FROM CMDLET TO FUNCTION – YOUR POWERSHELL BEGINNINGS

Mike Nelson

Code Palousa 2022

Thank You to the Code PaLOUsa Sponsors

Progress

WAYSTAR

Bastian SOLUTIONS
a TOYOTA ADVANCED LOGISTICS company

<code> LOUISVILLE

eblu SOLUTIONS

twilio

Scout

aws

cbts

<prosoft>

CGI

Friends of Code PaLOUsa

Asperitas

Couchbase

DATASTAX

mongoDB

redis

ROCKET Mortgage

Get-Mike

mikenelsonio

nelmedia

mikenelson-io

- 35+ years in tech
- Principal Technologist @ Cohesity
- Experience from Helpdesk to Architect
- Scripter, not a coder
- Passion for community, teaching, learning
- Beer, BBQ, & Gadgets

Microsoft MVP Most Valuable Professional

Citrix Technology Advocate CTA

vmware vEXPERT

/MYPRESENTATIONS/2022-AUGUST_CODE PALOUSA

GET-AGENDA

- Start-PowerShell (history)
- Show-PowerShell (the basics)
- Use-PowerShell (demos)

POWERSHELL (POSH)

Started as a scripting framework for automation & evolved into a command line interface (CLI) and a scripting language.

The default CLI automation standard for Microsoft products & their ecosystems.

Open source.

WHY USE POWERSHELL?

Those who don’t automate are doomed to repeat themselves.

GET-HISTORY

- Jeffrey Snover, Bruce Payette, & James Truher
- Project Monad in 2002
- The “Monad Manifesto”
- Jeffrey was demoted after creating Monad
- R.I.P. Jeffrey + Microsoft July 31, 2022

GET-VERSION

<=5.1

Windows

.Net Framework

powershell.exe

~~6.x~~

7.x

Windows-Linux-MacOS

.NET Core

pwsh.exe

-UseWindowsPowerShell

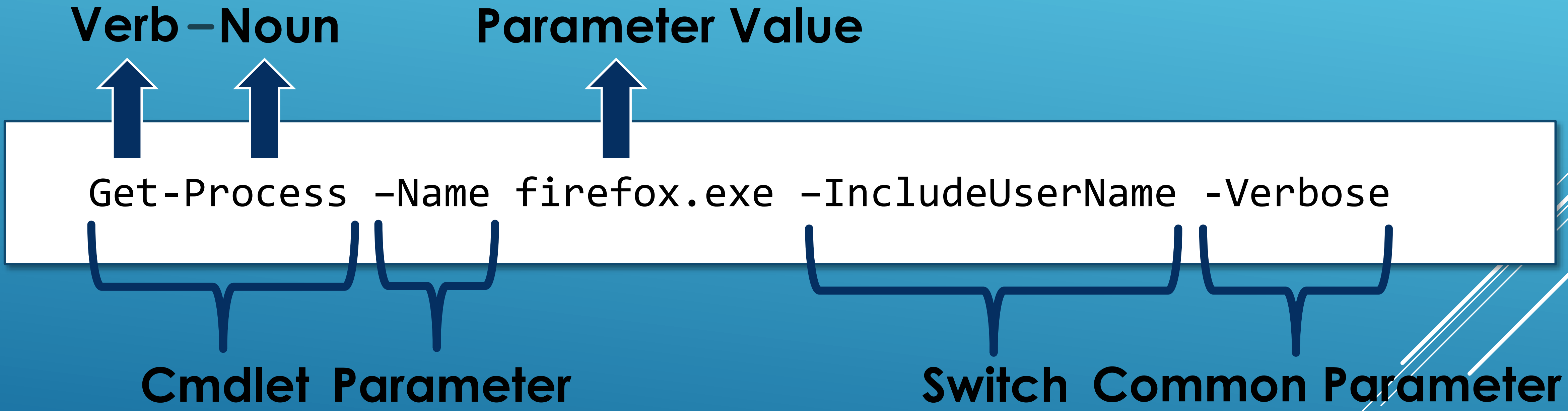
SHOW-PROFILE

- The PowerShell profile is a script that runs when a PowerShell session is started (unless the –noprofile switch is used)
- Basically, it is a logon script for PowerShell containing commands, aliases, variables, drives, functions, modules, etc.
- Profiles can be for all users, the current user, all hosts, and the current host. You can have a mix of none, some, or all of these and there is a precedence order.
- There is no default profile
- Your current user profile is stored in the $profile variable. To edit your current user profile with VSCode, type code $profile at a PowerShell prompt.

SHOW-VARIABLE

- A unit of memory in which a value is stored
- PowerShell variables are text strings represented by the dollar sign "$"prefix (ex. $a, $my_var, $var1, etc.)
- Although special characters and spaces allowed, variable names should be kept simple
- Types of variables:
  - User – user defined and deleted on exit (add to your PowerShell Profile to sustain)
  - Automatic – defined by Posh & not editable (ex. $PSHOME)
  - Preference – defaults defined & are user editable
- Get-Variable to show all variables defined in a session

GET-SYNTAX



```
Get-Process –Name firefox.exe –IncludeUserName -Verbose
```

CMDLETS (COMMAND-LETS)

- A type of command in PowerShell
- Common syntax & options
- Usually take object input & return objects
- Run standalone, combined, or stored in .ps1 file as scripts

SHOW-PARAMETER

- Allow for users to provide input or options
- A pre-hyphen (“-”) is not always necessary (ie. positional)
- Some parameters have default values (dev decision)
- Different Types:
  - Named -> default full name of parameter
  - Positional -> typed in a relative order (caution)
  - Dynamic -> only available under special conditions
  - Common -> built-in parameters
  - Sets -> expose different parameters & return different information

SHOW-PIPELINING

Pipeline operator

```
PS>Get-Process –Name firefox.exe –IncludeUserName | Stop-Process
```

Object returned by first cmdlet sent to second cmdlet

“One-liner”

SHOW-FUNCTION

A list of PowerShell statements that run like you had entered them on the command line.

```
function Get-FirefoxProcess { Get-Process firefox.exe }
function Get-FirefoxProcess {
  $a = Get-Process firefox.exe
    if ($a –eq $null) {
      Write-Host “No Firefox process present”}
    return $a
}
```

To run a function, you call it by name on the command line, in a script, or in a module.

```
function Get-FirefoxProcess {
  $a = Get-Process firefox.exe
    if ($a –eq $null) {
      Write-Host “No Firefox process present”}
    return $a
}
Get-FirefoxProcess
```

SHOW-MODULE

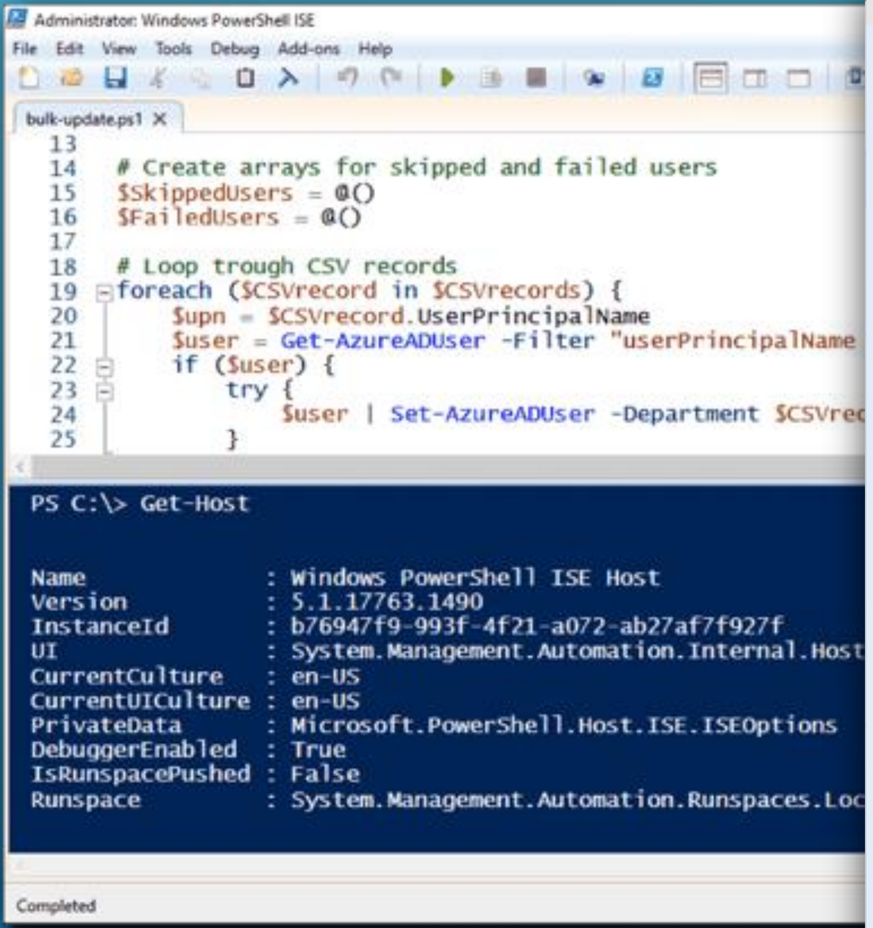
- Modules are a .psm1 file or a .dll that contain commands, providers, variables, functions, help context, aliases, & workflows
- Imported as itself (automatically via autoloading or manually via the Import-Module cmdlet) or launched from .psd1 manifest file (see next slide)
- Can be stored anywhere, but common paths is best (ex. $env:PSModulePath)
- Easy lifecycle management with Install-Module & Update-Module

SHOW-MODULEMANIFEST

- Module can be defined by a manifest file which has an extension of .psd1
- A manifest is not required for a module.
- Manifests are made up of a hash table of keys and values that describe the contents & attributes of a module, define prerequisites, and determine how components are processed.
- Think of it as a glossary and instruction set for a module for PowerShell.

SET-EDITOR

ISE (<=v5.1)



Administrator: Windows PowerShell ISE

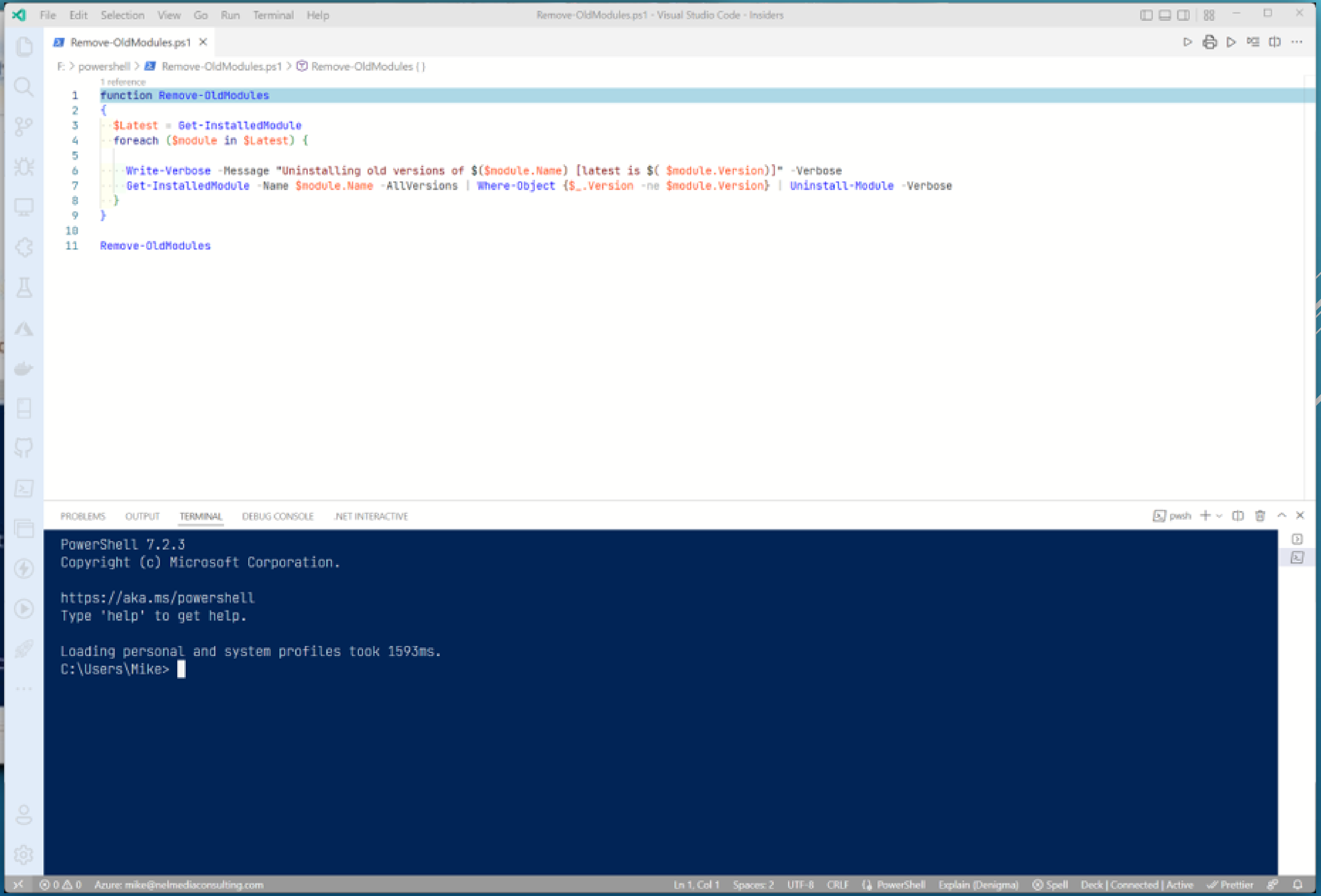
bulk-update.ps1

```
    # Create arrays for skipped and failed users
    $SkippedUsers = @()
    $FailedUsers = @()

    # Loop trough CSV records
    foreach ($CSVrecord in $CSVrecords) {
        $upn = $CSVrecord.UserPrincipalName
        $user = Get-AzureADUser -Filter "userPrincipalName
        if ($user) {
            try {
                $user | Set-AzureADUser -Department $CSVrec
            }
```

```
PS C:\> Get-Host


Name             : Windows PowerShell ISE Host
Version          : 5.1.17763.1490
InstanceId       : b76947f9-993f-4f21-a072-ab27af7f927f
UI               : System.Management.Automation.Internal.Host
CurrentCulture   : en-US
CurrentUICulture : en-US
PrivateData      : Microsoft.PowerShell.Host.ISE.ISEOptions
DebuggerEnabled  : True
IsRunspacePushed : False
Runspace         : System.Management.Automation.Runspaces.Loc
```

Completed

VSCode



Remove-OldModules.ps1 - Visual Studio Code - Insiders

F: > powershell > Remove-OldModules.ps1 > Remove-OldModules { }

```
function Remove-OldModules
{
  $Latest = Get-InstalledModule
  foreach ($module in $Latest) {

    Write-Verbose -Message "Uninstalling old versions of $($module.Name) [latest is $( $module.Version)]" -Verbose
    Get-InstalledModule -Name $module.Name -AllVersions | Where-Object {$_.Version -ne $module.Version} | Uninstall-Module -Verbose
  }
}

Remove-OldModules
```

```
PowerShell 7.2.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

Loading personal and system profiles took 1593ms.
C:\Users\Mike>
```

Azure: mike@nelmediaconsulting.com   Ln 1, Col 1   Spaces: 2   UTF-8   CRLF   PowerShell

CORE CMDLETS TO KNOW

- Get-Help
- Update-Help
- Get-Command
- Show-Command
- Get-Member
- Update-Module

DEMO STUFF

THANK YOU!

@mikenelsonio

Github - mikenelson-io

LinkedIn - nelmedia