



PowerShell Standards Agency

**Paul Broadwith
Craig Porteous**

About Us



Craig
Porteous

Data Engineer @ Incremental Group

Worked in BI with the Microsoft stack for more than 10 years.

SQL Glasgow UG co-leader. SQLGLA Creator.

 <https://craigporteous.com>

 @cporteous

 <https://github.com/cporteous>

 <https://www.linkedin.com/in/craigporteous/>



Paul
Broadwith

Senior Technical Engineer @ Chocolatey Software

25 years in IT in the defence, government, financial services and nuclear industry sectors.

Scottish PowerShell & DevOps User Group Founder

 <https://pauby.com>

 @pauby

 <https://github.com/pauby>

 <https://www.linkedin.com/in/paulbroadwith/>

#PowerShellStandardsAgency



**PowerShell
Standards
Agency**



PowerShell
Standards
Agency

There are
NO
standards!



Aims

Introduce some tools

Outline techniques to develop your own standard

Look at some advanced language features

Talk about the importance of Community



PowerShell
Standards
Agency

Tools

Visual Studio Code

PSScriptAnalyzer

Pester





PowerShell
Standards
Agency

Visual Studio Code



Replacement for PowerShell ISE

Cross-platform

Extensions for many different languages



PowerShell
Standards
Agency

PSScriptAnalyzer

*“PSScriptAnalyzer **checks the quality of Windows PowerShell code by running a set of rules.** The rules are **based on PowerShell best practices** identified by PowerShell Team and the community.”*



Pester

Tests your code

```
Executing all tests in '.'
```

```
Executing script C:\projects\pester_nohwnd\Examples\Planets\Get-Planet.Tests.ps1
```

```
Describing Get-Planet
```

```
[+] Given no parameters, it lists all 8 planets 39ms
```

```
Context Filtering by Name
```

```
[+] Given valid -Name 'Earth', it returns 'Earth' 27ms
```

```
[+] Given valid -Name 'ne*', it returns 'Neptune' 16ms
```

```
[+] Given valid -Name 'ur*', it returns 'Uranus' 17ms
```

```
[+] Given valid -Name 'm*', it returns 'Mercury Mars' 15ms
```

```
[+] Given invalid parameter -Name 'Alpha Centauri', it returns $null 17ms
```


```
Tests completed in 134ms.
```

```
Tests Passed: 6, Failed: 0, Skipped: 0, Pending: 0, Inconclusive: 0
```



Code Review

```
function Start_DeathStar {  
    Param ([string]$param1)  
    if(!$param1){  
        Write-host "You didn't provide a parameter"; Return}  
    if ($param1 -eq "Alderaan")  
    {  
        TurnON -param1 'Alderaan'  
    } else {  
        turnOFF }  
}  
  
function TurnON(Param ())  
{  
    $true  
}  
  
function turnOFF{  
    #this is False  
    $false}
```



The Road To Standards

Consistency

Readability

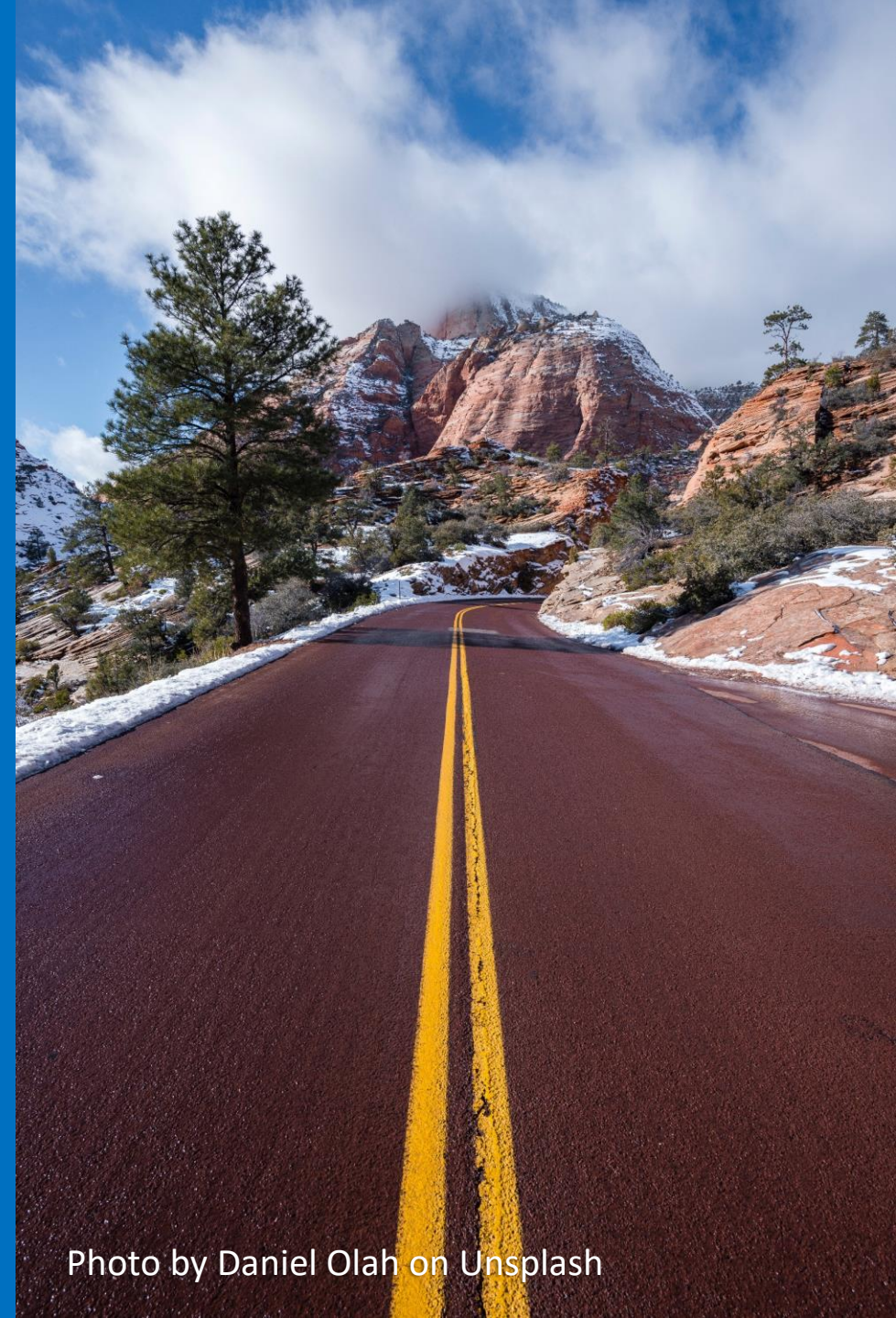


Photo by Daniel Olah on Unsplash



Bracing Style

There are three commonly used types:



Allman (or BSD Indent Style)

Kernighan & Ritchie (OTBS)

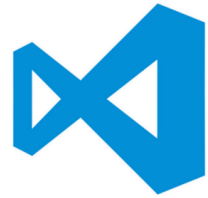


Stroustrup



Bracing Style

DEMO 1



Auto formatting with Visual Studio Code

SHIFT + ALT + F

(Reformats your document to your chosen brace style)

To set your brace style:

Ctrl + ,

Search for 'powershell brace'

Bracket Pair
Colorizer



Bracing Style

DEMO 2





PowerShell
Standards
Agency

Tabs vs Spaces



Whichever style you choose

BE CONSISTENT!

(and remember Pester!)

The Road To Standards

Consistency

Readability

Naming

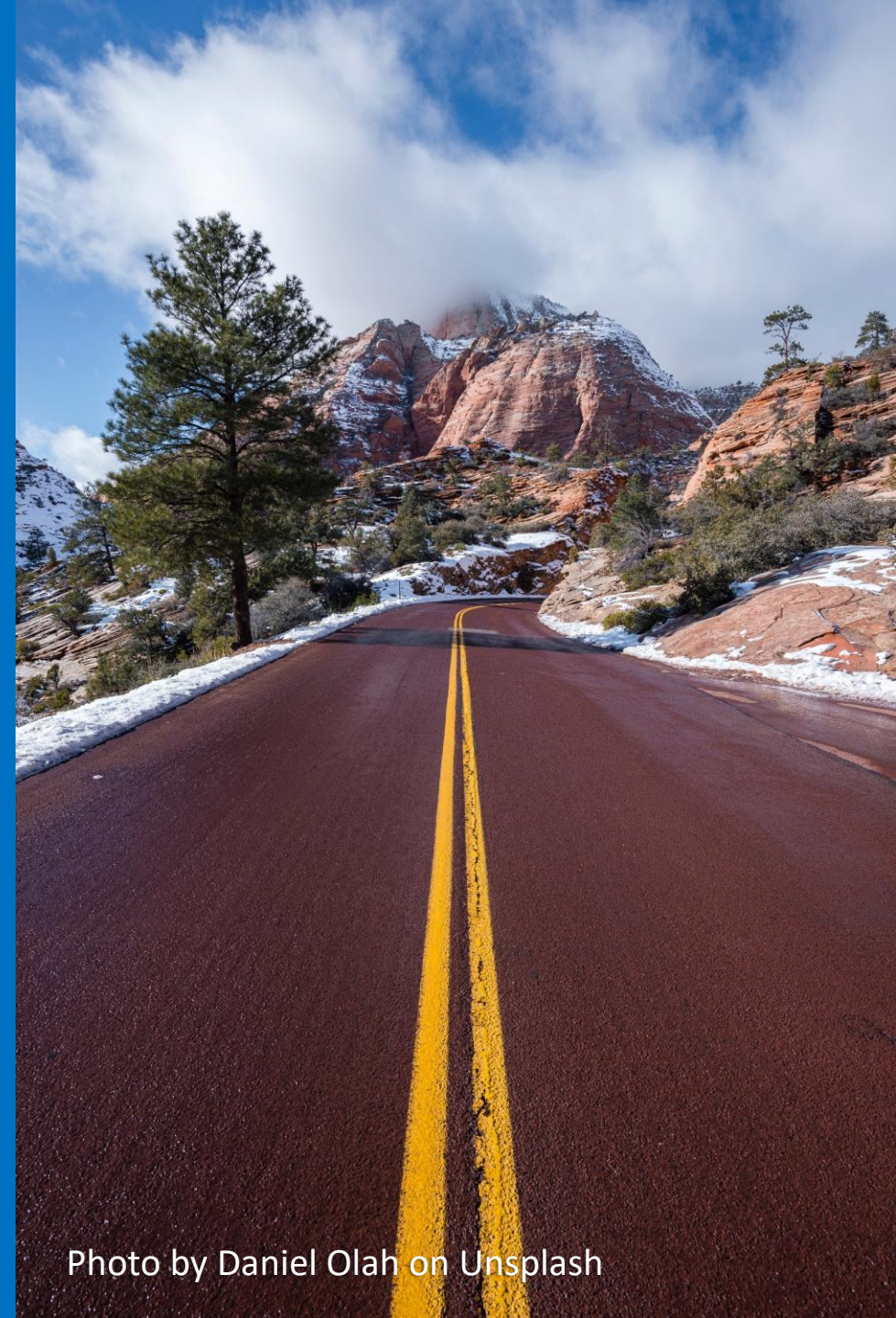


Photo by Daniel Olah on Unsplash



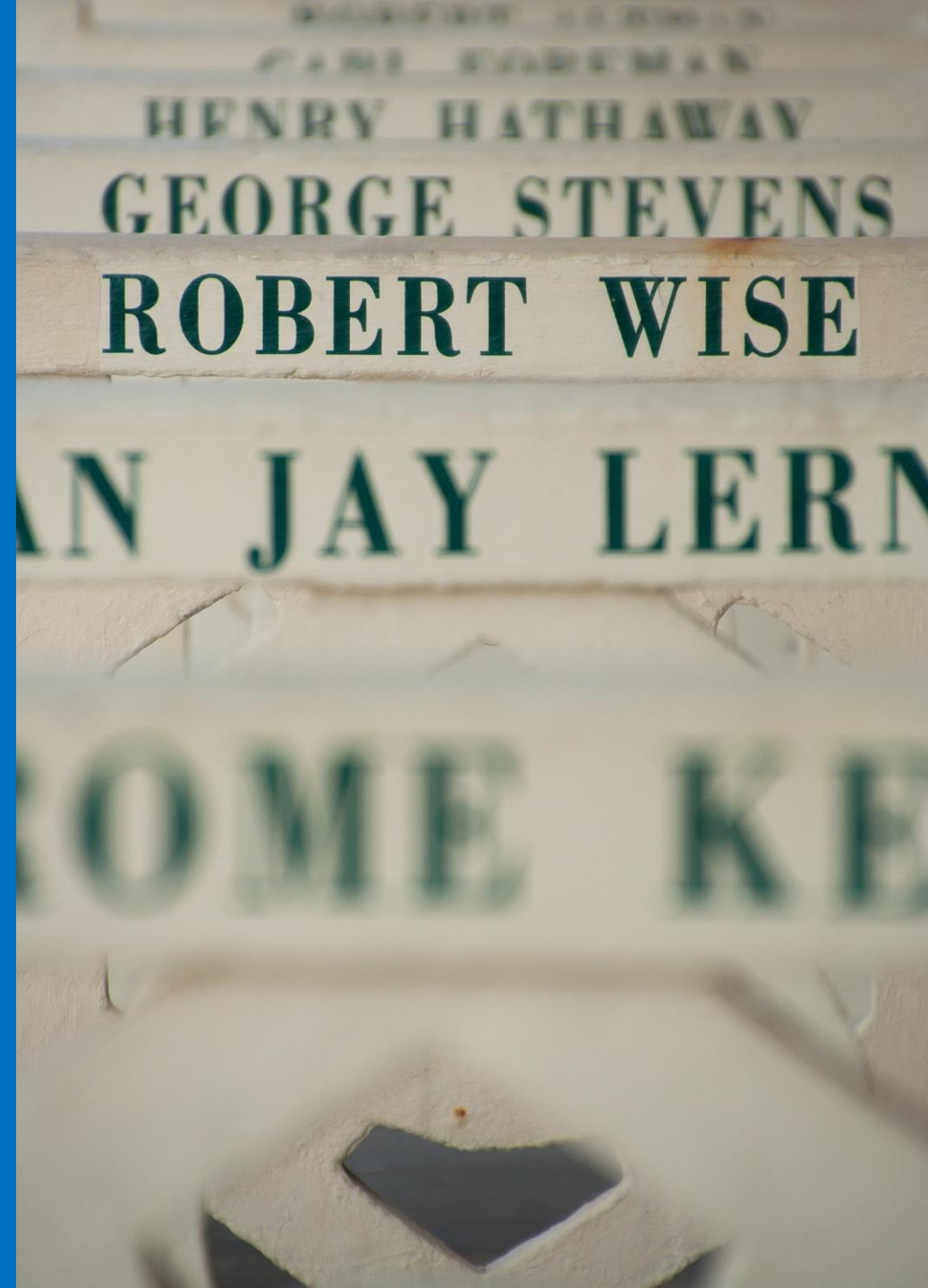
PowerShell
Standards
Agency

Naming Style

Variables

Functions

Parameters





Variables

Avoid Data Type Names

Describe its Use

Use camelCase

Bad Naming

```
$myVariable = 10  
  
$stringName = 'Luke'  
  
$whosTheDaddy = 'Darth Vader'  
  
$zigazigahh = ' a song'
```

Good Naming

```
$totalFish = 10  
  
$firstName = 'Luke'  
  
$tieFighterPilot = 'Darth Vader'  
  
$song = 'Wannabe'
```



Functions

Use Verb-Noun naming.

Use approved verbs – find them with **Get-Verb**.

PSScriptAnalyzer warns of unapproved verbs.

```
PS C:\> Get-Verb
```

Verb	Group
----	-----
Add	Common
Clear	Common
Close	Common
Copy	Common
Enter	Common
Exit	Common
Find	Common
Format	Common
Get	Common
Hide	Common
Join	Common
Lock	Common
Move	Common
New	Common
Open	Common
Optimize	Common
Pop	Common

Functions

DEMO 3



Functions

Use Singular Nouns

Use PascalCase

Bad Naming

```
function Get-Employees {}  
  
function ListAllStaff {}  
  
function start_death_star {}  
  
function fireboomstick {}
```

Good Naming

```
function Get-Employee {}  
  
function Get-AllStaff {}  
  
function Start-DeathStar {}  
  
function Invoke-BoomStick {}
```




Functions

Make them unique



dbatools

Verb-**Db**aNoun

Azure AD

Verb-**AzureAD**Noun

ReportingServicesTools

Verb-**RS**Noun



Parameters

Use Singular Nouns

Use PascalCase

Bad Naming

```
function Start-DeathStars {  
    Param (  
        [string[]]  
        $Targets  
    )  
}
```

Good Naming

```
function Start-DeathStar {  
    Param (  
        [string[]]  
        $Target  
    )  
}
```



Parameters

-Path

-ComputerName

Common parameter names

-Name

-Credential

-Force



PowerShell
Standards
Agency

Whichever style you choose

BE CONSISTENT!

The Road To Standards

Consistency

Readability

Naming

Comments

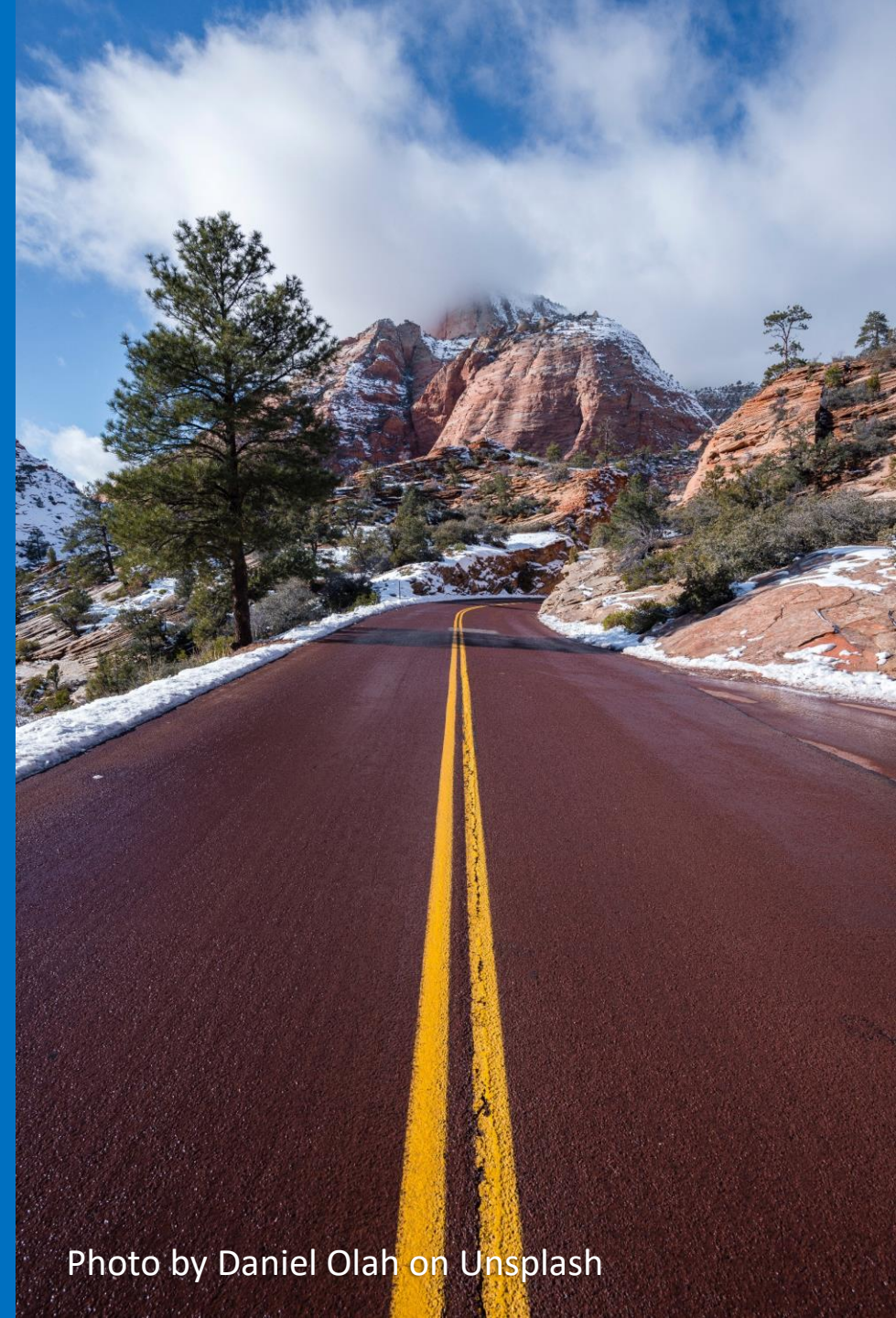


Photo by Daniel Olah on Unsplash



PowerShell
Standards
Agency

Commenting

Good code almost comments itself.

Delicate balance of help without
obscuring the code.



Photo by [David Preston](#) on [Unsplash](#)



Commenting



Caitlin Hudon 
@beeonaposity

Follow



Jumping back into code you wrote ages ago
like

- 10% luck
- 20% skill
- 15% concentrated power of will
- 5% pleasure
- 50% pain
- 100% wishing you used descriptive names

Avoid technical debt!



Commenting

Better
Comments



Single Line Comment **# ...**

```
# A single line comment
```

Block Comment **<# ... #>**

```
<# Comments over a  
|   few lines. #>
```

Rewrap





Commenting

```
# setting $myVar to 10
$myVar = 10
# Copying Notepad.exe from C:\Windows to C:\Temp
Copy-Item -Path "C:\Windows\notepad.exe" -Destination "C:\Temp\"
```



```
$fileList = Get-ChildItem "C:\Windows" |
    Where-Object { $_.PsIsContainer -eq $false } |
    Group-Object -property extension |
    Sort-Object -Property count -Descending |
    Select-Object -First 5
```



```
# the number of extensions to get
$top = 10
Copy-Item -Path "C:\Windows\notepad.exe" -Destination "C:\Temp\"

# Getting the top 5 file extensions used in the Windows folder
$topExtensions = Get-ChildItem "C:\Windows" |
    Where-Object { $_.PsIsContainer -eq $false } |
    Group-Object -property extension |
    Sort-Object -Property count -Descending |
    Select-Object -First $top
```

The Road To Standards

Consistency

Readability

Naming

Comments

Write Help

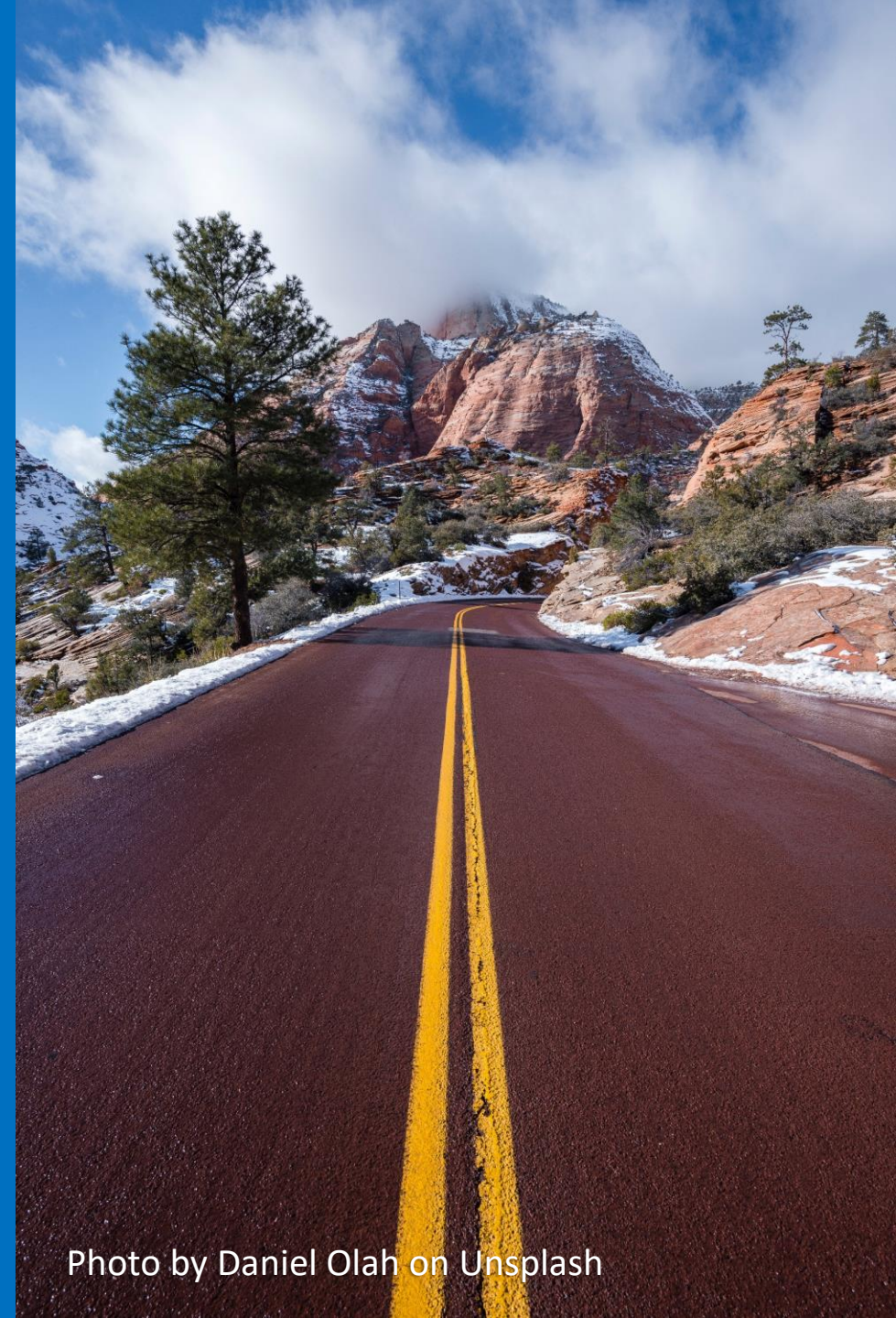


Photo by Daniel Olah on Unsplash



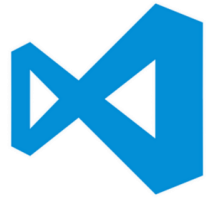
PowerShell
Standards
Agency

Comment based Help

DEMO 4



PowerShell
Standards
Agency



Auto inserting help with Visual Studio Code

```
##
```

Before or after your function name to have VS Code automatically insert the comment based help.



PowerShell
Standards
Agency

Whichever way you add comments and help

BE CONSISTENT!



PowerShell
Standards
Agency

Language Features

Advanced Functions

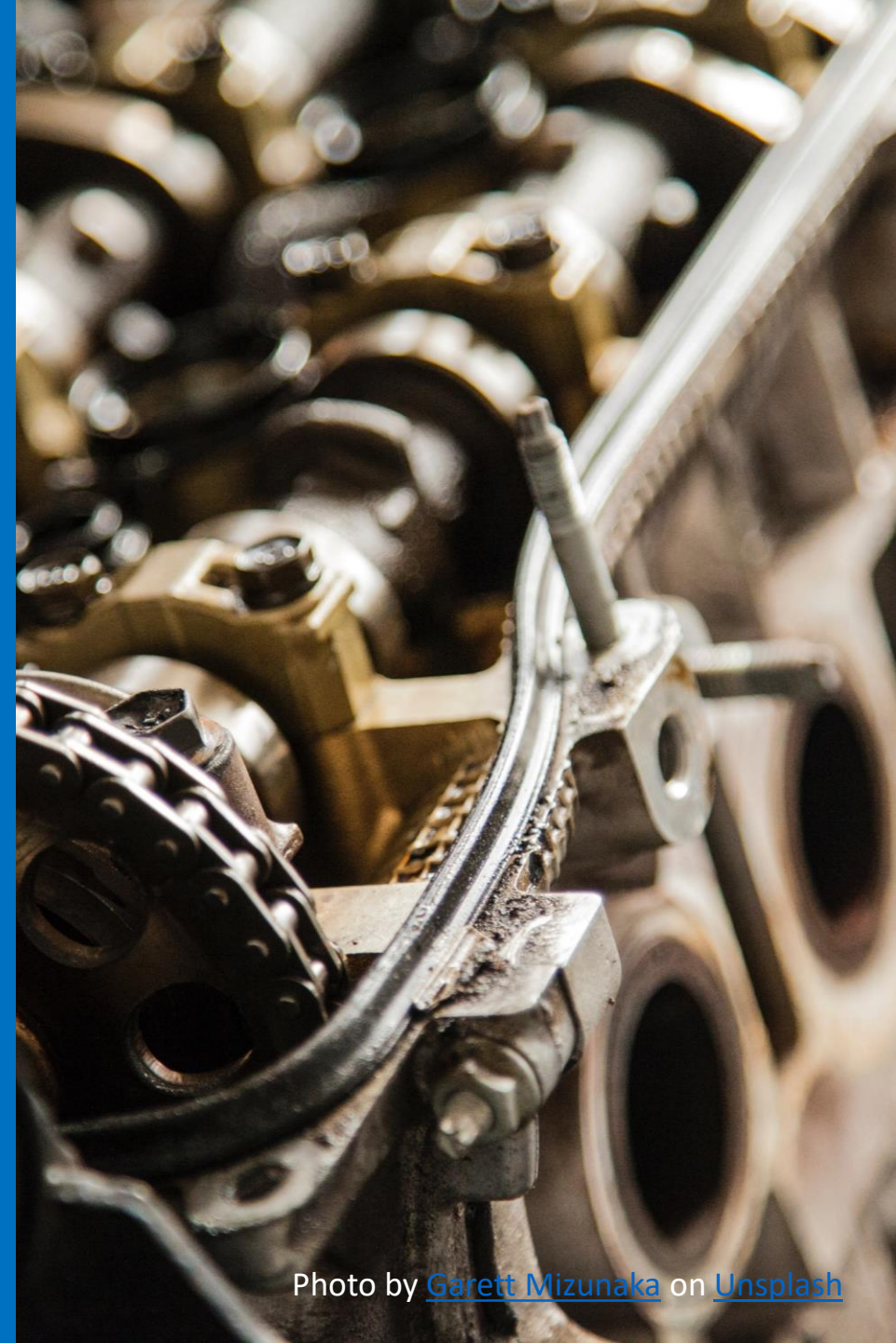


Photo by [Garett Mizunaka](#) on [Unsplash](#)



Advanced Functions

Advanced Functions

[CmdletBinding()]
Param()

-ErrorAction

-Debug

-Verbose

Access to the pipeline

Etc...



Advanced Functions

Use **Write-Verbose**

```
function Set-Quota {  
    [CmdletBinding()]  
    Param (  
        [int]$Quota  
    )  
  
    Write-Verbose "Setting file quota to '$Quota'."  
    Set-FileQuota -FileQuota $Quota  
}
```

Add **-Verbose** to the call.

```
PS C:\> Set-Quota 10 -Verbose  
VERBOSE: Setting file quota to '10'.
```




PowerShell
Standards
Agency

Advanced Functions

DEMO 5



PowerShell
Standards
Agency

Language Features

Advanced Functions

Built-In Validation

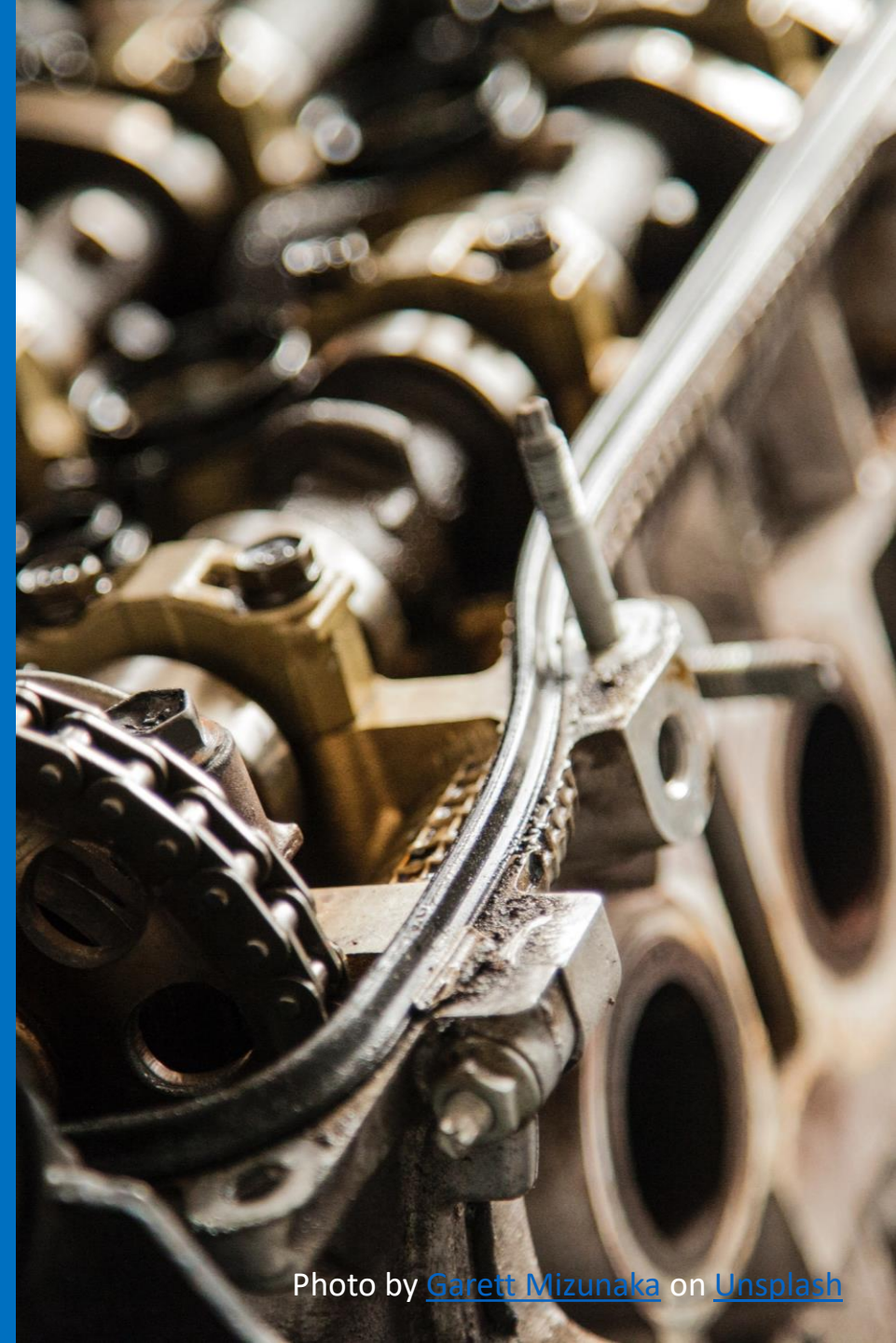


Photo by [Garett Mizunaka](#) on [Unsplash](#)



Leverage Built-In Validation

#Requires

States code pre-requisites

```
#Requires -Version <N>[.<n>]  
#Requires -PSSnapin <PSSnapin-Name> [-Version <N>[.<n>]]  
#Requires -Modules { <Module-Name> | <Hashtable> }  
#Requires -ShellId <ShellId>  
#Requires -RunAsAdministrator
```



Leverage Built-In Validation

Parameter Attributes

Mandatory

HelpMessage

```
function Start-DeathStar {  
    ....[CmdletBinding()]  
    ....Param (  
        ....[Parameter(Mandatory,  
        ....    HelpMessage = 'Planet to target')]  
        ....[string]  
        ....$Target  
    ....)  
}
```

```
C:\> Start-DeathStar
```

```
cmdlet Start-DeathStar at command pipeline position 1  
Supply values for the following parameters:  
(Type !? for Help.)  
Target: !?  
Planet to target  
Target:
```



Leverage Built-In Validation

Validation Attributes

```
function Start-DeathStar {  
    Param (  
        [ValidateSet(  
            'Tatooine', 'Alderaan')]  
        [string]$Target  
    )  
}
```

[ValidateCount(min, max)]

[ValidateLength(min, max)]

[ValidatePattern(<REGEX>)]

[ValidateScript({<SCRIPTBLOCK>})]

[ValidateRange(min, max)]

[ValidateSet('Value1', 'Value2')]



PowerShell
Standards
Agency

Language Features

Advanced Functions

Built-In Validation

Use the Pipeline and Objects

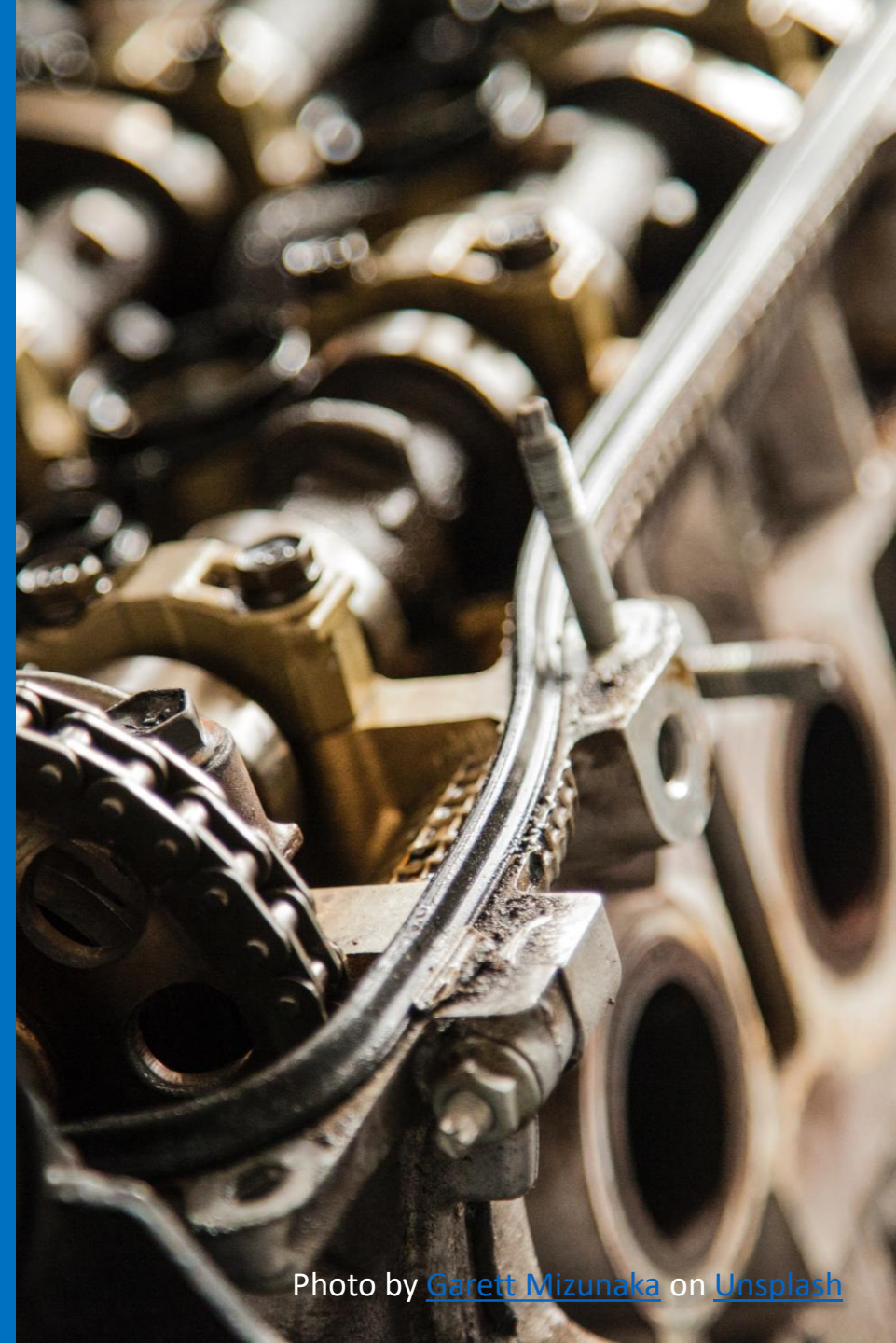


Photo by [Garett Mizunaka](#) on [Unsplash](#)



Use the Pipeline and Objects

Get-Service

Select-Object Name

ConvertTo-Json

```
PS C:\> Get-Service | Select-Object Name | ConvertTo-Json  
[  
  {  
    "Name": "AdobeFlashPlayerUpdateSvc"  
  },  
  {  
    "Name": "AJRouter"  
  },  
  {  
    "Name": "ALG"  
  },  
]
```



PowerShell
Standards
Agency

Language Features

Advanced Functions

Built-In Validation

Use the Pipeline and Objects

Filter left, format right

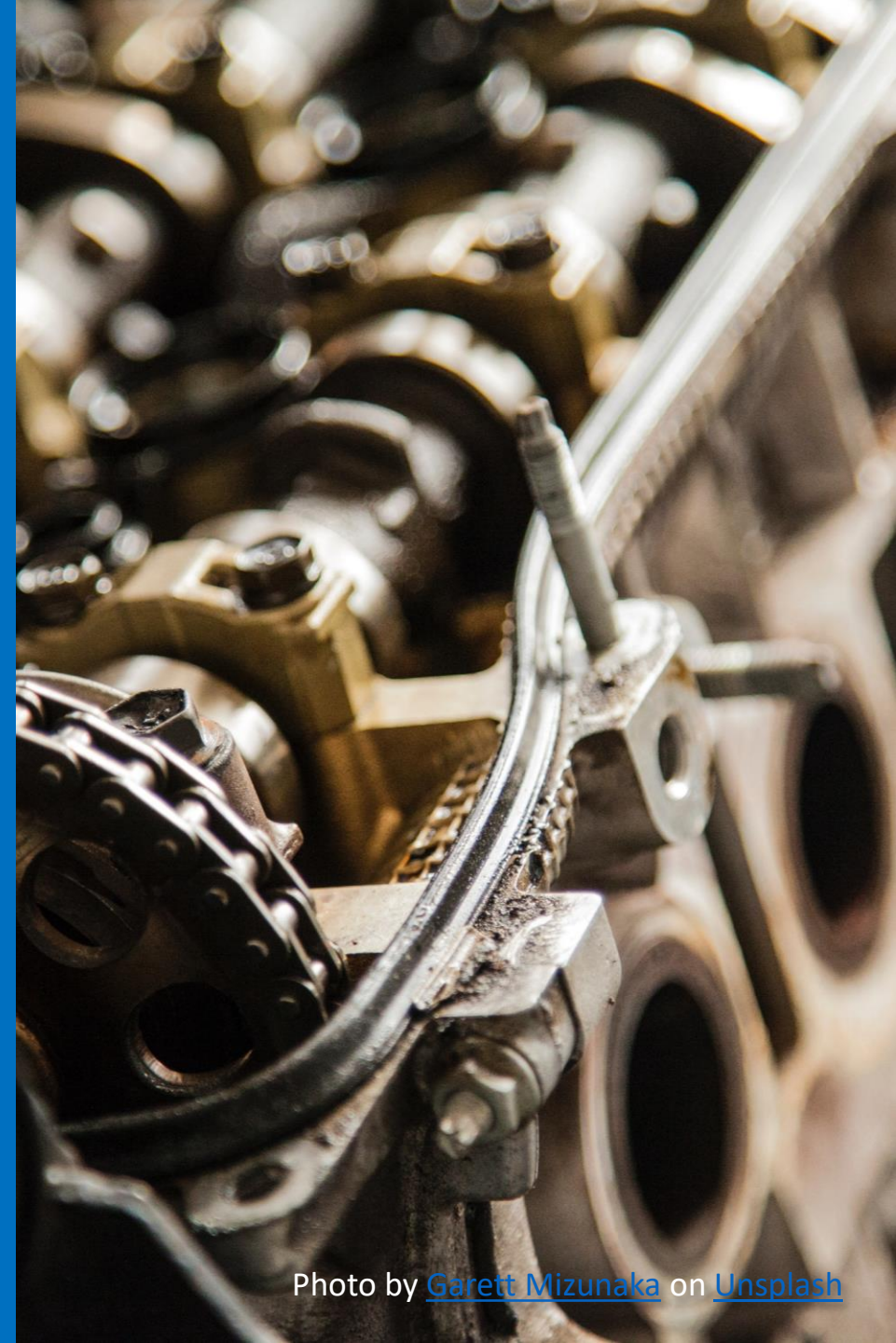


Photo by [Garett Mizunaka](#) on [Unsplash](#)



Filter Left, Format Right



```
# Measure unfiltered command
$noFilterTiming = Measure-Command{
    Get-ChildItem -Path 'C:\' -Recurse | Where-Object Extension -eq ".exe" | Select-Object FullName
}

# Measure Filter first command
$filterTiming = Measure-Command{
    Get-ChildItem -Path 'C:\' -Recurse -Filter *.exe | Select-Object FullName
}
```

00:18

00:07



PowerShell
Standards
Agency

Language Features

Advanced Functions

Built-In Validation

Use the Pipeline and Objects

Filter left, format right

Don't use Aliases

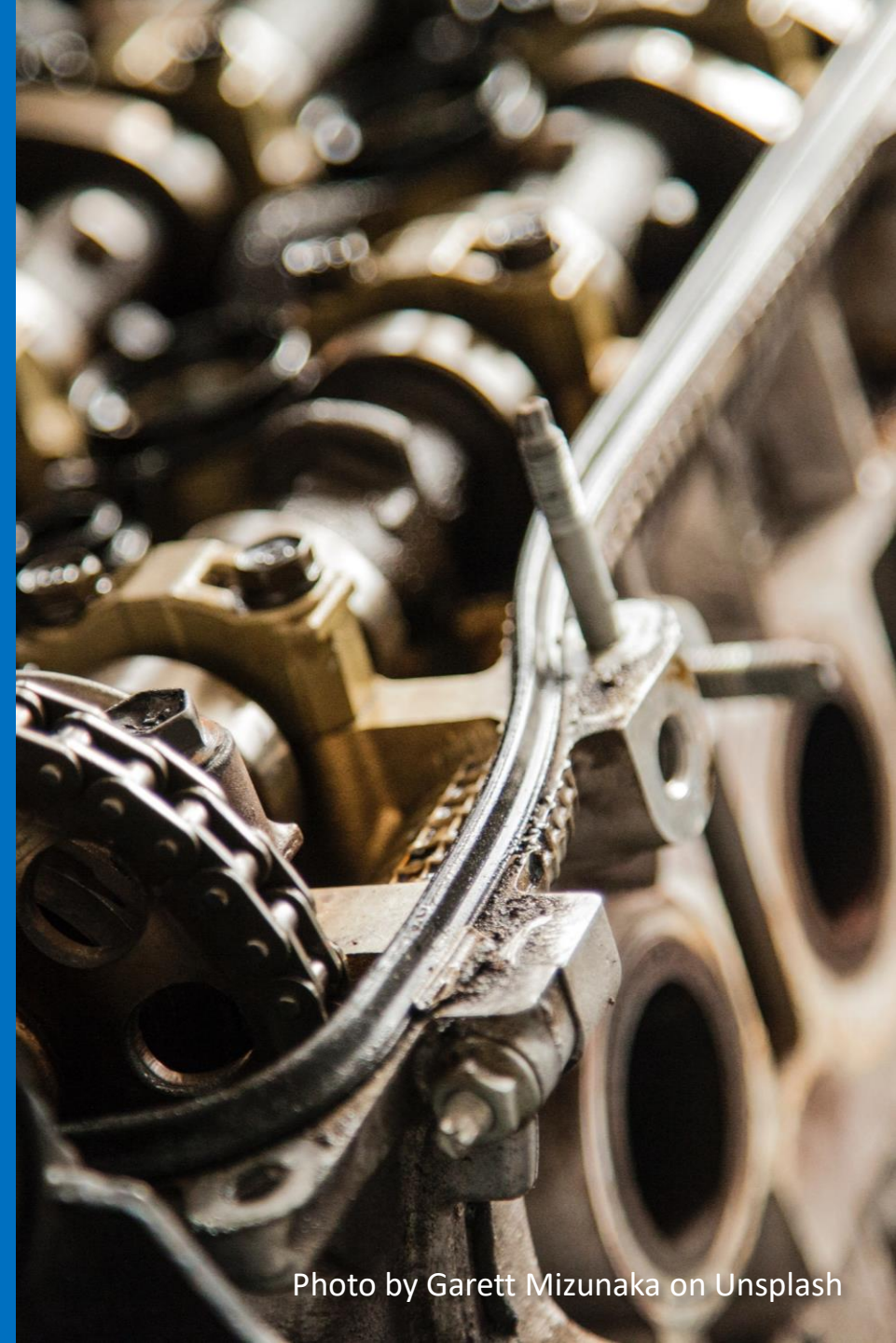


Photo by Garrett Mizunaka on Unsplash



Don't use Aliases

ls > Get-ChildItem

cd > Set-Location

cat > Get-Content

cp > Copy-Item





Code Review

```
function Start_DeathStar {
    Param ([string]$param1)
    if(!$param1){
    Write-host "You didn't provide a parameter"; Return}
    if ($param1 -eq "Alderaan")
    {
        TurnON -param1 'Alderaan'
    } else {
        turnOFF }
    }

function TurnON{Param ()
    $true
}

function turnOFF{
    #this is False
    $false}
```

```
# Help removed for brevity
function Start-DeathStar {
    [CmdletBinding()]
    Param (
        [Parameter(Mandatory)]
        [string]
        $Target
    )
    if ($Target -eq 'Alderaan') {
        Write-Verbose 'Targetting Alderaan'
        Enable-PlanetBlowerUpperGun
    }
    else {
        Write-Verbose 'Unknown target'
        Disable-PlanetBlowerUpperGun
    }
}
# Help removed for brevity
function Enable-PlanetBlowerUpperGun {
    [CmdletBinding()]
    Param ()

    Write-Verbose 'Enabled gun.'
}
# Help removed for brevity
function Disable-PlanetBlowerUpperGun {
    [CmdletBinding()]
    Param ()

    Write-Verbose 'Disabled gun.'
}
```



Consistent Standard

Develop a
consistent standard

Respect other standards

Work with Project /Organisation

Evolve your own standard

Read and learn from others

Contribute to Open Source



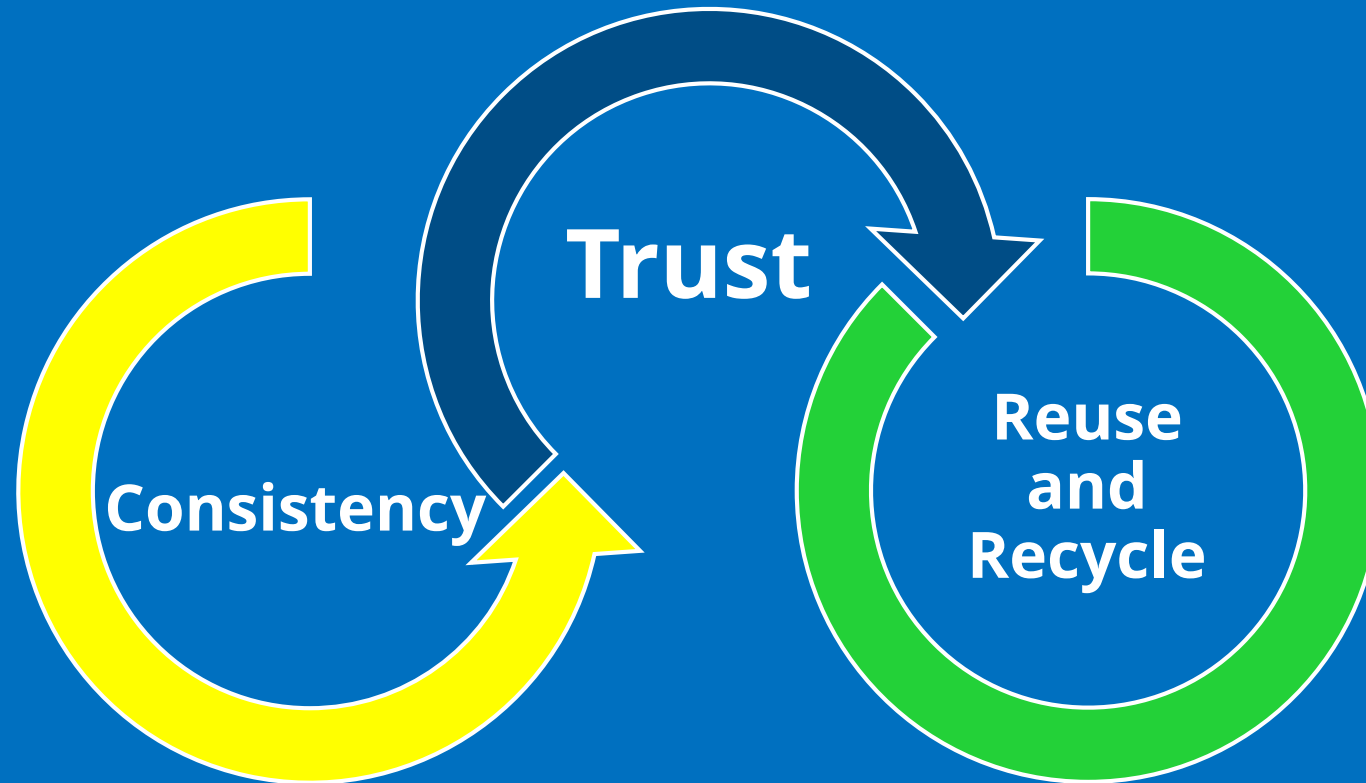
PowerShell
Standards
Agency

Get Involved in Open Source





Consistency > Trust > Reuse



Consistency breeds Trust leads to Reuse and Recycle



PowerShell
Standards
Agency

Thank You!

Questions?

#PowerShellStandardsAgency





Resources

pau.by/psa

[#PowerShellStandardsAgency](https://twitter.com/PowerShellStandardsAgency)