Learning the PowerShell Language

USING POWERSHELL VARIABLES

Jeff Hicks
AUTHOR/TEACHER

@jeffhicks | https://jdhitsolutions.com

Welcome

This is the next step in your PowerShell journey

Running basic and simple commands is the first step

Learn the language to do more

Getting Ready

Windows 10 Desktop

PowerShell 7.x installed

Course downloads

Optional: Visual Studio Code

What is a Variable?

A “container” that holds PowerShell “things”

A variable is nothing without something in it

Variables make PowerShell re-usable

PowerShell starts with many pre-defined variables

You can define your own

You can change values

Variables are not persistent

```
PS C:\> $a = 1
```

Creating Variables

Assign a value

The variable name is ‘a`

Use the $ to reference it in PowerShell

```
PS C:\> $a

1
```

Creating Variables

```
PS C:\> $a = 2
```

Creating Variables

Assign a new value

Lasts for the duration of your PowerShell session

```
PS C:\> $a = 2
```

Variables are Placeholders

```
PS C:\> $a = 2
PS C:\> Get-Vegetable | Select-Object -First $a

UPC     Count Name                 State    Color
---     ----- ----                 -----    -----
4078       12 corn                 Roasted  yellow
4064        4 tomato               Raw      red
```

Variables are Placeholders

```
PS C:\> $a = 5
```

Variables are Placeholders

Change the variable value

```
PS C:\> $a = 5
PS C:\> $b = Get-Process | Select-Object -First $a
```

Variables are Placeholders

Results will be saved to variable b

```
PS C:\> $b | Select-Object Name
Name
----
ApplicationFrameHost
Box
Box.Desktop.UpdateService
BoxUI
cdarbsvc_v1.0.0_x64
```

Variables are Placeholders

```
PS C:\> $b | Sort-Object ws -Descending | Select-Object name,ws

Name                                   WS
----                                   --
Box                             131850240
BoxUI                            65785856
ApplicationFrameHost             41967616
Box.Desktop.UpdateService        34680832
cdarbsvc_v1.0.0_x64               3141632
```

Variables are Placeholders

Don’t need to re-run Get-Process

Use when working with results from long-running commands

Variable Cmdlets

Get-Variable

New-Variable

Set-Variable

Remove-Variable

Variables are generally independent

You can remove a variable without affecting the original source

You can remove the original source without affecting the variable

Test everything

Demo

Variables in Action

```
PS C:\> $name = "Jeff"
```

Variable Expansion

Typical string usage

```
PS C:\> $name = "Jeff"
PS C:\> "Hello, my name is $name."
```

Variable Expansion

Typical string usage

Showing in the console but you’ll do this more often in scripting

```
PS C:\> $name = "Jeff"
PS C:\> "Hello, my name is $name."
Hello, my name is Jeff.
```

Variable Expansion

Variables are expanded in double quotes

Works for simple values

```
PS C:\> $name = "Jeff"
PS C:\> 'Hello, my name is $name.'
```

Variable Expansion

But be careful of quoting

```
PS C:\> $name = "Jeff"
PS C:\> 'Hello, my name is $name.'
Hello, my name is $name.
```

Variable Expansion

Variables are not expanded within single quotes

```
PS C:\> $svc = Get-Service BITS
PS C:\> $svc | select name,status


Name  Status
----  ------
BITS Stopped
```

Complex Variable Expansion

An object with two properties

```
PS C:\> $svc = Get-Service BITS
PS C:\> $svc | select name,status


Name  Status
----  ------
BITS Stopped

PS C:\> "$svc.name is $svc.status"
```

Complex Variable Expansion

An object with two properties

This will fail

Need to use subexpressions

Demo

Variable Expansion

Advanced Options

Tee-Object

OutVariable

PipelineVariable

```
PS C:\> Get-Process ls* | Tee -Variable p
```

Tee-Object

Get expression result AND save to a variable

```
PS C:\> Get-Process ls* | Tee -Variable p

 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
      6     1.17       3.34       0.11    1452   0 LsaIso
     28    12.40      26.09   2,050.42    1460   0 lsass
     19    74.40       0.65       0.34    3392   1 LSB
```

Tee-Object

Get expression result AND save to a variable

```
PS C:\> $p | measure-object ws -sum

Count             : 3
Average           :
Sum               : 31789056
Maximum           :
Minimum           :
StandardDeviation :
Property          : WS
```

Tee-Object

Use the variable as a placeholder

```
PS C:\> $p | measure-object ws -sum -outvariable m

Count             : 3
Average           :
Sum               : 31789056
Maximum           :
Minimum           :
StandardDeviation :
Property          : WS
```

OutVariable

Common cmdlet parameter

Save output from a pipeline segment

```
PS C:\> $m.sum
31789056
```

OutVariable

I could also have used Tee-Object

```
PS C:\> 1..5 |
foreach-object -pipelinevariable a {$_} |
foreach-object -pipelinevariable b {$_*2} |
foreach-object { "$a * 2 = $b"}
```

PipelineVariable

Advanced concept

Save pipeline segment output across a pipeline

Temporary, in-memory variable

```
PS C:\> 1..5 |
foreach-object -pipelinevariable a {$_} |
foreach-object -pipelinevariable b {$_*2} |
foreach-object { "$a * 2 = $b"}
1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
4 * 2 = 8
5 * 2 = 10
```

PipelineVariable

Special use case scenarios

Demo

Other Variable Options