PowerShell クイックリファレンス

コマンドレット、パイプライン、オブジェクト、スクリプト、モジュール

基本

コマンドとヘルプ

```
Get-Help Get-Process         # show help for cmdlet
Get-Help Get-Process -Online # open online docs
Get-Command *service*        # find commands by name
Get-Alias ls                 # show alias target
```

よく使うエイリアス

ls → Get-ChildItem	ファイルとディレクトリを一覧表示
cd → Set-Location	ディレクトリを変更
cp → Copy-Item	ファイルまたはディレクトリをコピー
mv → Move-Item	移動またはリネーム
rm → Remove-Item	ファイルまたはディレクトリを削除
cat → Get-Content	ファイルの内容を読み取り
echo → Write-Output	パイプラインに出力
cls → Clear-Host	コンソールをクリア

変数

変数の基本

```
$name = "Alice"          # string
$count = 42              # integer
$pi = 3.14               # double
$list = @(1, 2, 3)       # array
$hash = @{a=1; b=2}      # hashtable
```

自動変数

$_	現在のパイプラインオブジェクト
$PSVersionTable	PowerShell のバージョン情報
$HOME	ユーザーのホームディレクトリ
$PWD	現在のディレクトリ
$null	Null 値
$true / $false	真偽値定数
$Error	最近のエラーの配列
$LASTEXITCODE	最後のネイティブコマンドの終了コード

環境変数

```
$env:PATH                    # read env var
$env:MY_VAR = "value"        # set env var
Get-ChildItem Env:           # list all env vars
```

演算子

比較演算子

-eq -ne	等しい / 等しくない
-gt -lt	より大きい / より小さい
-ge -le	以上 / 以下
-like -notlike	ワイルドカードマッチ（*****、**?**）
-match -notmatch	正規表現マッチ
-contains	コレクションが値を含む
-in -notin	コレクション内の値

論理演算子とその他

-and -or -not	論理演算子
!	論理 NOT（エイリアス）
-replace	正規表現置換：**'hi' -replace 'h','b'**
-split -join	文字列を分割 / 結合
..	範囲：**1..5** → 1,2,3,4,5
?:	三項演算子（v7+）：**$x ? 'yes' : 'no'**

制御フロー

if / elseif / else

```
if ($age -ge 18) {
    "Adult"
} elseif ($age -ge 13) {
    "Teen"
} else {
    "Child"
}
```

switch

```
switch ($color) {
    "red"   { "Stop" }
    "green" { "Go" }
    default { "Unknown" }
}
```

ループ

```
foreach ($item in $list) { $item }
for ($i=0; $i -lt 5; $i++) { $i }
while ($x -lt 10) { $x++ }
1..5 | ForEach-Object { $_ * 2 }
```

関数

関数の定義

```
function Get-Greeting {
    param([string]$Name = "World")
    "Hello, $Name!"
}
Get-Greeting -Name "Alice"
```

高度なパラメータ

```
function Copy-SafeFile {
    [CmdletBinding()]
    param(
        [Parameter(Mandatory)][string]$Path,
        [Parameter(Mandatory)][string]$Dest
    )
    Copy-Item $Path $Dest -WhatIf:$WhatIfPreference
}
```

オブジェクトとパイプライン

パイプラインの基本

```
Get-Process | Sort-Object CPU -Desc | Select-Object -First 5
Get-Service | Where-Object Status -eq "Running"
Get-ChildItem | Measure-Object -Property Length -Sum
```

主要なパイプラインコマンドレット

Where-Object	オブジェクトをフィルタリング：**| Where {$_.CPU -gt 10}**
Select-Object	プロパティを選択：**| Select Name, CPU**
Sort-Object	ソート：**| Sort CPU -Desc**
ForEach-Object	各要素を変換：**| ForEach {$_.Name}**
Measure-Object	カウント、合計、平均、最小、最大
Group-Object	プロパティ値でグループ化
Format-Table	テーブルとして表示：**| ft -Auto**
Export-Csv	CSV にエクスポート：**| Export-Csv out.csv**

ファイル

ファイル操作

```
Get-Content log.txt                    # read file
Set-Content out.txt "hello"            # write (overwrite)
Add-Content out.txt "more"             # append
Get-Content log.txt | Select-String "error"  # grep
```

パスとファイルのコマンドレット

Test-Path $path	ファイル/ディレクトリの存在確認
New-Item -Type File	ファイルを作成
New-Item -Type Directory	ディレクトリを作成
Resolve-Path	絶対パスを取得
Join-Path	パスセグメントを結合
Split-Path	親またはリーフを取得
Get-ItemProperty	ファイルの属性とメタデータ
Remove-Item -Recurse	ディレクトリを再帰的に削除

文字列

文字列の基本

```
"Hello, $name"               # interpolation (double quotes)
'Hello, $name'               # literal (single quotes)
"Length: $($list.Count)"      # expression in string
@"
Multi-line
here-string with $name
"@
```

文字列メソッド

.ToUpper() / .ToLower()	大文字/小文字に変換
.Trim()	前後の空白を削除
.Split(',')	配列に分割
.Replace('a','b')	部分文字列を置換
.StartsWith() / .EndsWith()	プレフィックス / サフィックスを確認
.Substring(0,5)	部分文字列を抽出
.Contains('text')	含むか確認
-f operator	**'{0} is {1}' -f 'sky','blue'**

モジュール

モジュールの管理

```
Get-Module -ListAvailable          # installed modules
Find-Module -Name Az*              # search gallery
Install-Module -Name Pester        # install from gallery
Import-Module ActiveDirectory      # load module
```

モジュールコマンド

Get-Module	ロード済みモジュールを一覧表示
Import-Module	セッションにモジュールをロード
Remove-Module	セッションからモジュールをアンロード
Update-Module	インストール済みモジュールを更新
Get-Command -Module X	モジュール内のコマンドを一覧表示
$env:PSModulePath	モジュールの検索パス

PowerShell クイックリファレンス

よくあるパターン

エラー処理

```
try {
    Get-Item "C:\missing" -ErrorAction Stop
} catch {
    "Error: $_"
} finally {
    "Cleanup here"
}
```

実行ポリシーとリモート処理

Get-ExecutionPolicy	現在のスクリプトポリシーを表示
Set-ExecutionPolicy RemoteSigned	ローカルスクリプトを許可
Enter-PSSession -Computer SRV1	インタラクティブなリモートセッション
Invoke-Command -Computer SRV1 -Script {}	スクリプトブロックをリモートで実行
Start-Job { long-task }	バックグラウンドジョブを実行
Receive-Job -Id 1	バックグラウンドジョブの出力を取得