LESSON 6



PS>

Loops

Learn it all @MyWebUniversity



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 6

Loops

foreach, ForEach-Object, for, while, do and loop control

The foreach Statement

foreach.ps1

```powershell
$files = Get-ChildItem -Path .\logs -Filter *.log
foreach ($f in $files) {
    '{0,-20} {1,8:N1} KB' -f $f.Name, ($f.Length / 1KB)
}
```

- foreach loops over a collection already in memory. It is fast and easy to read, and the …
- Here we loop over log files and print each name with its size in kilobytes.

01

ForEach-Object in the Pipeline

foreach_object.ps1

```
Get-ChildItem *.txt | ForEach-Object { $_.Name.ToUpper() }

1..5 | ForEach-Object { $_ * $_ }            # 1 4 9 16 25

'web1', 'web2', 'db1' | ForEach-Object -Parallel {
    Test-Connection $_ -Count 1 -Quiet
} -ThrottleLimit 3
```

- ForEach-Object works inside the pipeline, one object at a time, with dollar underscore as …
- Because it streams, it uses little memory even with millions of objects. PowerShell 7 adds …

02

for, while and do

loops.ps1

```
for ($i = 1; $i -le 3; $i++) { "server$i" }

$tries = 0
do {
    $ok = Test-Connection 'web1' -Count 1 -Quiet
    $tries++; Start-Sleep -Seconds 2
} until ($ok -or $tries -ge 5)
```

- The for loop counts with a start, a condition and a step, like C. while repeats as long as …
- do-while and do-until always run at least once, which is useful for retry loops.

03

break, continue and Ranges

find_big.ps1

```
foreach ($f in Get-ChildItem -Recurse -File) {
    if ($f.Attributes -match 'System') { continue }
    if ($f.Length -gt 1GB) {
        "Found: $($f.FullName)"
        break
    }
}
```

- continue skips to the next item and break leaves the loop. The range operator, two dots, ...
- Here we skip system files and stop after finding the first file larger than one gigabyte.

04

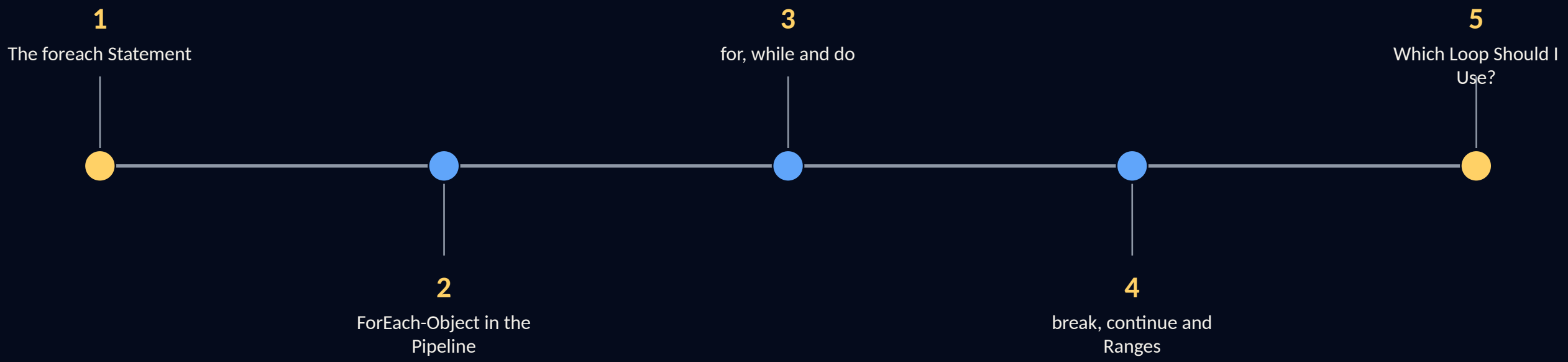
Which Loop Should I Use?

Loop	Best for
`foreach ($x in $list)`	data in memory, fast
`ForEach-Object`	pipelines, huge data, -Parallel
`for ($i...)`	you need the index
`do { } until ()`	retry until success

- Use foreach when the data is already in a variable and speed matters. Use ForEach-Object in …
- Use for when you need an index, and do-until for retrying something until it works.

05

Lesson Map


1
The foreach Statement
2
ForEach-Object in the Pipeline
3
for, while and do
4
break, continue and Ranges
5
Which Loop Should I Use?

Sources and Further Reading

foreach.ps1

```powershell
$files = Get-ChildItem -Path .\logs -Filter *.log
foreach ($f in $files) {
    '{0,-20} {1,8:N1} KB' -f $f.Name, ($f.Length / 1KB)
}
```

POWERSHELL SCRIPTING · 6

loops.ps1

```powershell
for ($i = 1; $i -le 3; $i++) { "server$i" }

$tries = 0
do {
    $ok = Test-Connection 'web1' -Count 1 -Quiet
    $tries++; Start-Sleep -Seconds 2
} until ($ok -or $tries -ge 5)
```

Thank you for watching

MyWebUniversity.com

Free technical education since 2000 — English and Dari

@MyWebUniversity · @MyWebUniversity-Dari