PowerShell Toolmaking Patterns and Practices

Unleashing the Power of Scripting Across Domains

Speakers

Jeff Hicks

PowerShell MVP | Author | Teacher

https://jdhitsolutions.github.io

Jordan Benzing

Microsoft MVP, Engineer

https://jordantheitguy.com/

Can’t see our slides? Can’t hear? Need to repeat the question? Call us out!

Expectations

- This is ***not*** a session to teach you PowerShell scripting
- Learn how to better apply what you already know
- Identify gaps in your knowledge and experience
- Answer your questions
- Demos galore

Why Does This Matter?

PowerShell is a management engine

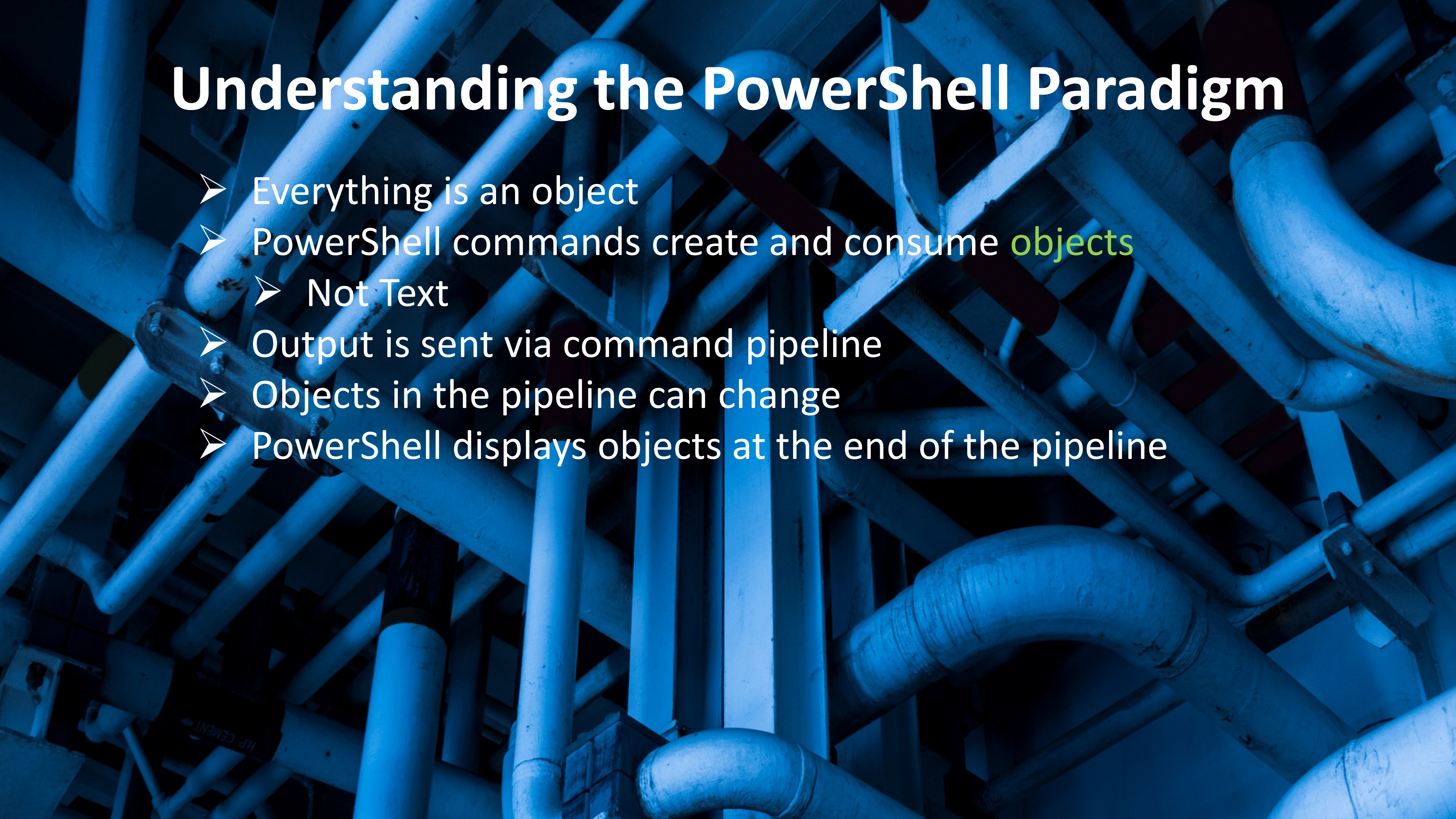
Interactive command console

Easy-to-learn scripting language

Learn once and apply everywhere

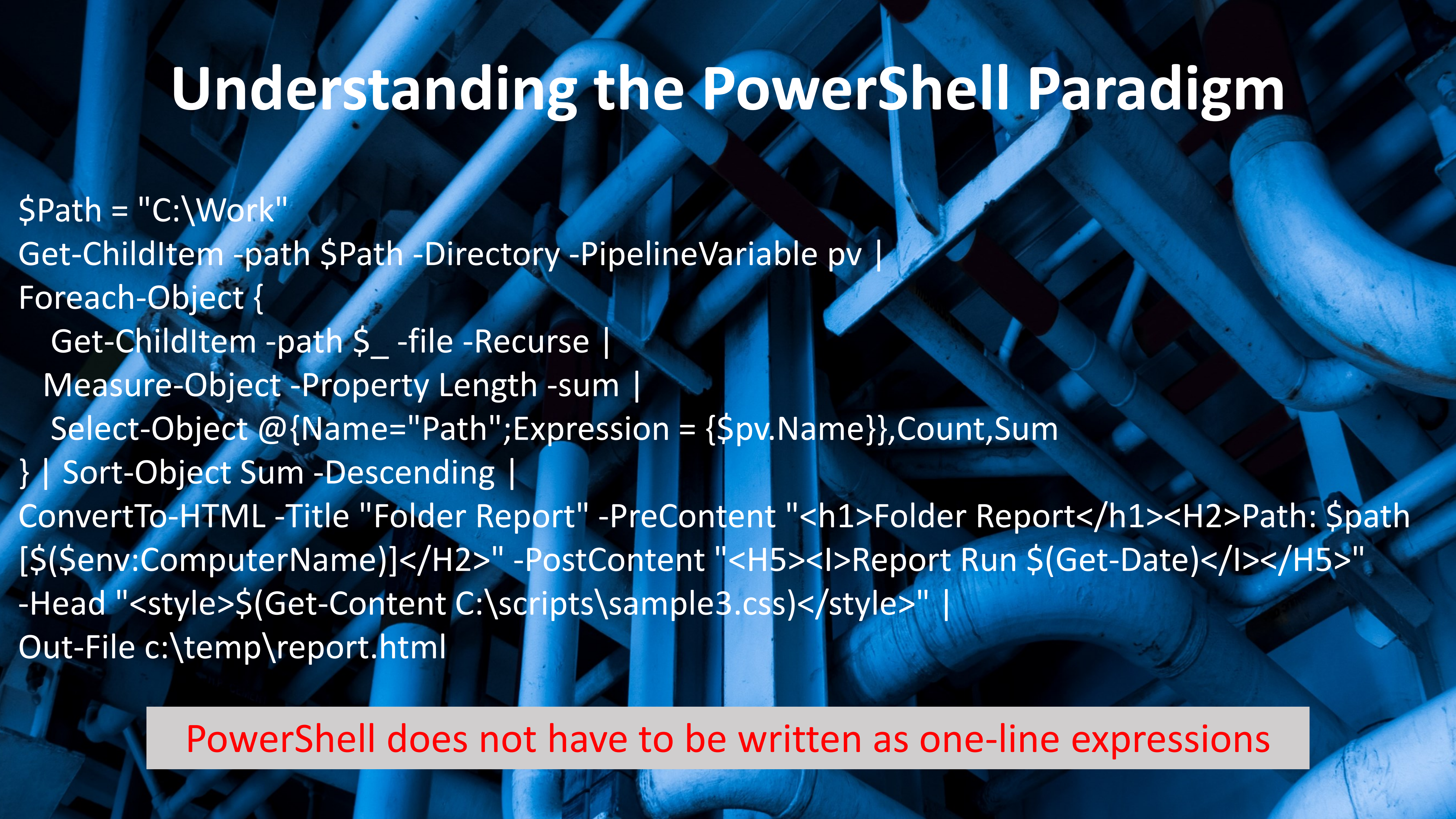
Recognize PowerShell’s limitations

Don’t force PowerShell to fit preconceptions



Understanding the PowerShell Paradigm

- Everything is an object
- PowerShell commands create and consume objects
  - Not Text
- Output is sent via command pipeline
- Objects in the pipeline can change
- PowerShell displays objects at the end of the pipeline

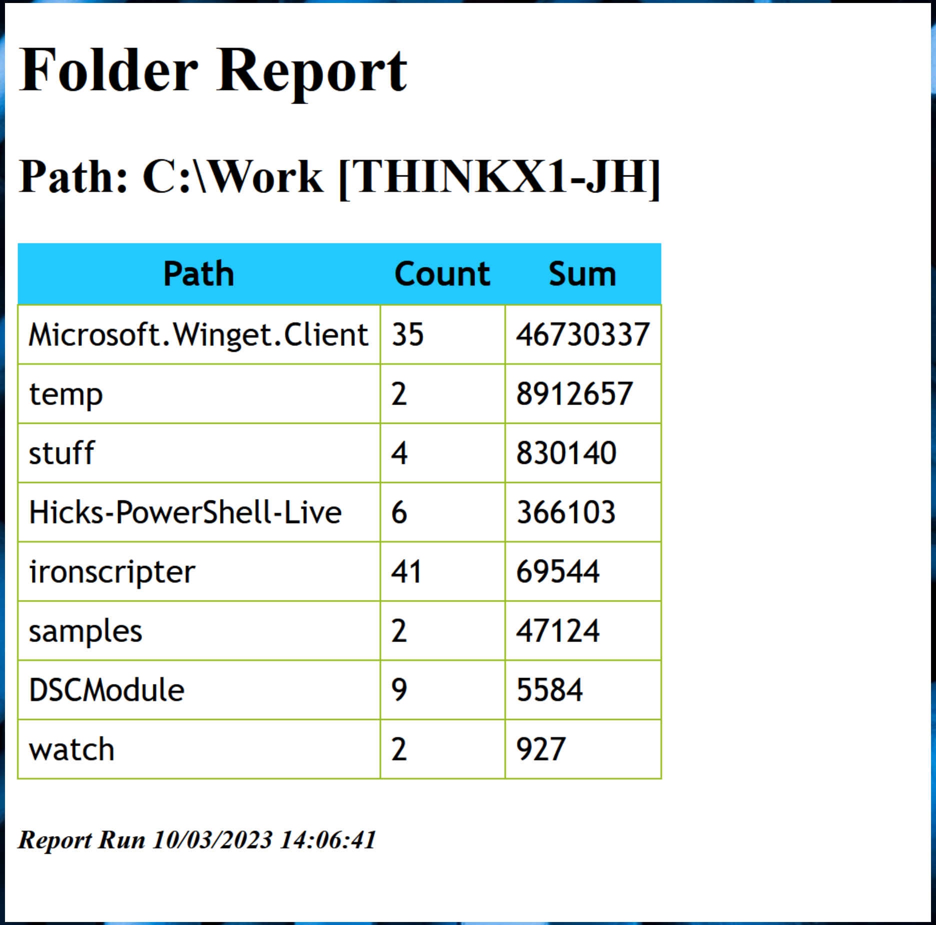


Understanding the PowerShell Paradigm

```
$Path = "C:\Work"
Get-ChildItem -path $Path -Directory -PipelineVariable pv |
Foreach-Object {
    Get-ChildItem -path $_ -file -Recurse |
   Measure-Object -Property Length -sum |
    Select-Object @{Name="Path";Expression = {$pv.Name}},Count,Sum
} | Sort-Object Sum -Descending |
ConvertTo-HTML -Title "Folder Report" -PreContent "<h1>Folder Report</h1><H2>Path: $path
[$($env:ComputerName)]</H2>" -PostContent "<H5><I>Report Run $(Get-Date)</I></H5>"
-Head "<style>$(Get-Content C:\scripts\sample3.css)</style>" |
Out-File c:\temp\report.html
```

PowerShell does not have to be written as one-line expressions

Understanding the PowerShell Paradigm



Folder Report

Path: C:\Work [THINKX1-JH]

Path	Count	Sum
Microsoft.Winget.Client	35	46730337
temp	2	8912657
stuff	4	830140
Hicks-PowerShell-Live	6	366103
ironscripter	41	69544
samples	2	47124
DSCModule	9	5584
watch	2	927

Report Run 10/03/2023 14:06:41

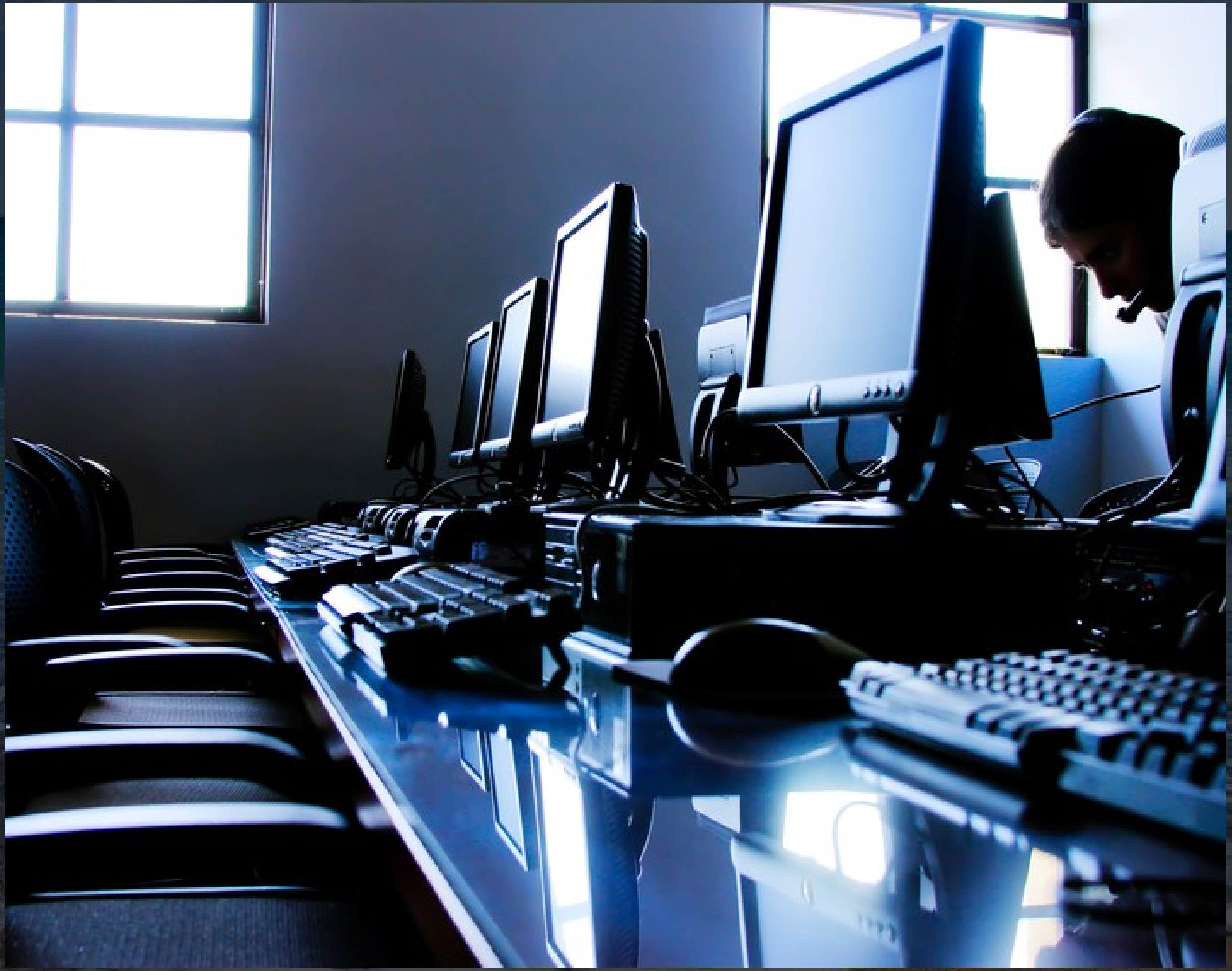
Language Elements

- Variables
  - Hold temporary values
- Arrays
  - Collections or groups of objects
  - Not required to be the same type
- Hashtables
  - Name/Value pair
  - Dictionary object
  - Ordered option
- Script blocks
  - Executable blocks of code
  - Can parameterize

Read the help!

Console to Script

- Start with commands at the console
- Create a basic script
    - Parameterize variables
    - Basic error handling
- Create a simple function
    - Basic parameters
- Refine to an advanced function
    - Pipeline input
    - Advanced parameters
    - Parameter sets
    - SupportsShouldProcess (-WhatIf)
- Package related functions into a module



Toolmaking Design

- Who will be using your tool?
- What are their expectations?
- How will they consume the output?
- Assume nothing
- Don’t force the user to do your work

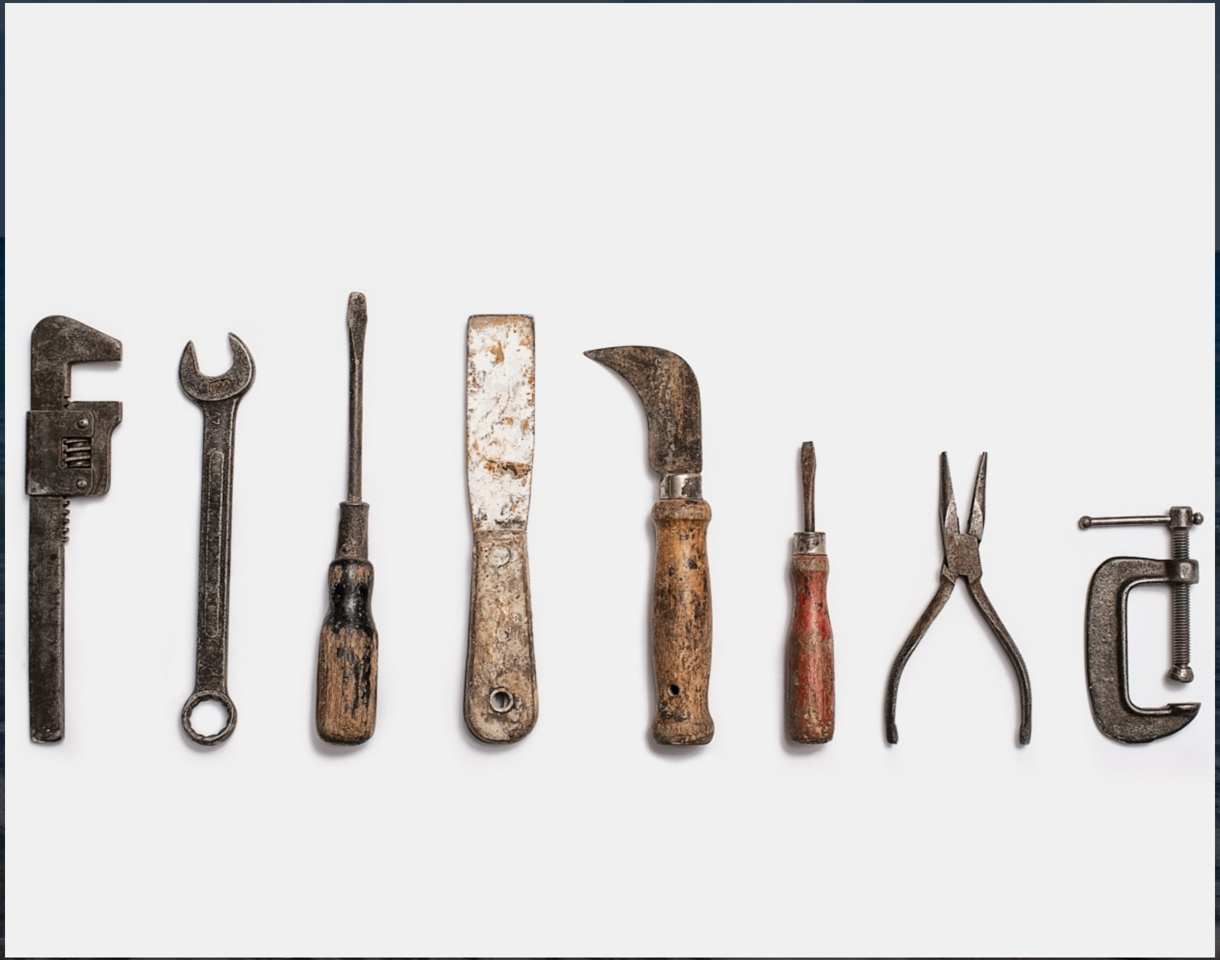
DEMO

PowerShell Patterns and Practices

Focus on techniques and concepts

Toolmaking Tips

- ✓ Learn how to use the help system and use it often
- ✓ Follow scripting best practices
  - ✓ Full cmdlet and parameter names
  - ✓ Follow naming standards and conventions
  - ✓ Code formatting matters
  - ✓ Document from the beginning
  - ✓ Separate formatting from output
  - ✓ Separate data from code
- ✓ Script in layers
- ✓ Test in a clean PowerShell environment
  - ✓ Learn Pester
  - ✓ `powershell | pwsh -nologo -noprofile`



Extended Q&A

https://github.com/jdhitsolutions/MMSMiami-2023

Save the Dates

May 5-9, 2024

Oct 20-23, 2024