ExecuteGlobal   "Sub socket" & socketClientsAmount & "_OnConnect()"                 & vbCr & _
                    "SocketClient" & socketClientsAmount & ".isConnected = True"    & vbCr & _
                    "End Sub"
    ExecuteGlobal   "Sub socket" & socketClientsAmount & "_OnDataArrival(ByVal bytesTotal)"  & vbCr & _
                    "Dim newData"                                                           & vbCr & _
                    "SocketClient" & socketClientsAmount & ".socket.getData newData"            & vbCr & _
                    "SocketClient" & socketClientsAmount & ".newData = newData"                 & vbCr & _
                    "SocketClient" & socketClientsAmount & ".gotNewAnswer = True"               & vbCr & _
                    "End Sub"
    'Erzeuge SocketClientObject, das gleich zurückgegeben wird
    ExecuteGlobal   "Dim " &  SocketClientName                                          & vbCr & _
                    "Set " & SocketClientName & " = new SocketClientObject"
    'Gebe neues Objekt zurück, mit den gewünschten Attributen
    Execute                                         _
    "Set SocketClient = " & SocketClientName  & vbCr & _
    "SocketClient.serverIP = """ & serverIP & """"      & vbCr & _
    "SocketClient.port = " & port
    SocketClient.isConnected = False
    SocketClient.gotNewAnswer = False
    Set SocketClient.socket = CreateObject("OSWINSCK.Winsock")
    WScript.ConnectObject SocketClient.socket, "socket" & socketClientsAmount & "_"
    socketClientsAmount = socketClientsAmount+ 1
    SocketClient.socket.Connect serverIP, port
    End Function
'------------------------------
```

```
Class SocketClientObject
	public isConnected, socket, serverIP, port,  newData, gotNewAnswer
	'--------------------------------
	public Sub sendData(data)
		socket.SendData data
		End Sub
	'--------------------------------
	Public Function data
		data = newData
		newData = ""
		gotNewAnswer = False
		End Function
	'--------------------------------
	Private Sub Class_Terminate()
		dosCommand		"regsvr32 /s /u """ & ScriptPath & "resources\oswinsck.dll"""
		End Sub
	End Class

'\_____________/\_____________/\_____________/




Function isOnline
	Dim output
	'Gehe zunächst von Offline-Status aus:
	isOnline = False
	'Versuche, Microsoft-Seite zu erreichen:
	output = dosCommand("ping www.microsoft.com -n 1")
	'Falls microsoft.com-IP "207.46..." gefunden, stelle Online-Status fest:
	if occurrencesIn(output, "207.46") > 0 Then
		isOnline = True
		Exit Function
		End If
	'Vielleicht nur Microsoft-Server nicht erreichbar und PC trotzdem online, darum yahoo.com probieren:
	output = dosCommand("ping www.yahoo.com -n 1")
	'Falls yahoo.com-IP "69.147..." gefunden, stelle Online-Status fest:
	if occurrencesIn(output, "69.147") > 0 Then
		isOnline = True
		Exit Function
		End If
	'Vielleicht nur Yahoo-Server nicht erreichbar und PC trotzdem online, darum denic.de probieren:
	output = dosCommand("ping www.denic.de -n 1")
	'Falls denic.de-IP "81.91..." gefunden, stelle Online-Status fest:
	if occurrencesIn(output, "81.91") > 0 Then
		isOnline = True
		Exit Function
		End If
	End Function

Function currentIPs
	Dim IPConfigSet, IPConfig, ipList, i, strComputer : strComputer = "."
	'Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
	Set IPConfigSet = objWMIService.ExecQuery("Select IPAddress from Win32_NetworkAdapterConfiguration where IPEnabled=TRUE")
	For Each IPConfig in IPConfigSet
		If Not IsNull(IPConfig.IPAddress) Then
			For i=LBound(IPConfig.IPAddress) to UBound(IPConfig.IPAddress)
				ipList = ipList & IPConfig.IPAddress(i) & "\r\n"
				Next
			currentIPs = Split(ipList, "\r\n")
			End If
		Next
	End Function

Function currentIP
	Dim IPsArray : IPsArray = currentIPs
	currentIP = IPsArray(0)
	End Function

Function urlEncode(sRawURL)
	On Error Resume Next
	Dim iLoop, sRtn, Tmp, sTmp
	Const sValidChars = "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz:/.?=_-$(){}~&"
	If Len(sRawURL) > 0 Then
		' Loop through each char
		For iLoop = 1 To Len(sRawURL)
			sTmp = Mid(sRawURL, iLoop, 1)
			If InStr(1, sValidChars, sTmp, vbBinaryCompare) = 0 Then
				' If not ValidChar, convert to HEX and p
				'	refix with %
				sTmp = Hex(Asc(sTmp))
				If sTmp = "20" Then
					sTmp = "+"
```

```
                        ElseIf Len(sTmp) = 1 Then
                            sTmp = "%0" & sTmp
                        Else
                            sTmp = "%" & sTmp
                        End If
                    End If
                    sRtn = sRtn & sTmp
                    Next
                urlEncode = sRtn
                End If
        Exit Function
        If Err Then urlEncode = ""
        On Error Goto 0
        End Function


'/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
'      == U P L O A D E R - F E A T U R E ==

Class Uploader
        '||||||||||||||||||||||||||||||||||||||||||||||||||||||
        Private uploadPage
        Private htaFilename
        Private bereitschaftsNr
        Private htaProcess
        '||||||||||||||||||||||||||||||||||||||||||||||||||||||
        Public Sub disconnect
            htaProcess.kill
        If datSystem.FileExists(htaFilename) Then
            datSystem.DeleteFile htaFilename, True
            End If
        If datSystem.FileExists(HOMEDRIVE & "\uploaderBusy") Then
            datSystem.DeleteFile HOMEDRIVE & "\uploaderBusy"
            End If
        If datSystem.FileExists(HOMEDRIVE & "\uploaderNotBusy") Then
            datSystem.DeleteFile HOMEDRIVE & "\uploaderNotBusy"
            End If
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Function busy
        If datSystem.FileExists(HOMEDRIVE & "\uploaderBusy") Then
            WScript.sleep 33 : busy = true
        Else If datSystem.FileExists(HOMEDRIVE & "\uploaderNotBusy") Then
            WScript.sleep 33 : busy = False
        Else
            WScript.sleep 33 : busy = true
            End If : End If
        End Function
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Sub signalizeUploaderBusy(signalizeBusiness)
        Dim businessSignalFile
        If signalizeBusiness = true Then
            If Not datSystem.FileExists(HOMEDRIVE & "\uploaderNotBusy") AND Not datSystem.FileExists(HOMEDRIVE & "\uploaderBusy") Then
                Set businessSignalFile = datSystem.CreateTextFile(HOMEDRIVE & "\uploaderBusy")
            Else If datSystem.FileExists(HOMEDRIVE & "\uploaderNotBusy") Then
                Rename HOMEDRIVE & "\uploaderNotBusy", "uploaderBusy"
                End If
            End If
        Else
            If Not datSystem.FileExists(HOMEDRIVE & "\uploaderNotBusy") AND Not datSystem.FileExists(HOMEDRIVE & "\uploaderBusy") Then
                Set businessSignalFile = datSystem.CreateTextFile(HOMEDRIVE & "\uploaderNotBusy")
            Else If datSystem.FileExists(HOMEDRIVE & "\uploaderBusy") Then
                Rename HOMEDRIVE & "\uploaderBusy", "uploaderNotBusy"
                End If
            End If : End If
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Sub upload(file)
        Dim failureNumber : failureNumber = 0
        signalizeUploaderBusy True
        'Wenn Upload-Programm bereit (feststellbar an Erhöhung der Skripttitel-Nr.)...
        Do Until WshShell.AppActivate("Uploader-Script" & bereitschaftsNr)
            failureNumber = failureNumber + 1
            If failureNumber > 100 Then bereitschaftsNr = bereitschaftsNr - 1 ' Erkl.: Falls Skripttitel-Nr. zu niedrig.
            WScript.sleep 33
            Loop
        failureNumber = 0 : WScript.sleep 333
        '... gib Pfadnamen der hochzuladenden Datei ins Feld ein und bestätige:
        WshShell.Sendkeys file & "{ENTER}(%{ESC})"
        bereitschaftsNr = bereitschaftsNr + 1 ' Erkl.: Skripttitel-Nr. bei jedem Upload höher ("Uploader-Skript1", "Uploader-Skript2").
        End Sub
    '||||||||||||||||||||||||||||||||||||||||||||||||||||||
    Public Sub connectWith(url)
        Dim seitentitel, htaDatei
```

```
'==Prüfe ob Uploader.php auf Server vorhanden:==
        uploadPage.navigate url & "/Uploader.php?hta=no"
        Do While (uploadPage.Busy)
                WScript.sleep 33
                Loop
        seitentitel = uploadPage.Document.all.tags("title")(0).innerText
        If NOT seitentitel = "Uploader-Script" AND  occurrencesIn(seitentitel,"404") > 0 Then
                msgBox    "In dem angegebenen Webverzeichnis fehlt das Skript ""Uploader.php""." & vbCr & _
                          "Sie finden das Skript im Utilities-Verzeichnis Ihrer VBSplus-Umgebung." & vbCr & _
                          "Bitte laden Sie dieses vorher hoch oder korrigieren die Web-Pfadangaben.", _
                          vbExclamation + vbOkOnly, _
                          "Upload-Fehler"
                uploadPage.Quit
                WScript.Quit
                End If
        uploadPage.Quit
'==Erstelle Upload-HTA-Applikation:==
        htaFilename = WshShell.ExpandEnvironmentStrings("%TEMP%\uploader.hta")
        Set htaDatei = datSystem.CreateTextFile(htaFilename,True,False)
        htaDatei.WriteLine "<HTML>"
        htaDatei.WriteLine "<HEAD>"
        htaDatei.WriteLine "   <Script language=""Javascript"">"
        htaDatei.WriteLine "         window.moveTo(900,700);"
        htaDatei.WriteLine "         window.resizeTo(100,100);"
        htaDatei.WriteLine "         </Script>"
        htaDatei.WriteLine "   <HTA:Application ID = ""thisHTA"""
        htaDatei.WriteLine "                ApplicationName = ""app"""
        htaDatei.WriteLine "                MaximizeButton = ""no"""
        htaDatei.WriteLine "                MinimizeButton = ""no"""
        htaDatei.WriteLine "                ShowInTaskBar = ""no"""
        htaDatei.WriteLine "                SingleInstance = ""yes"""
        htaDatei.WriteLine "                SysMenu = ""no"""
        htaDatei.WriteLine "                Navigable = ""Yes"""
        htaDatei.WriteLine "                Version = ""1.0"""
        htaDatei.WriteLine "                WindowState = ""normal"""
        htaDatei.WriteLine "                >"
        htaDatei.WriteLine "   </HEAD>"
        htaDatei.WriteLine "<BODY>"
        htaDatei.WriteLine "<Script language=""Javascript"">"
        htaDatei.WriteLine "   location.href = """ & url & "/Uploader.php?hta=yes"""
        htaDatei.WriteLine "   </Script>"
        htaDatei.WriteLine "</BODY>"
        htaDatei.WriteLine "</HTML>"
        htaDatei.Close
        Do Until datSystem.FileExists(htaFilename)
                WScript.sleep 33
                Loop
        Set htaProcess = new OwnProcess
        htaProcess.start "mshta", htaFilename
        WScript.sleep 1000
        WshShell.Sendkeys "(%{ESC})"
    End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Public Sub Class_Initialize()
    'Aufräumen von evtl. zuletzt Liegengebliebenem:
    If datSystem.FileExists(HOMEDRIVE & "\uploaderBusy") Then
        datSystem.DeleteFile HOMEDRIVE & "\uploaderBusy", true
        End if
    If datSystem.FileExists(HOMEDRIVE & "\uploaderNotBusy") Then
        datSystem.DeleteFile HOMEDRIVE & "\uploaderNotBusy", true
        End If
    'Lade Testbrowser vor:
    Set uploadPage = WScript.CreateObject("InternetExplorer.Application")
    uploadPage.visible = 0
    Do While (uploadPage.Busy)
        WScript.sleep 33
        Loop
    End Sub
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||
End Class

'\_____________/\_____________/\_____________/

'\_____________/\_____________/\_____________/\_____________/\_____________/

'/_____________\/_____________\/_____________\/_____________\/_____________\
```

```
'     == O T H E R  F E A T U R E S ==


    Function pathPartsInQuotes(path)
        Dim chain, newPath
        If Not Right(path,1)="\" Then path = path & "\"         'Falls letztes Zeichen kein Backslash, dann füge eins hinzu, damit letztes Pfadelement kein einseitiges Anf.-zeichen bekommt
        chain = Split(path,"\")                                 ' Teile Pfad in Einzelteile
        newPath = Join(chain,"""\""")                           ' Setze ihn mit Anf.-Zeichen neu zusammen
        if Right(newPath,2)="\""" Then newPath = Left(newPath,Len(newPath)-1)   ' Falls Pfad jetzt mit überfl. Anf.-Zeichen endet, weg damit
        pathPartsInQuotes = replaceAll(newPath, ":""",":")      'mach aus C:"\ usw. C:\
        End Function


    Function dosCommandFast(command)
        Dim oExec, input, i
        Set oExec    = WshShell.Exec("cmd /c @Echo Off && " & command)
        input = ""
        Do While NOT oExec.StdOut.AtEndOfStream
            input = input & oExec.StdOut.ReadLine
            Loop
        dosCommandFast = input
        End Function

    Sub dosCommandLight(command)
        Dim objStartup , objConfig , objProcess , intProcessID , errReturn
        Const HIDDEN_WINDOW = 12 'Verhindere DOS-Fenster
        Set objStartup = objWMIService.Get("Win32_ProcessStartup")
        Set objConfig = objStartup.SpawnInstance_
        objConfig.ShowWindow = HIDDEN_WINDOW
        Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
        errReturn = objProcess.Create("cmd /c " & command, null, objConfig, intProcessID)
        End Sub

    Function dosCommand(command)
        Dim objStartup , objConfig , objProcess , intProcessID , errReturn, execOutputFile, execOutputFilename, execOutputStream, output
        execOutputFilename = wshShell.ExpandEnvironmentStrings("%TEMP%\execOutput.dat")
        'Lösche evtl. von früher liegengebliebene Outputs:
        If datSystem.FileExists(execOutputFilename) Then
            On Error Resume Next
            datSystem.DeleteFile(execOutputFilename)
            On Error Goto 0
            End if
        'Übergebene Befehl im Verborgenen an CMD-Konsole:
        Const HIDDEN_WINDOW = 12 'Verhindere DOS-Fenster
        Set objStartup = objWMIService.Get("Win32_ProcessStartup")
        Set objConfig = objStartup.SpawnInstance_
        objConfig.ShowWindow = HIDDEN_WINDOW
        Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
        errReturn = objProcess.Create("cmd /c " & command & " > """ & execOutputFilename & """", null, objConfig, intProcessID)
        'Fange Rückgaben auf:
        Do Until datSystem.FileExists(execOutputFilename)
            Loop
        Set execOutputFile = datSystem.GetFile(execOutputFilename)
        Set execOutputStream = execOutputFile.OpenAsTextStream(1,0)
        If execOutputFile.Size > 0 Then
            Do Until Not execOutputStream.AtEndOfStream
                WScript.sleep 33
                Loop
            dosCommand = execOutputStream.ReadAll
        Else
            dosCommand = ""
        End If
        Do Until execOutputStream.AtEndOfStream
            WScript.sleep 33
            Loop
        execOutputStream.close
        'Verlasse Funktion erst nach Nichtmehr-Existenz der Outputdatei oder bei Fehler:
        On Error Resume Next
        execOutputFile.Delete True
        Do Until Not datSystem.FileExists(execOutputFilename)
            WScript.sleep 33
            execOutputFile.Delete True
            Loop
        On Error Goto 0
        End Function



    Sub changeFileVisibility(filename)
        Dim objFile
        Set objFile = datSystem.GetFile(filename)
        If objFile.Attributes = objFile.Attributes AND 2 Then
            objFile.Attributes = objFile.Attributes XOR 2
            End If
```

```
    End Sub


Sub restartSystem
    DIm colOperatingSystems, objOperatingSystem
    Set colOperatingSystems = GetObject("winmgmts:{(Shutdown)}").ExecQuery("Select * from Win32_OperatingSystem")
    For Each objOperatingSystem in colOperatingSystems
        ObjOperatingSystem.Win32Shutdown(2+4)
        Next
    'Parameter für ObjOperatingSystem.Win32Shutdown(2+4)
    '0 Abmelden
    '0 + 4 Abmelden & Programme zum beenden zwingen(sollten diese sich nicht schließen lassen).
    '1 Herunterfahren
    '1 + 4 Herunterfahren Programme zum beenden zwingen(sollten diese sich nicht schließen lassen).
    '2 Neu starten
    '2 + 4 Neu starten Programme zum beenden zwingen(sollten diese sich nicht schließen lassen).
    '8 Abschalten
    '8 + 4 Abschalten Programme zum beenden zwingen(sollten diese sich nicht schließen lassen).
    '  © Boris Toll 2004
    '  www.boris-toll.at
    End Sub


Function randomNumber(max)
    Dim Zufallszahl
    Randomize
    Zufallszahl = Rnd()
    randomNumber=Round(Zufallszahl*max)
    End Function

Function elementExistsInArray(toBeFound, array)
    Dim element
    For Each element in array
        If element = toBeFound Then
            elementExistsInArray = True
            Exit Function
            End If
        Next
    elementExistsInArray = False
    End Function


Sub endAll
    Set WshShell = Nothing
    Set javascriptInterface = Nothing
    Set shellApp = Nothing
    Set datSystem = Nothing
    Set DriveList = Nothing
    Set objWMIService = Nothing
    WScript.Quit
    End Sub

'\_____________/\_____________/\_____________/\_____________/\_____________/


Function skypeInstalled()
    on error resume next
    Set oSkype = CreateObject("Skype.Detection")
    skypeInstalled = IsObject(oSkype)
    Set oSkype = nothing
    End Function
```