The Best PowerShell Cheat Sheet

lazyadmin.nl/powershell/powershell-cheat-sheet

August 23, 2024

Looking for a good PowerShell cheat sheet? Then you are in the right place. I have been working with PowerShell for the last 10 years. During that time I have written down the most useful cmdlets, operators, or useful commands in many places, just to remind them.

But I have now spent some time on making the best PowerShell Cheat Sheet. This one is not only for beginners, but also advanced users will still find great value in it.

In this cheat sheet, you will find all the operators, tips on working with variables, flow control statements (if-else, loops, etc), and collections and hashtables. I have also added the new PowerShell 7 Ternary operators and the Null-coalescing operators.

PowerShell Cheat Sheet Download

To your copy of the cheat sheet, just fill in the form below and you will receive a PDF version that you can printout in your mailbox.

Where can I send the Free PowerShell cheat sheet to?

Most commands are also explained on this website, just use the search function in the top right corner to quickly look up a particular command. I will also briefly explain the different items in this article for your reference.

Good to know

The first section on the sheet contains some good-to-know commands. One of my most used commands is the display all parameters feature. You can do this by pressing `Ctrl +Space` after you have typed a cmdlet.

To make it even easier, I recommend adding the following command to your PowerShell Profile. This way you only have to press Tab twice to view all possible parameters.

```
Set-PSReadlineKeyHandler -Key Tab -Function Complete
```

Also the `Get-History` cmdlet beats pressing the Up Arrow key to find the command that you used a couple of minutes ago. Make sure you try it out.

Operators

PowerShell has a lot of operators, just like any other programming or scripting language, that you can use. On the sheet, you will find the most commonly used ones for your convenience. I have grouped the counterparts of the operators together to save some space.

Operator	Description
`$a += 5` *or* `$a -= 5`	Add and assign or Subtract and assign
`$a *= 2` *or* `$a /= 2`	Multiply and assign or Divide and assign
`2 -eq $a` *or* `2 -ne $a`	Greater than or Equal (variable on the right side)
`$a -gt 2` *or* `$a -lt 2`	Greater than or Less than
`2 -ge $a`	Less than or Equal (variable on the right side)
`2 -le $a`	Less than or Equal (variable on right side)

PowerShell Operators

I have written a complete article about PowerShell operators, make sure that you check it if you want to know more.

Variables & Objects

Assigning variables is pretty straightforward, but in PowerShell, there are some convenient methods to quickly create a range, or assign multiple variables. Also, the scopes in PowerShell can be useful in some occasions. But keep in mind, don't use the global scope for everything, there is often a better way.

Command	Description
`$a = "Hello"`	Assign a value to a variable
`$a, $b = "Hello","Bye"`	Multiple variable assignment
`$range = 1..10`	Create an array with a sequence
`$_`	Get current pipeline object
`$null`	Null value
`[type]$var`	Declared typed (int, string) variable
`$global:var`	Assign variable in global scope

PowerShell Variables & Objects

Flow Control Statements

Flow control statements allow you to control what your script should do next based on certain condition(s). The most commonly known flow control statement is of course the if-else statement.

Also, the For and ForEach loops are commonly used and often well-known. But less known are the new Ternary operator and the Null-coalescing operators that are available in PowerShell 7.

Statement	Description
`if ($condition) { } else { }`	Simple If-Else statement
`($x -gt 10) ? "High" : "Low"`	Ternary operator for if-else **(PS7)**
`switch ($var) { }`	Switch statement
`for ($i=0; $i -lt 10; $i++) { }`	Standard for loop
`foreach ($item in $collection) {}`	Loop to iterate collections
`while ($condition) { }`	While loop with a condition
`$result = $value ?? "Default"`	Assigns default if value is null **(PS7)**
`$value ??= DefaultValue`	Assigns if null **(PS7)**
`$result = ${object}?.Property`	Access property if object exists **(PS7)**
`$element = ${array}?[index]`	Access element if array exists**(PS7)**

Flow Control Statements

Collections & Hashtables

When working with datasets you can't really do without an array, hashtable, or object. In the PowerShell cheat sheet, you will find everything you need to create, assign, or retrieve information from one of the collection items.

Did you know that hashtables are also particularly useful when it comes to making your code more readable? With splatting, we can group the parameters of cmdlets in a nice table before assigning them. This makes it a lot easier to edit the value of the parameters, without the need for horizontal scrolling.

Command	Description
`$array = @('item1', 'item2', 'item3')`	Create array with value
`$array[index]`	Access array element
`$array.Length`	Get array length
`$array.Add(item)`	Add item to array
`$array.Remove(item)`	Remove item from array

`$hash = @{`
`key1 = 'value1';`
`key2 = 'value2';`
`}`
	Create hash table

`$hash.key1`	Access hash key
`$hash.key2 = 'new value'`	Assign value to hash key
`$hash.Add('key3', 'value')`	Add key-value pair
`$hash.Remove('key2')`	Remove key-value pair

`$obj = [PSCustomObject]@{`
`Prop1 = 'Val1';`
`Prop2 = 'Val2'`
`}`
	Custom object

PowerShell Collections & Hashtables

Input/Output

PowerShell is also great for exporting data out of systems (for example Microsoft 365) and generating reports. The basis cmdlets that you will need to know for this are listed in the sheet. But if you want to know more, then make sure you also check out the PowerShell Excel Module.

Command	Description
`$var = Read-Host`	Read user input
`Get-Content`	Read file content
`Import-Csv`	Import from CSV file
`Export-Csv`	Export to CSV file
`Write-Host "Hello"`	Write to console
`Write-Host "$($obj.test)"`	Write to console with obj property
`$var | Out-GridView`	Output to interactive grid view

Input/Output

Running Scripts & Processing

Running a PowerShell script is not that difficult. But do you know how to run a script in the current scope? Or as a background task? Most don't use these methods daily, but can come in very handy, so I have listed them on the cheat sheet for your reference.

Command	Description
`Start-Process notepad`	Start external process or application
`Start-ThreadJob -ScriptBlock { Get-Process }`	Run script or task in the background
`. .\script.ps1`	Run a script in the current scope (dot-sourcing)
`& .\script.ps1`	Run script in a new scope (call operator)

Running Scripts & Processing

Pipeline and Formatting

The pipeline character in PowerShell allows you to pass results for the left-hand side to another cmdlet or scriptblock on the right-hand side. This way you don't have to "store" results in a variable, and create loops for everything.

Most experienced PowerShell users know these methods, but when you are pretty new to PowerShell it's great to quickly check the correct way using the pipeline method.

Command	Description
`Get-Process | Sort-Object -Property Name`	Sort processes by name
`Get-ChildItem *.txt | Where-Object {$_.Length -gt 1KB}`	Filter text files larger than 1KB
`Get-Process | Select-Object -Property Name, ID, CPU`	Select process name, ID, and CPU usage
`Get-Process | ForEach-Object {$_.ProcessName}`	Process each item in the pipeline
`Get-ChildItem | Format-List -Property Name, Length`	Display file name and length in a list
`Get-Process | Format-Table -Property ID, CPU -AutoSize`	Display processes in a table with auto-sized columns

Pipeline and Formatting

Wrapping Up

Make sure that you download this PowerShell Cheat Sheet and print it out so you can quickly check it. If you like the sheet, make sure that you share it with your colleagues. If you don't like a printed version, then you can always bookmark this page, which contains the same information.

If you have any questions or know a command that really needs to be on the sheet, then just drop a comment below.