Working with Objects in the Pipeline

Jeff Hicks
AUTHOR/TEACHER

@jeffhicks | https://jdhitsolutions.com

Let the pipeline do the work for you.

Objects are Objects

Sort-Object

Group-Object

Select-Object

Measure-Object

```
PS C:\> Get-Service | Sort-Object -property DisplayName
```

Sort-Object

Specify a property name

Original objects are displayed

```
PS C:\> Get-Service | Group-Object -property StartType
```

Group-Object

Specify a property name

Cmdlet writes a new object to the pipeline

```
PS C:\> Get-ChildItem c:\work -file | Select-Object -first 3
```

Select-Object

You can select First or Last X number of objects

You can also skip Y number of objects

Cmdlet writes the original object to the pipeline

```
PS C:\> Get-Process | Select-Object -property ID,Name,WorkingSet
```

Select-Object

Specify properties

Cmdlet writes a new object to the pipeline with the same property names

```
PS C:\Data\> Get-ChildItem -file | Measure-Object -property Length -sum
```

Measure-Object

Specify properties to measure

Specify what type of measurement

Cmdlet writes a new object to the pipeline

Demo

Working with Objects

Working with Objects Individually

ForEach-Object

Do something with each object

$_

Write the results to the pipeline

```
PS C:\> 1..10 | ForEach-Object { $_ * 2 }
```

ForEach-Object

Each piped in object is processed individually

```
PS C:\> $servers | ForEach-Object –parallel { Get-WinEvent –logname Security
–Computername $_ -MaxEvents 5000 }
```

ForEach-Object -parallel

Each piped in object is used in the scriptblock

Runs in parallel

Justify the processing overhead

ForEach-Object has an alias of *foreach*

There is also a *foreach* keyword

The keyword is used more in scripting

You can use *foreach* and PowerShell will figure out what you mean

Help about_ForEach

Demo

ForEach-Object