

# From Cmdlet to Function – Your PowerShell Beginnings

Mike Nelson

Level: Beginner



# Mike Nelson

## Solutions Technologist @ Rubrik

Microsoft MVP, Azure Advisor, VMware vExpert, Citrix CTA

@nelmedia

mike@mikenelson.io



TechMentor Redmond 2019

**TECHMENTOR**  
IN-DEPTH TRAINING FOR IT PROS

# Get-Content

- PowerShell Basics
- Functions & More
- The Demo
  - Creating a script
  - Your first Function
  - Morphing into a Module

# HOW IT STARTED

Jeffrey Snover, Bruce Payette, &  
James Truher

Project Monad in 2002





# Snover Quotes

"This is rock science, not rocket science."

"I took a demotion to create PowerShell."

"PowerShell is a such a great product because I am a deeply flawed human being."

"Not updating from WS2003 is like the guy who jumps off a building and on the way down says, 'so far, so good!'"

"When in trouble, fear, or doubt - run in circles, scream, and shout."



# VERSIONS



5.1



~~Core~~

PowerShell 7



5.1

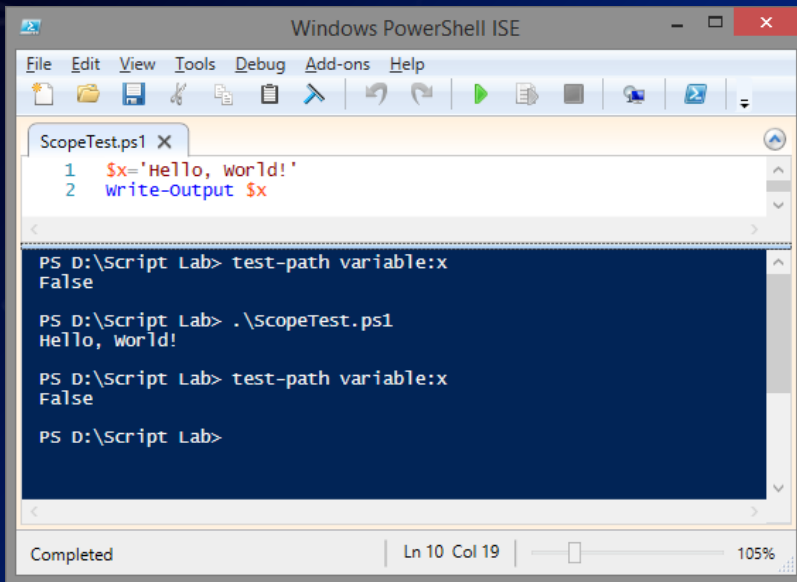
```
PS C:\Windows\system32> $M=Get-Command * | Measure
PS C:\Windows\system32> $M.Count
2526
PS C:\Windows\system32> $M=Get-Module -ListAvailable | Measure
PS C:\Windows\system32> $M.Count
79
```

6.2.2

```
PS C:\Program Files\PowerShell\6> $M=Get-Command * | Measure
PS C:\Program Files\PowerShell\6> $M.Count
2606
PS C:\Program Files\PowerShell\6> $M=Get-Module -ListAvailable | Measure
PS C:\Program Files\PowerShell\6> $M.Count
66
```

7.0p2

```
PS C:\Program Files\PowerShell\7-preview> $M=Get-Command * | Measure
PS C:\Program Files\PowerShell\7-preview> $M.Count
2542
PS C:\Program Files\PowerShell\7-preview> $M=Get-Module -ListAvailable | Measure
PS C:\Program Files\PowerShell\7-preview> $M.Count
66
```



The screenshot shows the Windows PowerShell ISE interface. The menu bar includes File, Edit, View, Tools, Debug, Add-ons, and Help. The toolbar contains icons for file operations and execution. A script file named 'ScopeTest.ps1' is open, containing two lines of code: `$x='Hello, world!'` and `write-Output $x`. The console window shows the following commands and output: `PS D:\Script Lab> test-path variable:x` returns `False`; `PS D:\Script Lab> .\ScopeTest.ps1` outputs `Hello, world!`; `PS D:\Script Lab> test-path variable:x` returns `False`; and `PS D:\Script Lab>` is the prompt. The status bar at the bottom indicates 'Completed', 'Ln 10 Col 19', and '105%' zoom.

```
File Edit View Tools Debug Add-ons Help
ScopeTest.ps1 X
1 $x='Hello, world!'
2 write-Output $x

PS D:\Script Lab> test-path variable:x
False

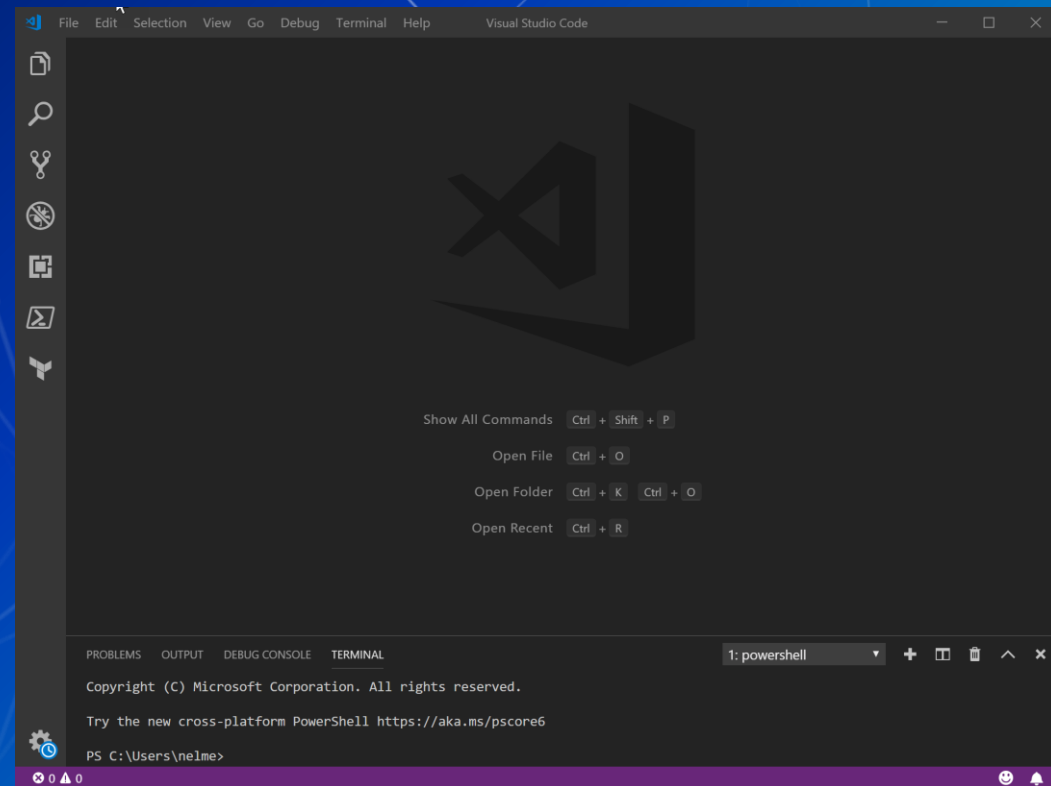
PS D:\Script Lab> .\ScopeTest.ps1
Hello, world!

PS D:\Script Lab> test-path variable:x
False

PS D:\Script Lab>

Completed | Ln 10 Col 19 | 105%
```

ISE



The screenshot shows the Visual Studio Code interface. The menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The main editor area is dark and displays the Visual Studio Code logo. Below the logo, there are keyboard shortcuts for 'Show All Commands' (Ctrl + Shift + P), 'Open File' (Ctrl + O), 'Open Folder' (Ctrl + K, Ctrl + O), and 'Open Recent' (Ctrl + R). The bottom status bar shows '1: powershell', a dropdown menu, and icons for adding, removing, and refreshing the terminal. The terminal window at the bottom displays the following text: 'Copyright (C) Microsoft Corporation. All rights reserved.', 'Try the new cross-platform PowerShell https://aka.ms/pscore6', and the prompt `PS C:\Users\nelme>`.

```
File Edit Selection View Go Debug Terminal Help Visual Studio Code

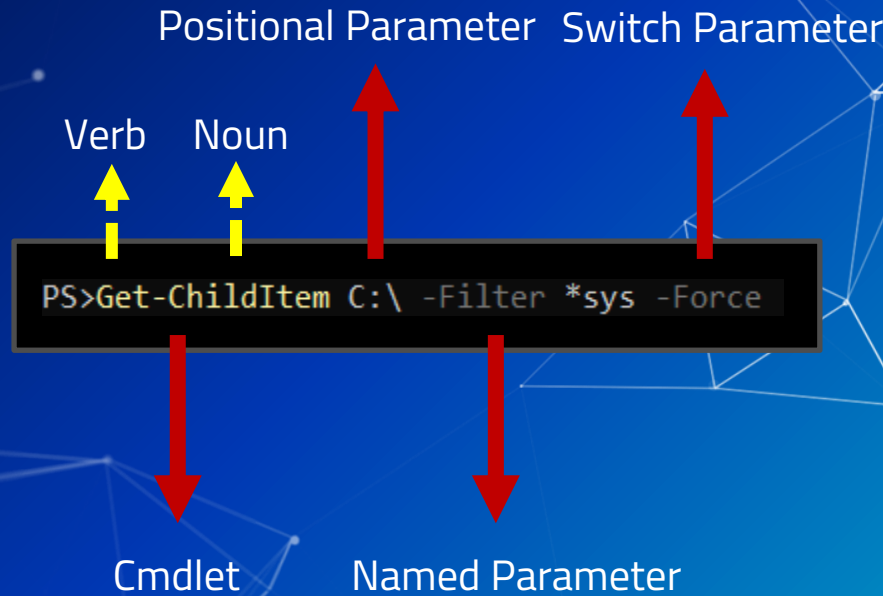
Show All Commands Ctrl + Shift + P
Open File Ctrl + O
Open Folder Ctrl + K Ctrl + O
Open Recent Ctrl + R

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\nelme>
```

VSCode



# SYNTAX



# CORE COMMANDS TO HELP LEARN

Get-Command

Get-Help

Update-Help

Get-Member

Show-Command

# LET'S MAKE IT EASY



PowerShell Gallery

Easy find, install, & update of Modules

Chocolatey

Easy updating of Core (and more)

Github search

Always give credit



“Create documented and  
reusable code, you will.”

*- PowerShell Yoda*





# FUNCTIONS

A list of statements you define

```
function helloworld {  
    Write-Host 'Hello World!'  
}
```

- You then simply call the function

- Statements then just run like you typed them

Functions can be in scripts, modules, profiles, and "called" from scripts

# MODULES

A package of commands in a .psm1 file

■ *PS>New-Module* is one way

■ 3 kinds

■ Script – **psm1** files that contain any valid PowerShell code

■ Binary – compiled DLL's

■ Dynamic – only available in memory

■ Manifest (optional) – **.psd1** file that describes a module, syntax & requirements, & how it is to be processed

*PS>New-ModuleManifest*

*PS>Test-ModuleManifest*

When would you / should you create a module?

# PIPELINES

Connecting commands together

- PS>Get-Process | Out-Host -Verbose
- Reduce the effort of writing long, complex commands
- Can utilize pipeline variables ( \$\_ or \$PSItem )
- Commonly used in "one-liners"

# Let's Create a Module!





# What are we going to do?

- Start by seeing some basics
- Create a script that runs an alarm clock
- Turn that script into a function
- Take that function & create a module

# THANKS!

**Any questions?**

You can find me at

■ @nelmedia

■ mike@mikenelson.io

