PSCONF.EU

PowerShell Conference Europe

# Chasing the seconds 2.0
## (Functions done right)

***Thorsten Butz***

VIENNA

20 - 23 June 2022

PSCONF.EU VIENNA 20 - 23 June 2022

# Many thanks to our sponsors:

SquaredUp

au2mator

PATCH MY PC

Chocolatey

System Frontier

ScriptRunner®

# about_Session

```
{
    "Title"    : "Chasing the seconds 2.0",
    "Subtitle": "Functions done right",
    "Speaker" : "Thorsten Butz",
    "Uri"      : "thorsten-butz.de",
    "Twitter" : "@thorstenbutz",
    "Podcast" : "slidingwindows.de"
}
```

@thorstenbutz

# about_Mugs

(


)

@thorstenbutz

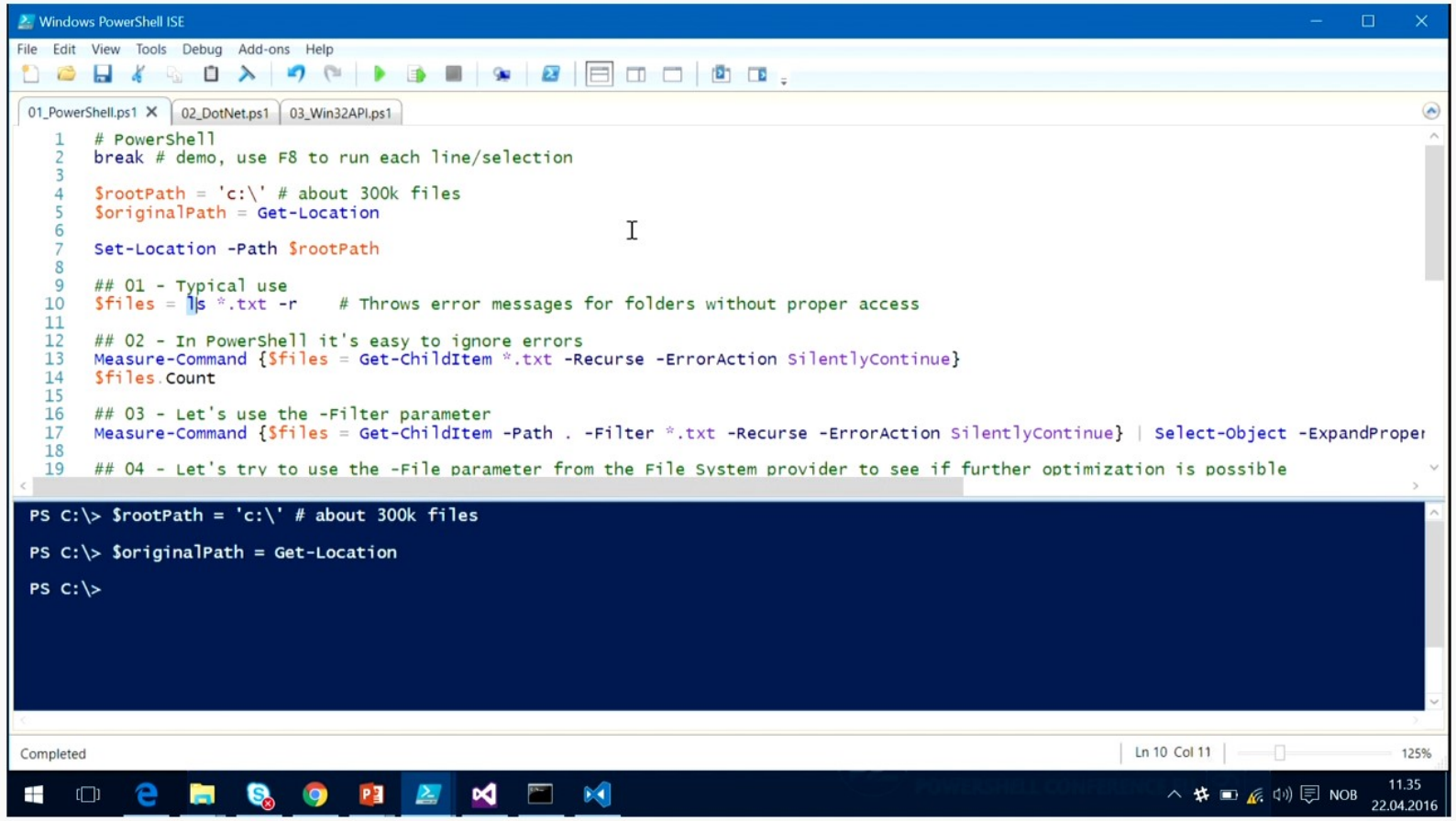
# Prologue

# Chasing the seconds 1.0


PSConfEU 2016

Øyvind Kallstad

# Chasing the seconds 1.0




Chasing the seconds: Optimizing file enumeration(Oyvind Kallstad)

1,191 views • May 13, 2016

https://youtu.be/erwAsXZnQ58

https://twitter.com/okallstad

https://github.com/psconfeu/2016/tree/master/Øyvind%20Kallstad

# Basics

# A well made cmdlet

```
$publicDNSServer = 'one.one.one.one','dns.google','dns9.quad9.net'

## A
Test-Connection -TargetName $publicDNSServer -TcpPort 53 -IPv4 -IPv6

## B
$publicDNSServer | Test-Connection -TcpPort 53

## C
Import-Csv -Path .\publicDNSServer.csv | Test-Connection -TcpPort 53
```

| | A | B |
|---|---|---|
| 1 | Targetname | Company |
| 2 | one.one.one.one | Cloudfare |
| 3 | dns.google | Google |
| 4 | dns9.quad9.net | Quad9 |

```
## Simple function

function foo {
  param (
    $bar
  )
  $bar * 23
}
foo -bar 42
```

```
## Advanced function (Script cmdlet)

function Invoke-Foo {
  [CmdletBinding]
  param (
    $bar
  )
  $bar * 23
  Write-Verbose -Message '(c) T. Butz'
}
Invoke-Foo -bar 42 -verbose
```

# Demo A

# In search of a blueprint

```
## Passing array as input

function Get-Honey {
  [CmdletBinding()]
    Param ([Parameter(
        Mandatory



      )]
      [string[]] $Flower
    )
    Begin {
    }
    Process {
    }
    End {
    }
}
```

```
## Pipeline support

function Get-Honey {
  [CmdletBinding()]
    Param ([Parameter(
        Mandatory,
        ValueFromPipeline,
        ValueFromPipelineByPropertyName
      )]
      [string] $Flower
    )
    Begin {
    }
    Process {
    }
    End {
    }
}
```

# Example tasks

- Get (basic) information about installed software
- Get (basic) hardware information

PARENTAL
ADVISORY
SIMPLIFIED CONTENT

# Example task 1

```
#################################################
## Registry: Get information about installed software
## (System-wide installations, basic version)
#################################################

$path = 'HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*',
        'HKLM:\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*'

Get-ItemProperty -Path $path |
  Where-Object -Property 'DisplayName'  |
    Select-Object -Property 'DisplayVersion','DisplayName','UninstallString'
```

# Example task 2

```
#################################################################
## WMI: Get some basic system information
#################################################################
$win32Bios = Get-CimInstance -ClassName Win32_BIOS
$win32OS =   Get-CimInstance -ClassName Win32_Operatingsystem

[PSCustomObject] @{
    'BIOSManufacturer' = $win32Bios.Manufacturer
    'BIOSVersion'      = $win32Bios.Version
    'BIOSReleaseDate'  = $win32Bios.ReleaseDate
    'OSCaption'        = $win32OS.Caption
    'OSBuildNumber'    = $win32OS.BuildNumber
    'OSInstallDate'    = $win32OS.InstallDate
    'OSLastBootUpTime' = $win32OS.LastBootUpTime
}
```

# Demo B

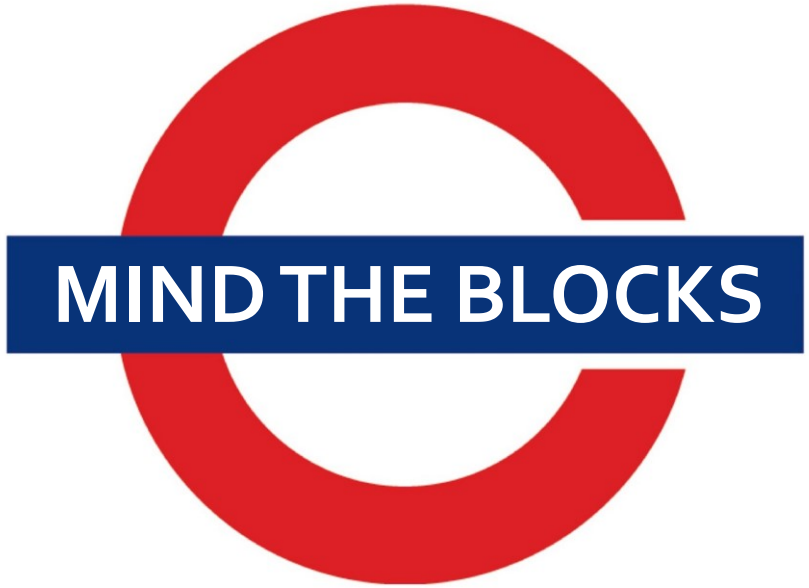
# Measurement results

average values from 10 runs each

| | SoftwareInventory All PCs online | HardwareInfo(WMI) All PCs online | SoftwareInventory 50 % offline | HardwareInfo(WMI) 50 % offline |
|---|---|---|---|---|
| 1 Array as input | 3809 ms | 1787 ms | 32924 ms | 32460 ms |
| 2 Pipelining `Process{}` | 25934 ms ≈ 25 sec | 24698 ms ≈ 25 sec | 518134 ms ≈ 9 min | 519516 ms ≈ 9 min |
| 3 Pipelining `End{}` | 3711 ms | 1658 ms ≈ 2 sec | 31480 ms | 37477 ms |
| 4 Pipelining `End{},Sessions` | 2000 ms ≈ 2 sec | 1855 ms | 22500 ms ≈ 23 sec | 22386 ms ≈ 22 sec |
| 5 Pipelining `End{},Sessions, ValidateScript` | 4817 ms | 4672 ms | 194415 ms | 193094 ms |
| 6 CIMSession | | 2689 ms | | 22473 ms |

# Summary

- The "input processing methods" (begin/process/end-blocks) are IMPORTANT !
- ValidateScript can be counter-productive (does not process async)
- Be user-friendly: enable pipeling the smart way !



# The blueprint

```
function Get-Honey {
  [CmdletBinding()]
    Param ([Parameter(
        Mandatory,ValueFromPipeline,ValueFromPipelineByPropertyName
      )]
      [string[]] $Flower
    )
    Begin {
    }
    Process {
       [string[]] $Flowers += $Flower
    }
    End {
      # Asnychronous processing
      Invoke-Bee -Destination $Flowers -ScriptBlock { Collect-Honey }
    }
}
```

KEEP

CALM

AND

ENJOY

#PSCONFEU

```
{
 "Venue"    : "PSConfEU 2022",
 "Title"    : "Chasing the seconds 2.0",
 "Subtitle": "Functions done right",
 "Speaker" : "Thorsten Butz",
 "Uri"      : "thorsten-butz.de",
 "Twitter" : "@thorstenbutz",
 "Podcast" : "slidingwindows.de"
}
```