1 Add comments when possible, especially if it the code is complex or a "clever trick".

2 Use the #Requires tag for the minimum version of Windows PowerShell required, required Modules, PSSnapin or Administrative rights.

2 Always validate input

4 Consistent casing, should be obvious, but windows in not case sensitive, so WE need to be.

5 Place user-defined variables at the top of script

6 Use local variables. The golden rule for programming avoid globals when you can!

7 Use multi-line, indented code blocks (parenthesis) in IF, FOR , switch, etc...

ie:

IF "%~1"=="/?" (

...

)

ELSE (

...

)

9 Give each command its own dedicated line, make it easier to read and interpret what is going on.

9a Avoid "one-liners" or at least be aware that they imply a code block

10 Specify extensions for external commands, like .exe or .bat

10a When using external commands, check their availability and version

11 Don’t use aliases in scripts. Use the full cmdlet name with its named parameters.

11a If possible, specify the full paths for external commands

12 use variables name that are simple but meaningful.

Don’t turn off pipeline errors ($ErrorActionPreference = “SilentlyContinue”).

Implement structured error handling by using Try+Catch+Finally or Trap to handle errors.

Avoid using backticks, they are easy to miss since they might look like dirt on the screen. Instead, use the pipeline (|) character where appropriate, or even splatting the parameters to the cmdlet.

indentation

Use Advanced Functions.

Use the Verb-Noun naming convention for your functions and filters. When picking out verbs, always use standard verbs. Use Get-Verb to see what verbs are available.

Do not use the Return keyword. Functions automatically return output to the calling process through the pipeline.

Use Test-Connection (with the -Quiet switch) to ensure a computer is online prior to connecting to it.

Make sure your script is lined up, indented properly, and easy to read. If you can read and understand your script, you will simplify your debugging process.

Keep your script independent from the place you run it (Use $myInvocation.MyCommand.Path or $PSScriptRoot for PowerShell 3.0 and above).

Make sure you test your functions and scripts in a clean environment with no dependencies on profiles, admin rights, and versions to ensure compatibility.

Test scripts in test environment before they are released to production.

Re-use scripts (Package modules, create a script repository).

Never use Write-Host

In a script, always use full parameter names.

Filter on the left. It is more efficient to filter returning data as close to the source of data as possible. For example, you do not want to return the entire contents of the system event log across the network to your work station, and then filter events for a specific event ID. Instead, you want to filter the system event log on the server, and then return the data.

. Pipe to the right. When writing code in the Windows PowerShell ISE, you want to format the code so that it is easy to read. This means avoiding really really long lines of code. The best way to break up your lines of code into readable chunks is to break on the right-hand side. The following code illustrates this.

PowerShell function should be used anytime that you need to write repeatable code

Name function appropraitly

it is a good idea to document within the script (through the use of comments) which external functions are used, and which version of those functions the script is designed to work with.