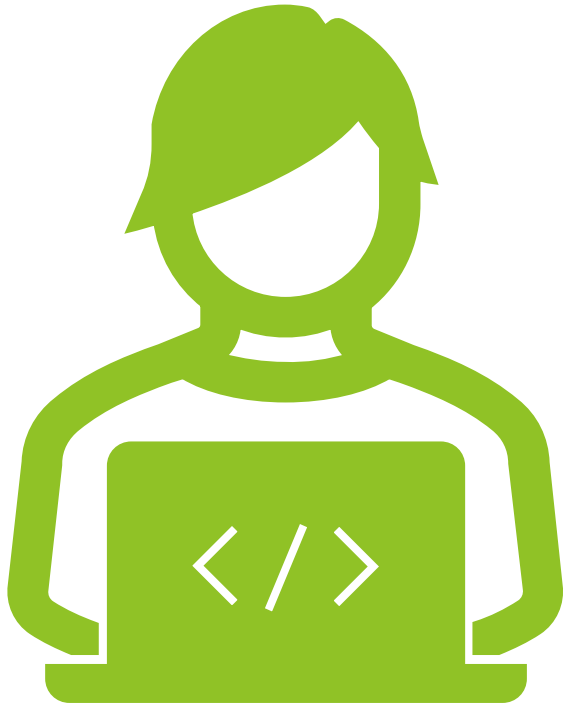


WshModeJs

Use JScript like Node.js

Question: How to operate Windows with JavaScript?

Hint: It's not Node.js!

Answer: It's JScript

Agenda

- Advantages of JScript
- The problem with JScript...
- That's where WshModeJs comes in!
  - some examples of code
  - How to install
  - Coding modernly with WshModeJs
- More features of WshModeJs
- Summary

Advantages of JScript

- No need to install external applications to execute.
- High compatibility, running from Windows 7 to the latest Windows 11.
- It can be used in environments where PowerShell is not available for various reasons.
- You can learn JavaScript!

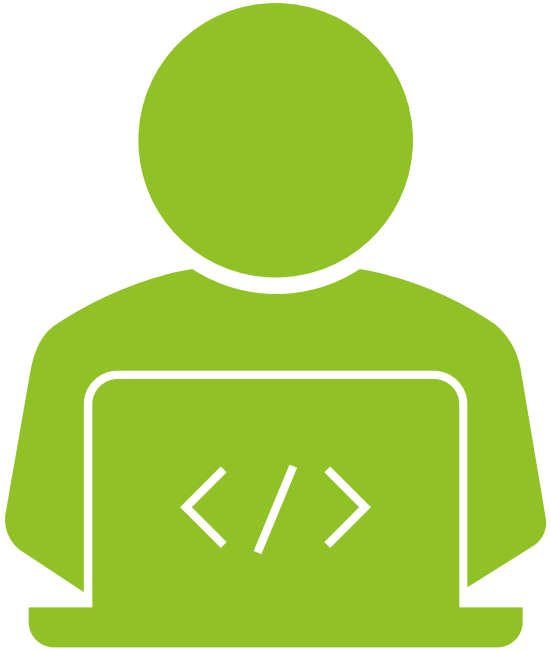
The problem with JScript...

Since it's an old language, even simple processes require effort in programming.

That's where

WshModeJs

comes in!



What is WshModeJs?

WshModeJs was named with the desire to handle JScript in a modern way, like Node.JS.

- Easy introduction
- You can simplify various codes when programming with JScript.
- OSS (MIT license)

Modern coding with WshModeJs

You can write the JScript code you want to achieve in a modern way.

```
var os = Wsh.OS; // Shorthand

// Check the administrative privileges of the execution user.
os.isAdmin(); // false

// Start the program with administrator privileges.
os.runAsAdmin('mklink', 'D:\\Temp-Symlink D:\\Temp', { shell: true });

// Creating, running, and deleting task scheduler
os.Task.create('MyTask', 'wscript.exe', '//job:run my-task.wsf');
os.Task.exists('MyTask'); // true
os.Task.xmlString('MyTask'); // Returns: The task XML string
os.Task.run('MyTask');
os.Task.del('MyTask');

// drive assignment
os.assignDriveLetter('\\\\MyNAS\\MultiMedia', 'M', 'myname', 'mY-p@ss');
// Return 'M'

// Get process information
var pIDs = os.getProcessIDs('Chrome.exe');
// Returns: [33221, 22044, 43113, 42292, 17412]
var pIDs = os.getProcessIDs('C:\\Program Files\\Git\\bin\\git.exe');
// Returns: [1732, 4316]

// Delete user
os.addUser('MyUserName', 'mY-P@ss');
os.attachAdminAuthorityToUser('MyUserName');
os.deleteUser('MyUserName');

// etc...
```

Introduction is very simple!

- Just copy WshModeJs to the same location as the JScript file and create a .wsf file.

https://github.com/tuckn/WshModeJs/tree/master?tab=readme-ov-file#installation

Introducing some modern code using WshModeJs

```
var fs = Wsh.FileSystem; // Shorthand

// Creates and removes a directory
fs.mkdirSync('D:\\MyDir');
fs.rmdirSync('D:\\MyDir');

// Copies and removes a file
fs.copyFileSync('D:\\SrcFile.path', 'R:\\DestFile.path');
fs.unlinkSync('D:\\MyFile.path');

// Copies a directory with XCOPY
fs.xcopySync('D:\\SrcDir', 'R:\\DestDir');

// Creates a symbolic-link
fs.linkSync('D:\\MyDir\\BackUp', 'C:\\BackUp-Symlink'); // Requires admin

// Gets information about the file
var stat = fs.statSync('D:\\My Dir\\File.path');
stat.isFile(); // true
stat.isDirectory(); // false
stat.isSymbolicLink(); // false

// Reads the contents of the directory
fs.readdirSync('D:\\testDir');
// Returns: [
//   'fileRoot1.txt',
//   'fileRoot2-Symlink.log', // <SYMLINKD>
//   'fileRoot2.log',
//   'DirBar',
//   'DirBar-Symlink', // <SYMLINKD>
//   'DirFoo' ]
```

File operations like Node.js

You can now manipulate files using a description similar to Node.js‘ FileSystem.

[Wsh.FileSystem]

https://tuckn.net/docs/WshFileSystem/

Easy to read and write files

- It is also possible to specify the character code when saving.
- You can also create a temporary file in the `%TEMP%` folder with a short code.

```
var fs = Wsh.FileSystem; // Shorthand

fs.readdirSync('D:\\testDir', { withFileTypes: true });
// Returns: [
//   { name: 'fileRoot1.txt',
//     path: 'D:\\testDir\\fileRoot1.txt',
//     attributes: 32,
//     isDirectory: false,
//     isFile: true,
//     isSymbolicLink: false },
//   ...
//   ..
//   { name: 'DirFoo.txt',
//     path: 'D:\\testDir\\DirFoo',
//     attributes: 16,
//     isDirectory: true,
//     isFile: false,
//     isSymbolicLink: false }]

// Writes a data to the file
fs.writeFileSync('D:\\my-note.txt', 'My note.', { encoding: 'utf8' });
fs.writeFileSync('D:\\MyNote.txt', 'My note.', { encoding: 'sjis' });
fs.writeFileSync('D:\\my-script.wsf', 'WScript.Echo("Foo");', {
  encoding: 'utf8',
  bom: true,
});

// Reads the file
var readText = fs.readFileSync('D:\\MyNote.txt', { encoding: 'sjis' });

// Create a temporary file
var tmpPath = fs.writeTmpFileSync('My Temp', { encoding: 'utf8' });
// Returns: 'C:\\Users\\UserName\\AppData\\Local\\Temp\\fs-writeTmpFileSync_rad6E884.tmp'
```

You can even handle JSON and CSV!

- In addition to specifying character codes, you can also specify line feed codes and indent strings.
- It is also possible to obtain and compare file hash values in one line is also possible.

[FileSystemExtra]

https://tuckn.net/docs/WshFileSystem/Wsh.FileSystemExtra.html

```
var fse = Wsh.FileSystemExtra; // Shorthand

// JSON
var testObj = {
  array: [1, 2, 3],
  bool: false,
  num: 42,
  obj: { a: 'A' },
  str: 'Some string',
};

// Writes
fse.writeJsonSync('D:\\test_sjis.json', testObj, {
  indent: '  ',
  lineEnding: '\r\n',
  encoding: 'sjis',
});
// Reads
var readObj = fse.readJsonSync('D:\\settings.json');

// CSV

var testArray = [
  ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'],
  ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10'],
  [
    '2020/1/1',
    "'007",
    'Has Space',
    '日本語',
    'I say "Yes!"',
    'Line\nBreak',
    'Foo,Bar,Baz',
  ],
];

// Writes
fse.writeCsvSync('D:\\test.csv', testArray);
// Reads
var readArray = fse.readCsvSync('D:\\logs.csv', { encoding: 'utf8' });
```

```
var os = Wsh.OS; // Shorthand

os.is64arch(); // true
os.tmpdir(); // 'C:\\Users\\YourUserName\\AppData\\Local\\Temp'
os.makeTmpPath(); // 'C:\\Users\\UserName\\AppData\\Local\\Temp\\rad6E884.tmp'
os.homedir(); // 'C:\\Users\\%UserName%'
os.hostname(); // 'MYPC0123'
os.cmdCodeset(); // 'shift_jis'

// write to event log
os.writeLogEvent.error('Error Log');
// Logs the error event in Windows Event Log.

os.writeLogEvent.info('Information Log');
os.writeLogEvent.success('Success Log');
os.writeLogEvent.warn('Warn Log');

console.log(process.cwd); // C:\tuckn\test
console.log(process.execPath); // C:\Windows\system32\cscript.exe
console.dir(process.execArgv); // ["//nologo", "//job:run"]
console.dir(process.argv);
// Outputs: [
//   "C:\Windows\system32\cscript.exe",
//   "D:\Run.wsf",
//   "-n",
//   "arg 1"]

console.dir(process.env);
// Outputs: {
//   ALLUSERSPROFILE: "C:\ProgramData",
//   APPDATA: "C:\Users\UserName\AppData\Roaming",
//   CommonProgramFiles: "C:\Program Files\Common Files",
//   CommonProgramFiles(x86): "C:\Program Files (x86)\Common Files",
//   CommonProgramW6432: "C:\Program Files\Common Files",
//   COMPUTERNAME: "MYPC0123",
//   ComSpec: "C:\WINDOWS\system32\cmd.exe",
//   HOMEDRIVE: "C:",
//   HOMEPATH: "\Users\UserName",
//   ... }
```

Of course, OS and processes can be handled easily.

I forgot to mention, but of course, you can also use the console command!

[Wsh.OS]

https://tuckn.net/docs/WshOS/

[Wsh.Process]

https://tuckn.net/docs/WshProcess/

There are also many useful functions like Lodash

- I also have many useful functions that will speed up your programming!

[WshUtil]

https://tuckn.net/docs/WshUtil/

[Lodash]

https://lodash.com/

```
var _ = Wsh.Util; // Shorthand

// Checks deep strict equality
_.isEqual([1, 2, 3], [1, 2, 3]); // true
_.isEqual([1, 2, 3], [1, 2]); // false
_.isEqual({ a: 'A', b: ['B'] }, { a: 'A', b: ['B'] }); // true
_.isEqual({ a: 'A', b: ['B'] }, { a: 'A', b: ['b'] }); // false

// Checks if a value is an empty enumerable object or non enumerable
_.isEmpty([]); // true
_.isEmpty([1]); // false
_.isEmpty({}); // true
_.isEmpty({ a: 'A' }); // false
_.isEmpty(''); // true
_.isEmpty('a'); // false
_.isEmpty('3'); // false
_.isEmpty(3); // true
_.isEmpty(undefined); // true - Because non enumerable object
_.isEmpty(null); // true
_.isEmpty(true); // true

// Gets a value from a object
var obj = { a: 1, b: { B: 2 }, c: [3, 4] };
_.get(obj, 'a'); // 1
_.get(obj, 'Z'); // undefined
_.get(obj, 'Z', 'defVal'); // 'defVal'
_.get(obj, 'b.B'); // 2
_.get(obj, ['b', 'B']); // 2
_.get(obj, 'c.1'); // 4

// Creates a unique ID
_.uuidv4(); // '9f1e53ba-3f08-4c9d-91c7-ad4226312f40'

// Creates a date string
_.createDateString(); // '20200528T065424+0900'
_.createDateString('yyyy-MM'); // '2020-05'

// Parses the date template literal to a date string.
_.parseDateLiteral('#{yyyy-MM-ddTHH:mm:ss}'); // '2020-01-02T15:04:05'
_.parseDateLiteral('#{yyyy/M/d H:m:s}'); // '2020/1/2 15:4:5'
_.parseDateLiteral('C:\\MyData\\#{yyyy-MM-dd}.txt'); // 'C:\MyData\2020-01-02.txt'
_.parseDateLiteral('\\\\MyNas\\#{yyyy}\\#{MM}\\#{dd}'); // '\\MyNas\2020\01\02'
```

Easily create custom commands with the CLI framework

- You can easily create your own commands using JScript.
- The CLI framework provides functionality similar to Click and argparse in Python and Thor in Ruby.

[WshCommander]

https://tuckn.net/docs/WshCommander/

```
var cmd = Wsh.Commander; // Shorthand

cmd.addProgram({
 /* The program schema A */
});

cmd.addProgram({
 /* The program schema B */
});

cmd.parse(/* WSH Arguments */);
```

For example.

```
var cmd = Wsh.Commander; // Shorthand

cmd.addProgram({
  command: 'play <consoleName> [gameTitle]',
  options: [
    ['-S, --speed [LV]', 'The game speed (Default: 5)', 5]
  ]
  action: function (consoleName, gameTitle, options) {
    if (typeof gameTitle === 'string') {
      console.log('play ' + gameTitle + ' on ' + consoleName);
    } else {
      console.log('play ' + consoleName);
    }
  }
});

cmd.parse(process.argv);
```

```
> cscript .\Run.wsf play "SEGA Saturn" "StreetFighter ZERO"
play StreetFighter ZERO on SEGA Saturn
```

More features available!

Read environment variables, read .env and JSON files as configuration values, and many other features.

- WshPolyfill
- WshUtil
- WshPath
- WshOS
- WshFileSystem
- WshProcess
- WshChildProcess
- WshNet
- WshModeJs
- WshCommander
- WshConfigStore
- WshDotEnv
- WshLogger

Summary of WshModeJs

- It is very easy to install and makes programming with JScript easy.
- It has many functions implemented that we can't introduce yet!

All modules are implemented 👇

[WshBasicPackage]

https://tuckn.net/docs/WshBasicPackage/

Please check out our other features.

Enjoy your wonderful JScript life

Thank you for watching until the end.

https://github.com/tuckn