



***Why Does it Work that Way?***  
Inside the PowerShell Runtime

In the beginning was the pipeline...

Pipeline

Need a place to put my stuff...

Session  
State

Pipeline

# Oh – and a language...

Session  
State

Language

Pipeline

# Need something to hold the pieces...

Execution Context/Automation Engine

Session  
State

Language

Pipeline

# Wrap it up in a single interface...

Runspace

Execution Context/Automation Engine

Session State

Language

Pipeline

# Huh? What's a "runspace"?

Runspace

Execution Context/Automation Engine

Session State

Language

Pipeline

# PSSession – User's Get It

PSSession

Runspace

Automation Engine/Execution Context

Session  
State

Language

Pipeline

# PSSession – User's Get It

PSSession

Runspace

Automation Engine/Execution Context

Session  
State

Language

Pipeline

# Inherited from the shell

- Expression-oriented language
  - Everything returns a value

```
1; 2; 3
```

```
Get-Date; Dir; Get-WmiObject Win32_BIOS
```
- `$x = foreach ($i in 1..10) { if ( $i % 2) { $i } }`
- Everything is a command
  - alias, function, cmdlet, script, applications

# Why scriptblocks?

- Added to solve a problem MMC had.
  - That bit never shipped but scriptblocks did
- First-class functions
  - $\$x = \{ \text{param } (\$x, \$y) \$x + \$y \}; \& \$x 2 3$
- Higher order functions
  - Functions that take functions as arguments
    - Where-Object, ForEach-Object

# ScriptBlocks are Compiled

- Shells use chunks of text – leads to runtime syntax errors
- Scriptblocks are fully compiled and parameterized
  - Eliminate (reduce?) code injection attacks