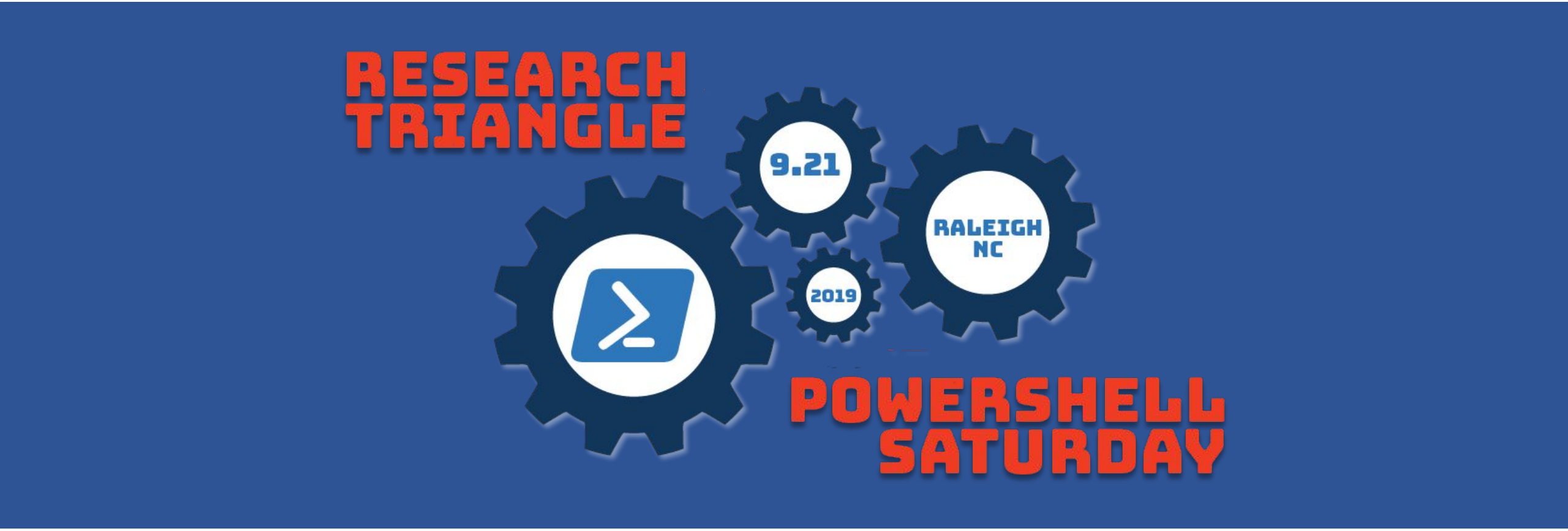
Streams and the Write Cmdlets

Justin Gehman



Sponsored by:

PLURALSIGHT

ScriptRunner®

A Story on Reusability

Packaging designed for reuse.

RESEARCH TRIANGLE 9.21 RALEIGH NC 2019 POWERSHELL SATURDAY

A Story on Reusability

Packaging can accept other, similar products.

RESEARCH TRIANGLE 9.21 RALEIGH NC 2019 POWERSHELL SATURDAY

A Story on Reusability

Packaging can accept different but still pretty similar products.

RESEARCH TRIANGLE 9.21 RALEIGH NC 2019 POWERSHELL SATURDAY

A Story on Reusability

Packaging that doesn't care about the context, it just does it's one task really well.

RESEARCH TRIANGLE 9.21 RALEIGH NC 2019 POWERSHELL SATURDAY

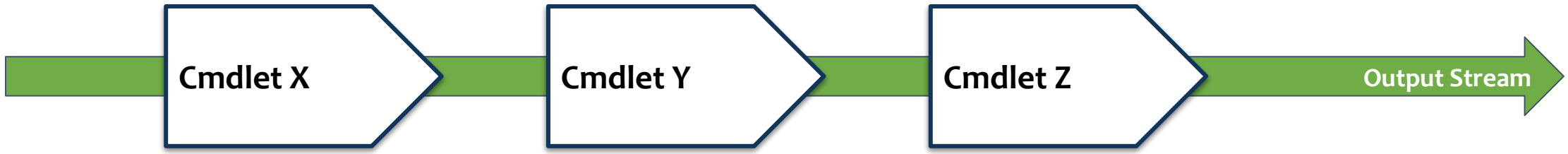
A Story on Reusability

Writing code for maximum reuse makes life easy and so much more fun.

RESEARCH TRIANGLE 9.21 RALEIGH NC 2019 POWERSHELL SATURDAY

Reusability, Output, and the Pipeline

Each cmdlet accepts and outputs objects on the output stream



PowerShell 2.0 Streams

Number	Name	Cmdlet	Introduced
1	Success	Write-Output	PowerShell 2.0
2	Error	Write-Error	PowerShell 2.0

Reusability, Output, and the Pipeline


```
Windows PowerShell
PS C:\> Get-Item *, 'Fake Item' | Sort-Object | Out-GridView
Get-Item : Cannot find path 'C:\Fake Item' because it does not exist.
At line:1 char:1
+ Get-Item *, 'Fake Item' | Sort-Object | Out-GridView
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\Fake Item:String) [Get-Item], ItemNotFou
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetItemCommand

PS C:\>
```

Get-Item *, 'Fake Item' | Sort-Object | Out-GridView

Filter

Add criteria

Mode	LastWriteTime	Length	Name
d-----	3/19/2019 12:52:43 AM		PerfLogs
d-r---	9/14/2019 5:44:26 PM		Program Files
d-r---	9/14/2019 5:03:23 PM		Program Files (x86
d-r---	8/30/2019 1:24:45 AM		Users
d-----	9/14/2019 5:44:33 PM		Windows

RESEARCH TRIANGLE
9.21
RALEIGH NC
2019
POWERSHELL SATURDAY

PowerShell 3.0 Streams

Number	Name	Cmdlet	Introduced
1	Success	Write-Output	PowerShell 2.0
2	Error	Write-Error	PowerShell 2.0
3	Warning	Write-Warning	PowerShell 3.0
4	Verbose	Write-Verbose	PowerShell 3.0
5	Debug	Write-Debug	PowerShell 3.0

Reusability, Output, and the Pipeline

Windows PowerShell

```
PS C:\> Find-Module PS* -Verbose | Sort-Object Name | Out-GridView
VERBOSE: Repository details, Name = 'PSGallery', Location = 'https://www.powershellgaller
VERBOSE: Repository details, Name = 'PSGallery', Location = 'https://www.powershellgaller
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
VERBOSE: The -Repository parameter was not specified.  PowerShellGet will use all of the
VERBOSE: Getting the provider object for the PackageManagement Provider 'NuGet'.
VERBOSE: The specified Location is 'https://www.powershellgallery.com/api/v2' and Package
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById(
VERBOSE: Total package yield:'0' for the specified package 'PS*'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/' for 'PS*'.
VERBOSE: Total package yield:'532' for the specified package 'PS*'.
PS C:\>
```

Find-Module PS* -Verbose | Sort-Object Name | Out-(

Filter

Add criteria

Version	Name	Repository	Descri
1.0.1	PS.B2	PSGallery	A Pow
1.1.3	PS.BitBucket	PSGallery	Bitbuc
0.0.2	ps.checkModuleUpdates	PSGallery	Power
1.0.19	PS.HealthChecks	PSGallery	A Pow
0.0.5	ps.slack	PSGallery	Power
0.0.1.5	PS1C	PSGallery	Power
0.1.0	PS1EAE	PSGallery	Wrapr

RESEARCH TRIANGLE 9.21 RALEIGH NC 2019 POWERSHELL SATURDAY

PowerShell 5.0+ Streams

Number	Name	Cmdlet	Introduced
1	Success	Write-Output	PowerShell 2.0
2	Error	Write-Error	PowerShell 2.0
3	Warning	Write-Warning	PowerShell 3.0
4	Verbose	Write-Verbose	PowerShell 3.0
5	Debug	Write-Debug	PowerShell 3.0
6	Information	Write-Information	PowerShell 5.0

Testing all the Write Cmdlets

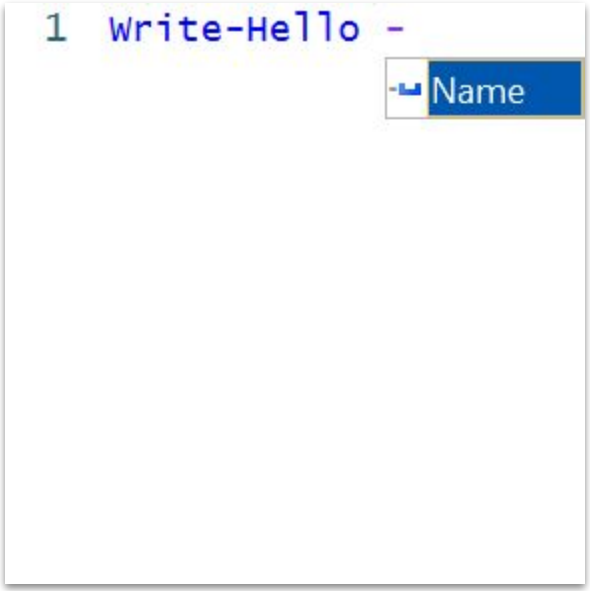
```powershell
function Write-Hello ([string[]]$Name = $env:USERNAME) {
    foreach ($item in $Name) {
        #Streams cmdlets
        Write-Debug -Message "WRITE-DEBUG: Hello, $item. Welcome to PSSaturday."
        Write-Error -Message "WRITE-ERROR: Hello, $item. Welcome to PSSaturday."
        Write-Information -MessageData "WRITE-INFORMATION: Hello, $item. Welcome to PSSaturday."
        Write-Output "WRITE-OUTPUT: Hello, $item. Welcome to PSSaturday."
        Write-Verbose -Message "WRITE-VERBOSE: Hello, $item. Welcome to PSSaturday."
        Write-Warning -Message "WRITE-WARNING: Hello, $item. Welcome to PSSaturday."

        #Non-streams cmdlets
        Write-Progress -Activity "WRITE-Progress: Hello, $item. Welcome to PSSaturday."
        Write-Host "WRITE-HOST: Hello, $item. Welcome to PSSaturday."

        "Hello, $item. This is the actual output."
    }
}
```

Testing all the Write Cmdlets (Again)

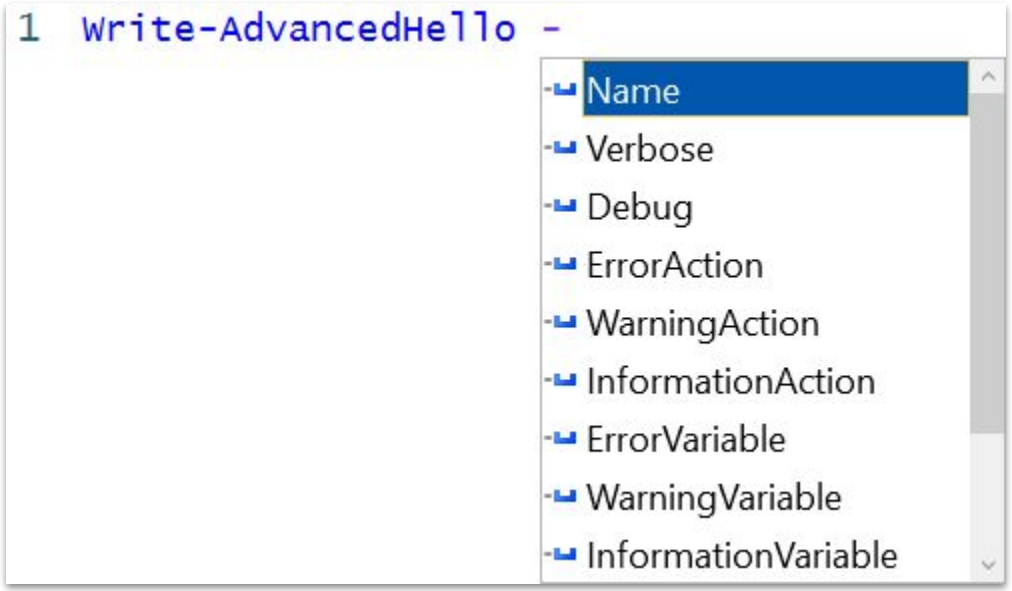
```powershell
function Write-AdvancedHello {
    [CmdletBinding()]
    Param ([string[]]$Name = $env:USERNAME)
    foreach ($item in $Name) {
        #Streams cmdlets
        Write-Debug -Message "WRITE-DEBUG: Hello, $item. Welcome to PSSaturday."
        Write-Error -Message "WRITE-ERROR: Hello, $item. Welcome to PSSaturday."
        Write-Information -MessageData "WRITE-INFORMATION: Hello, $item. Welcome to PSSaturday."
        Write-Output "WRITE-OUTPUT: Hello, $item. Welcome to PSSaturday."
        Write-Verbose -Message "WRITE-VERBOSE: Hello, $item. Welcome to PSSaturday."
        Write-Warning -Message "WRITE-WARNING: Hello, $item. Welcome to PSSaturday."

        #Non-streams cmdlets
        Write-Progress -Activity "WRITE-Progress: Hello, $item. Welcome to PSSaturday."
        Write-Host "WRITE-HOST: Hello, $item. Welcome to PSSaturday."

        "Hello, $item. This is the actual output."
    }
}
```

Always use “[CmdletBinding()]”

Without “[CmdletBinding()]”



With “[CmdletBinding()]”



RESEARCH TRIANGLE 9.21 RALEIGH NC 2019 POWERSHELL SATURDAY

Preference Variables

Cmdlet	Preference Variable	Available Values (Default)
Write-Output		
Write-Error	$ErrorActionPreference	Stop, Inquire, **(Continue)**, Suspend, SilentlyContinue
Write-Warning	$WarningPreference	Stop, Inquire, **(Continue)**, SilentlyContinue
Write-Verbose	$VerbosePreference	Stop, Inquire, Continue, **(SilentlyContinue)**
Write-Debug	$DebugPreference	Stop, Inquire, Continue, **(SilentlyContinue)**
Write-Information	$InformationActionPreference	Stop, Inquire, Continue, Suspend, **(SilentlyContinue)**

Write-Information vs Write-Host

	Write-Host (v5.0+)	Write-Information
Accepts Pipeline Input	✔	⛔
Writes to Information Stream	✔	✔
Supports message tagging	⛔	✔
Supports *InformationAction* and *InformationVariable*	🔺	✔
Supports foreground and background colors	✔	⛔

🔺 Yes, but *InformationAction* not affected by 'Continue', and 'SilentlyContinue' for backwards compatibility.

Related Links

- PowerShell Streams and using the right Write-* Cmdlet

- Write-Debug, Write-Error, Write-Host, Write-Information, Write-Output, Write-Progress, Write-Verbose

- about_CommonParameters
- about_Functions_Advanced
- about_Preference_Variables
- about_Redirection

- Understanding Streams, Redirection, and Write-Host in PowerShell
- Welcome to the PowerShell Information Stream