Downloads and installs newest help files
Changes the specified
Array subexpression operator
Returns TRUE if the scalar
Equal
The call operator
Type evaluator
Joins multiple strings
Range operator
Bitwise OR
Formats strings by using the
Splits a string
Bitwise OR (inclusive)
Bitwise OR (exclusive)
Bitwise NOT
Bitwise shift operators. Bit shift left, bit shift right
(arithmetic for signed, logical for unsigned values)

Other Operators
-Split Splits a string
“abcdefgh” -split “de”
-Join Joins multiple strings
“abc”, “def”, “ghi” -join “,”

Range operator
1..10 | foreach {$_ * 5}

42 -Is [int]
-Type convertor. Tries to convert the input object to the specified .NET Framework type.
$A = 42 -as [String]

-Is, -Isnot Type evaluator (Boolean). Tells whether an object is an instance of a specified .NET Framework type.

Logical Operators
-And, -Or, -Xor, -Not, ! Connect expressions and statements, allowing you to test for multiple conditions

Redirection Operators
>, >> The redirection operators enable you to send particular types of output (success, error, warning, verbose, and debug) to files and to the success output stream.

Output Streams
* All output
  1 Success output
  2 Errors
  3 Warning messages
  4 Verbose output
  5 Debug messages

# Writes warning output to warning.txt
Do-Something 3> warning.txt
# Appends verbose.txt with the verbose output
Do-Something 4>> verbose.txt
# Writes debug output to the output stream
Do-Something 5&1
# Redirects all streams to out.txt
Do-Something *> out.txt

Comma operator (Array constructor)
Dot-sourcing operator runs a script in the current scope
.c:\scripts\sample.ps1
Subexpression operator
$( ) Array subexpression operator
@() The call operator, also known as the “invocation operator,” lets you run commands that are stored in variables and represented by strings.

$a = "Get-Process" & $a
$sb = { Get-Process | Select -First 2 } & $sb

Windows PowerShell 3.0 Language Quick Reference
Created by http://powershellmagazine.com
### Arrays

- "a", "b", "c" Array of strings
- 1,2,3 Array of integers
- @() Empty array
- @([2]) Array of one element
- 1,2,3 Array within array
- "hi" Array of one element
- @([5]) Sixth element of array*
- @([2..20]) Returns elements 3 thru 21
- @([-1]) Returns the last array element
- @([-3..-1]) Displays the last three elements of the array
- @([1..4..6..9]) Displays the elements at index positions 1, 4, and 6 through 9
- @(Get-Process) Forces the result to an array using the array sub-expression operator
- $arr=1..10
- $arr=([System.Array](1, 2, 3)).Reverse() Reverses an array
- $arr[1] += 200 Adds to an existing value of the second array item (increases the value of the element)
- $b = $arr[0, 1 + 3..6] Creates a new array based on selected elements of an existing array
- $z = $arr + $b Combines two arrays into a single array, use the plus operator (+)

*Arrays are zero-based

### Associative Arrays (Hash tables)

- $hash = @{} Creates empty hash table
- @{$foo=1; bar='value2'} Creates and initialize a hash table
- [ordered] @{$a=1; b=2; c=3} Creates an ordered dictionary
- $hash = @{} Assigns 1 to key key1
- $hash = @{} Adds to an existing value of key1
- $hash["key1"] Returns value of key1
- $hash.GetEnumerator | sort Key Sorts a hash table by the Key property
- [pscustomobject] @{$x=1; y=2} Creates a custom object

### Comments

- # This is a comment because # is the first character of a token
- $a = "#This is not a comment..."
- $a = "something" # ...but this is.
- Write-Host Hello#world

### Block Comments

- <# This is a multi-line comment #>

### Object Properties

- An object’s properties can be referenced directly with the "." operator.
- $a = Get-Date
- $a | Get-Member -MemberType Property
- $a = Get-Member -MemberType Method
- $a.Date
- $a.TimeOfDay.Hours
- $a | Get-Member -MemberType Property
- $a | Get-Member -MemberType Method

### Methods

- Methods can be called on objects.

### Strings

- "This is a string, this $variable is expanded as is $(2+2)"
- "This is a string, this $variable is not expanded"
- @ This is a here-string which can contain anything including carriage returns and quotes. Expressions are evaluated: $(2+2*5). Note that the end marker of the here-string must be at the beginning of a line!
- @ This is a here-string which can contain anything including carriage returns and quotes. Expressions are evaluated: $(2+2*5)

### Variables

- Format: ${scope:name} or ${anyname} or ${any path}

- $path = "C:\Windows\System32"
- Get-ChildItem $env:ProgramFiles(x86) $processes = Get-Process

- $global:a = 1 # visible everywhere
- $local:a = 1 # defined in this scope and visible to children
- $private:a = 1 # same as local but invisible to child scopes
- $script:a = 1 # visible to everything is this script

# Using scope indicates a local variable in remote commands and with Start-Job

- $localVar = Read-Host "Directory, please"
- Invoke-Command -ComputerName localhost -ScriptBlock { dir $using:localVar }
- Start-Job { dir $using:localVar -Recurse}
- $env:Path += ";D:\Scripts"
Get-Command -Noun Variable # the Variable Cmdlets
Get-ChildItem variable: # listing all variables using the variable drive

# strongly-typed variable (can contain only integers)
[int]$number=8

# attributes can be used on variables
[ValidateRange(1,10)]$number = 1

# flip variables
$a=1;$b=2
$a,$b = $b,$a

# multi assignment
$a,$b,$c = 0
$a,$b,$c = 'a','b','c'
a,$b,$c = 'a b c'.split()

# create read only variable (can be overwitten with -Force)
Set-Variable -Name ReadOnlyVar -Value 3 -Option ReadOnly

# create Constant variable (cannot be overwitten)
Set-Variable -Name Pi -Value 3.14 -Option Constant

<table>
<thead>
<tr>
<th>Windows PowerShell Preference Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$ConfirmPreference</strong></td>
</tr>
<tr>
<td>Determines whether Windows PowerShell automatically prompts you for confirmation before running a cmdlet or function</td>
</tr>
<tr>
<td><strong>$DebugPreference</strong></td>
</tr>
<tr>
<td>Determines how Windows PowerShell responds to debugging</td>
</tr>
<tr>
<td><strong>$ErrorActionPreference</strong></td>
</tr>
<tr>
<td>Determines how Windows PowerShell responds to a non-terminating error</td>
</tr>
<tr>
<td><strong>$ErrorView</strong></td>
</tr>
<tr>
<td>Determines the display format of error messages in Windows PowerShell</td>
</tr>
<tr>
<td><strong>$FormatEnumerationLimit</strong></td>
</tr>
<tr>
<td>Determines how many enumerated items are included in a display</td>
</tr>
<tr>
<td><strong>$MaximumHistoryCount</strong></td>
</tr>
<tr>
<td>Determines how many commands are saved in the command history for the current session</td>
</tr>
<tr>
<td><strong>$OFS</strong></td>
</tr>
<tr>
<td>Output Field Separator. Specifies the character that separates the elements of an array when the array is converted to a string. The default value is: Space.</td>
</tr>
<tr>
<td><strong>$OutputEncoding</strong></td>
</tr>
<tr>
<td>Determines the character encoding method that Windows PowerShell uses when it sends text to other applications</td>
</tr>
<tr>
<td><strong>$PSDefaultParameterValues</strong></td>
</tr>
<tr>
<td>Specifies default values for the parameters of cmdlets and advanced functions</td>
</tr>
<tr>
<td><strong>$PSEmailServer</strong></td>
</tr>
<tr>
<td>Specifies the default e-mail server that is used to send e-mail messages</td>
</tr>
<tr>
<td><strong>$PSModuleAutoLoadingPreference</strong></td>
</tr>
<tr>
<td>Enables and disables automatic importing of modules in the session. &quot;All&quot; is the default.</td>
</tr>
<tr>
<td><strong>$PSSessionApplicationName</strong></td>
</tr>
<tr>
<td>Specifies the default application name for a remote command that uses WS-Management technology</td>
</tr>
<tr>
<td><strong>$PSSessionConfigurationName</strong></td>
</tr>
<tr>
<td>Specifies the default session configuration that is used for PSessions created in the current session</td>
</tr>
<tr>
<td><strong>$PSSessionOption</strong></td>
</tr>
<tr>
<td>Establishes the default values for advanced user options in a remote session</td>
</tr>
<tr>
<td><strong>$VerbosePreference</strong></td>
</tr>
<tr>
<td>Determines how Windows PowerShell responds to verbose messages generated by a script, cmdlet or provider</td>
</tr>
<tr>
<td><strong>$WarningPreference</strong></td>
</tr>
<tr>
<td>Determines how Windows PowerShell responds to warning messages generated by a script, cmdlet or provider</td>
</tr>
<tr>
<td><strong>$WhatIfPreference</strong></td>
</tr>
<tr>
<td>Determines whether WhatIf is automatically enabled for every command that supports it</td>
</tr>
</tbody>
</table>

**Windows PowerShell Automatic Variables**

- `$` Last token of the previous command line
- `?` Boolean status of last command
- `^` First token of the previous command line
- `$_` $PSItem Current pipeline object
- `$Args` Arguments to a script or function
- `$Error` Array of errors from previous commands
- `$ForEach` Reference to the enumerator in a foreach loop
- `$Home` The user’s home directory
- `$Host` Reference to the application hosting the POWERSHIELD language
- `$Input` Enumerator of objects piped to a script
- `$LastExitCode` Exit code of last program or script
- `$Matches` Exit code of last program or script
- `$MyInvocation` An object with information about the current command
- `$PSHome` The installation location of Windows PowerShell
- `$profile` The standard profile (may not be present)
- `$Switch` Enumerator in a switch statement
- `$True` Boolean value for TRUE
- `$False` Boolean value for FALSE
- `$SCulture` Current culture
- `$SUCulture` Current UI culture
- `$PsVersionTable` Details about the version of Windows PowerShell
- `$PWD` The full path of the current directory
- `$PSVersionTable` Details about the version of Windows PowerShell
# Windows PowerShell Learning Resources

## Microsoft Resources

- **Microsoft Windows PowerShell**
  - http://www.microsoft.com/powershell
- **Windows PowerShell Team Blog**
  - http://blogs.msdn.com/PowerShell
- **MS TechNet Script Center**
- **PowerShell Forum**
- **Hey, Scripting Guy! Blog**
- **Windows PowerShell Survival Guide**

## Community Resources

- **PowerShell Community**
  - http://powershellcommunity.org
- **PowerShell Code Repository**
  - http://poshcode.org
- **PowerShell.com Community**
  - http://powershell.com
- **PowerGUI.org Community**
  - http://powergui.org
- **PowerShell Community Groups**
  - http://powershellgroup.org
- **PowerShell Magazine**
  - http://powershellmagazine.com

## Free eBooks and Guides

- **Mastering PowerShell, Second Edition - Dr. Tobias Weltner**
- **Secrets of PowerShell Remoting - Don Jones and Dr. Tobias Weltner**
  - http://powershellbooks.com
- **Administrator's Guide to Windows PowerShell Remoting**
  - Dr. Tobias Weltner, Aleksandar Nikolic, Richard Giles
- **Layman's Guide to PowerShell 2.0 Remoting - Ravikanth Chaganti**
  - http://www.ravichaganti.com/blog/?page_id=1301
- **WMI Query Language via PowerShell - Ravikanth Chaganti**
  - http://www.ravichaganti.com/blog/?page_id=2134
- **PowerShell 2.0 One Cmdlet at a Time - Jonathan Medd**
- **Effective Windows PowerShell - Keith Hill**
  - http://rkeithhill.wordpress.com/2009/03/08/effective-windows-powershell-the-free-ebook/

## Books

- Don Jones, Learn Windows PowerShell in a Month of Lunches
- Bruce Payette, Windows PowerShell in Action, Second Edition