W06 ScriptBlock w PowerShell

ScriptBlock w PowerShell to obiekt reprezentujący blok kodu ujęty w nawiasy klamrowe { }. Można go przechowywać w zmiennej, przekazywać jako parametr, uruchamiać oraz stosować w wielu kontekstach jak potoki, zdarzenia, zadania w tle czy zdalne wywołania.

Przykład 1: Tworzenie ScriptBlock i wykonanie

```
$code = { Get-Date }

& $code

$code.Invoke()
```

Przykład 2: ScriptBlock jako parametr funkcji

```
function Invoke-WithLogging {
    param (
        [ScriptBlock]$Script
    )
    Write-Host "Start"
    & $Script
    Write-Host "End"
}

Invoke-WithLogging -Script { Get-Process | Select-Object -First 3 }
```

Przykład 3: ScriptBlock z parametrem

```
$script = {
    param($x, $y)
    return $x + $y
}

$script.Invoke(5, 7)
```

Przykład 4: Użycie z Where-Object i ForEach-Object

```
Get-Service | Where-Object { $_.Status -eq 'Running' }

1..5 | ForEach-Object { $_ * $_ }
```

Przykład 5: Funkcja przyjmująca ScriptBlock

```
function Process-List {
    param(
        [int[]]$Numbers,
        [ScriptBlock]$Transform
    )
    foreach ($n in $Numbers) {
        & $Transform.Invoke($n)
    }
}

Process-List -Numbers @(1, 2, 3, 4) -Transform { param($x) "$x^2 =
$($x*$x)" }
```

Przykład 6: ScriptBlock jako mini-funkcja

```
$multiply = { param($a, $b) $a * $b }
& $multiply 3 4
```

Zastosowanie w zaawansowanych scenariuszach

ScriptBlocki są używane także w:
- Start-Job -ScriptBlock { ... }
- Register-ObjectEvent -Action { ... }
- Invoke-Command -ScriptBlock { ... } -ComputerName ...

Zalety ScriptBlocków

- Są parsowane przy tworzeniu
- Są bezpieczniejsze niż ciągi znaków
- Można do nich przypisać parametry
- Działają w wielu kontekstach PowerShella