Best practices Richard Siddeway Powershell in depth

* [PowerShell](https://habr.com/hub/powershell/" \o "Вы не подписаны на этот хаб)

**перевод Powershell in depth. глава 40.**  
  
в этой главе содержатся:  
лучшее из практики для всех случаев  
лучшее из практики для продвинутых скриптов  
лучшее из практики для промышленного использования  
  
Во всей этой книге давались полезные советы, практические трюки как облегчить ваши усилия, облегчить поддержку скриптов, ускорение исполнения, достижение большей гибкости. В этой главе мы мы собрали наши рекомендации и организовали для удобного использования. Лучшая наша рекомендация это использовать PowerShell каждый день. Приведенные ниже рекомендации не идут в порядке важности.  
  
**40.1 PowerShell general best practices**  
эти рекомендации даны вам чтобы стать лучше в powershell

* Читайте справочную систему, она содержит массу полезной информации, в особенности примеры.
* Используя powershell 3 сделайте задачу в планировщике для регулярного обновления справки
* Установите политику исполнения скриптов как минимум *RemoteSigned*
* Используйте конвеер. PowerShell спроектирован под использование конвеером. Если вы будете кодировать в стиле старых скриптовых языков вы потеряете большую часть функциональности и создадите себе самому много работы.
* Давайте переменным осмысленные имена, например *$computer* лучше чем *$x*
* Избегайте имен с пробелами или специальными знаками например *${computer}*
* Никогда не устанавливайте переменную *$ErrorActionPreference* (или *$VerbosePreference* или любую другую «preference» переменную) глобально в оболочке, скрипте или функции. Вместо этого используйте параметры, такие как *-ErrorAction*, или для функции параметр *-Verbose*, для установки параметра в том месте где требуется.
* Избегайте применять перебор коллекций — использование *ForEach-Object* или *ForEach*, кроме тех случаев, когда нет больше способа по другому достичь цели. (PowerShell не любит конструкций типа For)
* Используйте в основном одинарные ковычки, если вам не нужно делать подстановку переменной или вычислять выражения внутри строки. Если вы работаетет с SQL запросами то помните что одинарные ковычки используются им для строк и вам нужно специально делать изоляцию или применять двойные ковычки.
* Замена строк, или мультипликация намного проще чем конкатенация строк.
* Используйте встроенные константы, PowerShell понимает KB, MB, GB, TB и PB (встроенные переменные)
* Избегайте использования нативных .NET классов и методов, за исключением случаев когда нет другой альтернативы
* Будте осторожны с кодом скачиваемым с интернета, всегда дважды проверяйте что код делает. Ваша среда может быть другой чем среда автора и может вызвать в вашей среде много проблем.
* Фильтруй рано, форматируй поздно! Делай выборку данных как можно раньше сокращая их объем, но форматируй данные как можно позже, лучше всего непосредственно перед выводом на дисплей.

**40.2 лучшее из практики для продвинутых скриптов**

* Эти рекомендации созданы для написания лучших скриптов
* Указывай явно тип переменной, пиши *[string]$logfile*
* Присваивай переменным явное значение после создания, если эта переменная в этом скопе. (проще — инициализируй всегда)
* Давай имена функциям и потокам в стиле гагол-действие например*Get-DiskInfo*
* Когда скрипт выполняющий задачу начинает использоваться до того как будет оформлен в виде отдельной функции, давайте файлу скрипта имя в стиле командлетов, например *Set-UserAttributes.ps1*
* Давая имя функции или скрипту в виде глагол-существительное добавь две или три буквы префикса к существительному. Обычно это сокращение от названия вашей организации. Например для компании *«great things»* это может быть *GT*, тогда имя функции могло бы выглядеть как *Get-GTUserInfo*или скрипт мог бы называться *Set-GTUserInfo.ps1*
* Если вы создаете приватные (не экспортируемые) переменные в модуле, дайвайте этим переменным отчетливые имена. Множество разработчики используют знак подчеркивания чтобы выделить эти имена, например *$\_private*или *$\_counter*.
* Когда определяешь параметры скрипта или функции, используй имена параметра совпадающие с именами параметров стандартных командлетов исполняющих теже функции. Например имя параметра *-Computername* предпочтительнее чем *-host* или *-machines,* т.к. является стандартным для командлетов. Вы всегда можете определить алиас если вам так нравится.
* Избегай использовать Write-Host кроме случаев когда нужно вывести сообщение именно на экран. Если вы используете Write-Host, используйте выделение цветом или выделение цветом заднего фона так ваше сообщение будет выделено на экране.
* Используйте*[CmdletBinding]*в ваших скриптах и функциях чтобы использовать verbose, debugging и продвинутый функционал.
* Если ваша функция производит изменения в системе, добавьте поддержку *-WhatIf*и *-Confirm*
* Используйте *Write-Verbose*для вывода прогресса исполнения, чтобы можно было использовать этот параметр для отслеживания в каком месте идет выполнение вашей функции, или чем она занимается в сейчас.
* Используйте *Write-Debug* для вывода сообщений необходимых для более успешной отладки, помните, что *Write-Debug*останавливает исполнение скрипта и дает возможность приостановить его работу.
* Помните что *Write-Warning*существует для тех случаев когда вам нужно вывести информационное сообщение на экран
* **Скрипты и функции должны производить один, и только один тип данных на выходе.**Обычно это объект и он может быть специально сконструированным видом объекта объединияющим в себе свойства множества разных объектов.
* Всегда пишите справку для ваших скриптов и функций, даже если это справка основанная на комментариях. XML справка обычно используется если вам нужно делать справку на разных языках.
* Избегайте изменять алиасы, переменные, и другие элементы другой области видимости чем ваша область. (заумно получилось. проще — не меняйте ничего в других скопах, всячески избегайте этого иначе получите код плохо поддающийся рефакторингу и отладке)
* *Разделяйте задачу на четкие, небольшие куски по своей функциональности и выделите каждый в виде функции.* Пример, скрипт выполняющий 10 разных вещей должен быть разбит на 10 разных функций, и скрипт должен делать вызовы этих функций в нужной последовательности.
* Подписывате примеры кода что вы планируете распространять вместе со скриптом своим сертификатом, если вы планируете выкладывать это в паблик. Да это требует наличия у вас сертификата для подписи кода, для этого придется гдето его получить. Но это хороший способ для паблика проверить что ваш код не был подправлен кем либо.
* Избегайте использовать венгерскую нотацию в именах переменных. Соглашения типа *$strName*или *$intCounter*для powershell устарели и не имеют смысла. (проще — не используй именование из старой школы, называй просто. В PowerShell нет типов как таковых, нет нужды в венгерской нотации)
* Используйте отступы для блоков кода таких как { ваш код здесь } в контрукциях, ваш код станет читаемым
* Используйте *Write-Verbose*, *Write-Debug* и тому подобные вставки для документирования скриптов и функций, используйте эти конструкции вместо вставки комментариев для этих же целей.
* В скриптах, функциях или *workflow* избегайте алиасов (за исключением общепринятых например *Dir*) и сокращений имен параметров. Пишите полностью имена параметров и командлетов для лучшего понимания и читаемости.
* Избегайте использования символа переноса (`) в конце строки для переноса остальной части на другую строчку. Вместо этого прерывайте строку в «естественном» месте. Делайте перенос строки после любого из следующих символов *( {,; |*эти символы разрешают продолжить текущую строчку с новой строки.
* Если вы используете прокси функции, убедитесь что вы опубликовали ее в вашем модуле
* Используйте *Test-Path* и проверяйте им существование папки и файлов
* *Try..Catch...Finally*должны быть использованы везде где исключение может вызвать проблемы исполнения
* **Храните код логически простым**, например избегайте двойного отрицания в *if* выражении
* *Не используйте блокнот как редактор скриптов*, за исключением совсем простых действий. Используйте Powershell ISE. Блокнот подходит для быстрого просмотра кода небольших скриптов, т.к. блокнот является редактором по умолчанию
* *Избегайте использовать слово Return*. **Вместо этого думайте как передаются объекты по конвееру**. (есть специальная глава описывающая как правильно нужно передавать параметры. Настолько специфичная вещь для пошика что заслуживает отдельной статьи)

**40.3 лучшее из практики для промышленного использования**  
Эти рекомендации составлены для лучшего использования в промышленной среде.

* Используйте груповые политики для распространения настроек *PowerShell remoting*и политики исполнения скриптов
* Используйте удаленное подключение и*CIM сессии*для доступа если удаленных машин больше чем одна.
* Используйте задачи *(PowerShell jobs)* если вам нужно исполнить долгие операции
* Используйте workflows когда вам нужны скрипты умеющие пережить перезагрузку, прерывание или параллельные вычисления
* Храните скрипты в системе контроля версий, у вас будет резервная копия и вы сможете вернуться к старой версии при необходимости.
* Ограничивайте доступ к вашим скриптам, только для тех кому они нужны.
* Используйте *Test-Connection* для проверки доступности удаленной машины до того как начнете работать с ней. (проще — проверяй доступность хоста всегда)
* Убедитесь в том что PowerShell remoting включен на ваших серверах.
* Кредиталы должны быть заданы до того как будут использованы, не создавайте их в коде, особенно если вам нужно больше чем один. (проще — не генерируй и не храни пароли в скриптах, запрашивай их перед использованием)
* Используйте *DCOM WS-MAN* для *CIM сессий*если это возможно.
* Разработайте стандартизированные шаблоны скриптов, особенно если скрипты пишет команда администраторов.
* Установите политику исполнения *AllSigned*, используйте подпись кода сертификатом из вашего PKI (попозже сделаю отдельный перевод главы Security debate)
* Создайте общедоступный скрипт и определите в нем общие функции, алиасы, переменные которые могут понадобится вашей команде. Храните этот скрипт в общедоступном месте, *доступном по UNC* и затем добавьте ее в ваш профиль.
* Мы надеемся что это понятно без слов.*Тестируйте все в тестовой среде*. С современными возможностями виртуализации стало просто делать проверки. Вам не понадобится какоето особенно дорогое оборудование, вы можете начать с бесплатного VirtualBox или установить триальную версию продукта от Microsoft