Windows Script Host (WSH) Cheatsheet

A comprehensive cheat sheet for Windows Script Host (WSH), covering essential objects, methods, and properties for automating tasks in Windows environments using VBScript or JScript.

WSH Core Objects

WScript Object

The `WScript` object is the root object for all WSH scripts. It provides access to various properties and methods for controlling the script's execution environment.

Properties:

- `WScript.Arguments`: A collection of command-line arguments passed to the script.
- `WScript.FullName`: The full path to the `WScript.exe` or `CScript.exe` executable.
- `WScript.Name`: The name of the script host (`WScript` or `CScript`).
- `WScript.Path`: The path to the directory containing the script.
- `WScript.ScriptFullName`: The full path to the script file.
- `WScript.ScriptName`: The name of the script file.

Methods:

- `WScript.CreateObject(strProgID)`: Creates an instance of a COM object.
- `WScript.Echo(args)`: Displays a message in a message box or to the console (depending on the host).
- `WScript.GetObject(strPathname, strProgID)`: Retrieves an active COM object or creates a new one from a file.
- `WScript.Quit(intExitCode)`: Terminates the script's execution.
- `WScript.Sleep(intMilliseconds)`: Pauses the script's execution for a specified number of milliseconds.

WshShell Object

The `WshShell` object provides access to the Windows shell environment, allowing scripts to perform tasks such as creating shortcuts, accessing environment variables, and running external programs.

Methods:

- `WshShell.CreateShortcut(strPathname)`: Creates a shortcut file.
- `WshShell.ExpandEnvironmentStrings(strString)`: Expands environment variables in a string.
- `WshShell.LogEvent(intType, strEvent)`: Logs an event to the Windows event log.
- `WshShell.Popup(strText, intSecondsToWait, strTitle, intType)`: Displays a pop-up message box.
- `WshShell.RegRead(strName)`: Reads a value from the Windows registry.
- `WshShell.RegWrite(strName, varValue, strType)`: Writes a value to the Windows registry.
- `WshShell.Run(strCommand, intWindowStyle, bWaitOnReturn)`: Runs an external program.

Properties:

- `WshShell.Environment(strType)`: Returns a collection of environment variables.

File System Operations

FileSystemObject (FSO)

The `FileSystemObject` provides a comprehensive set of methods and properties for working with files and folders.

Creating an instance:

```
Set fso = WScript.CreateObject("Scripting.FileSystemObject")
```

Methods:

- `fso.CopyFile(strSource, strDestination, [bOverwrite])`: Copies one or more files from one location to another.
- `fso.CopyFolder(strSource, strDestination, [bOverwrite])`: Copies one or more folders from one location to another.
- `fso.CreateFolder(strPath)`: Creates a folder.
- `fso.CreateTextFile(strPath, [bOverwrite], [bUnicode])`: Creates a text file.
- `fso.DeleteFile(strPath, [bForce])`: Deletes one or more files.
- `fso.DeleteFolder(strPath, [bForce])`: Deletes one or more folders.
- `fso.FileExists(strPath)`: Returns `True` if a file exists.
- `fso.FolderExists(strPath)`: Returns `True` if a folder exists.
- `fso.GetFile(strPath)`: Returns a `File` object.
- `fso.GetFolder(strPath)`: Returns a `Folder` object.
- `fso.GetParentFolderName(strPath)`: Returns the parent folder name.
- `fso.GetSpecialFolder(intFolder)`: Returns a special folder path (e.g., Windows, System, Temporary).

Properties:

- `fso.Drives`: Returns a `Drives` collection.

File Object

The `File` object represents a single file and provides methods and properties for working with file attributes and contents.

Methods:

- `file.OpenAsTextStream([intIOMode], [intFormat])`: Opens the file as a text stream.
- `file.Delete([bForce])`: Deletes the file.
- `file.Copy(strDestination, [bOverwrite])`: Copies the file to another location.
- `file.Move(strDestination)`: Moves the file to another location.

Properties:

- `file.Attributes`: Gets or sets the file attributes.
- `file.DateCreated`: Returns the date and time the file was created.
- `file.DateLastAccessed`: Returns the date and time the file was last accessed.
- `file.DateLastModified`: Returns the date and time the file was last modified.
- `file.Name`: Gets or sets the name of the file.
- `file.ParentFolder`: Returns the parent folder object.
- `file.Path`: Returns the full path to the file.
- `file.Size`: Returns the size of the file in bytes.
- `file.Type`: Returns the type of the file.

TextStream Object

The `TextStream` object allows you to read from and write to text files.

Methods:

- `textStream.Close()`: Closes the text stream.
- `textStream.Read([intCharacters])`: Reads a specified number of characters from the text stream.
- `textStream.ReadAll()`: Reads the entire text stream.
- `textStream.ReadLine()`: Reads a single line from the text stream.
- `textStream.Skip([intCharacters])`: Skips a specified number of characters in the text stream.
- `textStream.SkipLine()`: Skips the next line in the text stream.
- `textStream.Write(strText)`: Writes a string to the text stream.
- `textStream.WriteLine(strText)`: Writes a string followed by a newline character to the text stream.

Properties:

- `textStream.AtEndOfLine`: Returns `True` if the current position is at the end of a line.
- `textStream.AtEndOfStream`: Returns `True` if the current position is at the end of the stream.
- `textStream.Column`: Returns the current column number.

Networking and System Information

WScript.Network Object

The `WScript.Network` object provides access to network-related information and functionality.

Methods:

- `network.AddWindowsPrinterConnection(strPrinterPath, [strDriverName], [strPort])` : Adds a connection to a network printer.
- `network.MapNetworkDrive(strDrive, strShare, [bPermanent], [strUser], [strPassword])` : Maps a network drive.
- `network.RemoveNetworkDrive(strDrive, [bForce], [bUpdateProfile])` : Removes a mapped network drive.
- `network.RemoveWindowsPrinterConnection(strPrinterPath, [bForce], [bUpdateProfile])` : Removes a printer connection.

Properties:

- `network.ComputerName` : Returns the name of the computer.
- `network.UserDomain` : Returns the domain of the current user.
- `network.UserName` : Returns the name of the current user.

Environment Variables

`WshShell.Environment("System")`	Access system-wide environment variables.
`WshShell.Environment("User")`	Access user-specific environment variables.
`WshShell.Environment("Volatile")`	Access volatile environment variables (exist only for the current session).
`WshShell.Environment("Process")`	Access environment variables specific to the current process.
`WshShell.ExpandEnvironmentStrings("%PATH%")`	Expands environment variables within a string. Example shows accessing the PATH variable.

Registry Access

`WshShell.RegRead(strName)`	Reads a value from the registry. `strName` is the full path to the registry key and value.
`WshShell.RegWrite(strName, varValue, strType)`	Writes a value to the registry. `strName` is the full path, `varValue` is the value to write, and `strType` is the data type (e.g., `REG_SZ` , `REG_DWORD`).
`strType` values	• `REG_SZ` : String value • `REG_DWORD` : 32-bit integer value • `REG_BINARY` : Binary data
Example: Read	`value = WshShell.RegRead("HKCU\\Software\\MyKey\\MyValue")`
Example: Write	`WshShell.RegWrite "HKCU\\Software\\MyKey\\MyValue", "MyString", "REG_SZ"`

Advanced Scripting Techniques

Running External Programs

The `WshShell.Run` method is used to execute external programs or commands. It allows you to control the window style and wait for the program to finish.

`WshShell.Run(strCommand, [intWindowStyle], [bWaitOnReturn])`

- `strCommand` : The command to execute.
- `intWindowStyle` : Optional window style (e.g., `0` for hidden, `1` for normal, `3` for maximized).
- `bWaitOnReturn` : Optional boolean value indicating whether to wait for the program to finish (`True`) or not (`False`).

Example:

```
Set WshShell = CreateObject("WScript.Shell")
ReturnCode = WshShell.Run("notepad.exe", 1, True)
WScript.Echo "Notepad returned: " & ReturnCode
```

Error Handling

Error handling in VBScript can be achieved using the `On Error Resume Next` statement, which allows the script to continue execution even if an error occurs. The `Err` object can be used to retrieve information about the error.

Example:

```
On Error Resume Next

' Code that might cause an error
WScript.Echo 1 / 0

If Err.Number <> 0 Then
    WScript.Echo "Error: " & Err.Description
    Err.Clear
End If
```

- `Err.Number` : Returns the error number.
- `Err.Description` : Returns a description of the error.
- `Err.Clear` : Clears the error object.

Event Logging

The `WshShell.LogEvent` method allows you to log events to the Windows event log.

`WshShell.LogEvent(intType, strEvent)`

- `intType` : The event type (e.g., `0` for success, `1` for error, `2` for warning, `4` for information).
- `strEvent` : The event message.

Example:

```
Set WshShell = CreateObject("WScript.Shell")
WshShell.LogEvent 4, "Script executed successfully."
```