

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

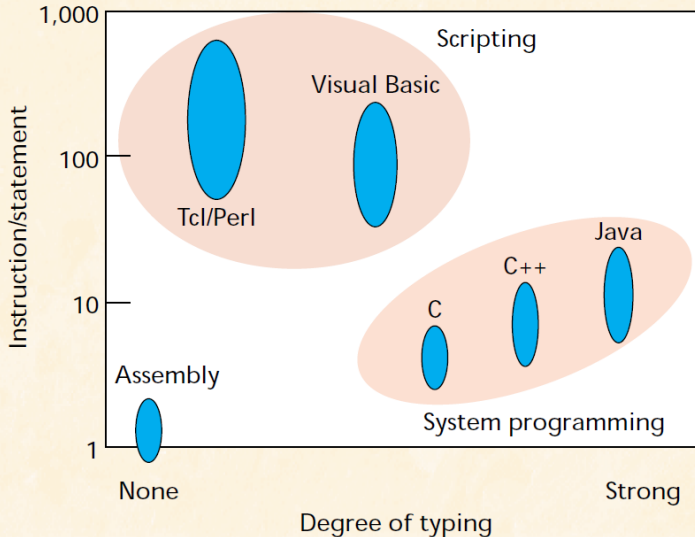
PowerShell
only

Credits

PowerShell Tutorial

Erik Hjelmås

March 21, 2012



(OUSTERHOUT, J., "Scripting: Higher-Level Programming for the 21st Century",
IEEE Computer, Vol. 31, No. 3, March 1998, pp. 23-30.)

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

If/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

WARNING!

The following presentation is NOT meant to be a comprehensive/complete tour of the PowerShell language.

The purpose is to get you started with some basic program constructions which you will recognize based on some-sort-of-programming-background.

At the end of the presentation (Credits section) you will find pointers to more comprehensive material (reference material).

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

If/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

Practice

You need a Windows host running on a physical or virtual machine with working access to the internet, and with PowerShell v2.0 installed.

Log in and open a terminal window, download the examples as we go along from

```
http://www.hig.no/~erikh/tutorial-powershell/FILENAME
```

(or download all at once with filename powershell-examples.zip but remember to unblock before unzip)

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

Hello World

```
# hello.ps1
```

```
Write-Host "hello world!"
```

execute as long as filename ends with .ps1:

```
.\hello.ps1
```

or direct from command line cmd (DOSPROMPT)

```
powershell -command "Write-Host \"hello world!\""
```

or direct from command line powershell

```
Write-Host "hello world!"
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Single Variables

```
# single-var.ps1

$firstname="Mysil"
$lastname="Bergsprekken"
$fullname="$firstname $lastname"
Write-Host "Hello $fullname, may I call you `
          $firstname`?"
```

All variables are prefixed with \$

Single and Double Quotes

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

If/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

```
# quotes.ps1
```

```
$name="Mysil"
```

```
Write-Host Hello $name
```

```
Write-Host "Hello $name"
```

```
Write-Host 'Hello $name'
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

One-dimensional arrays:

```
# array.ps1

$os=@("linux", "windows")
$os+=@("mac")
Write-Host $os[1]      # print windows
Write-Host $os        # print array values
Write-Host $os.Count  # length of array
```

Arrays are created with @(...)

Associative Arrays

```
# assoc-array.ps1

$user=@{
    "frodeh" = "Frode Haug";
    "ivarm"  = "Ivar Moe"
}

$user+=@{"lailas"="Laila Skiaker"}

Write-Host $user["ivarm"]    # print Ivar Moe
Write-Host @user             # print array values
Write-Host @user.Keys        # print array keys
Write-Host $user.Count       # print length of array
```

Associative arrays are created with `@{...}`

A simple object used as a struct:

```
# struct.ps1

$myhost=New-Object PSObject -Property `
    @{os=" ";
      sw=@();
      user=@{}}

$myhost.os="linux"
$myhost.sw+=@("gcc","flex","vim")
$myhost.user+=@{
    "frodeh"="Frode Haug";
    "monicas"="Monica Strand"
}

Write-Host $myhost.os
Write-Host $myhost.sw[2]
Write-Host $myhost.user["monicas"]
```

Command-Line Arguments

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

All command-line arguments in the array `$args`

Scriptname retrieved from the object `$MyInvocation`

```
# cli-args.ps1
```

```
Write-Host "I am" $MyInvocation.InvocationName `
           "and have" $args.Count "arguments" `
           "first is" $args[0]
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Input From User

```
# input-user.ps1
```

```
$something=Read-Host "Say something here"  
Write-Host "you said" $something
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Input From the Pipeline

```
# input-pipe.ps1

$something="$input"
Write-Host "you said" $something
```

can be executed as

```
Write-Output "hey hey!" | .\input-pipe.ps1
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Input From Files

```
# input-file.ps1

$file=Get-Content hello.ps1
Write-Host @file -Separator "`n"
```

Input from System Commands

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

```
# input-commands.ps1

$name=(Get-WmiObject Win32_OperatingSystem).Name
$kernel=(Get-WmiObject `
    Win32_OperatingSystem).Version
Write-Host "I am running on $name, version" `
    "$kernel in $(Get-Location)"
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

```
# if.ps1

if ($args.Length -ne 1) {
    Write-Host "usage:" `
        $MyInvocation.InvocationName `
        "<argument>"
}
```


Comparison

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Operator	Meaning
-lt	Less than
-gt	Greater than
-le	Less than or equal to
-ge	Greater than or equal to
-eq	Equal to
-ne	Not equal to

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

Boolean

Operator	Meaning
-not	Not
!	Not
-and	And
-or	Or

```
# if-num-string.ps1

if ($args.Count -ne 2) {
    Write-Host "usage:" `
        $MyInvocation.InvocationName `
        "<argument> <argument>"

    exit 0
} elseif ($args[0] -gt $args[1]) {
    Write-Host $args[0] "larger than" $args[1]
} else {
    Write-Host $args[0] "smaller than or" `
        "equal to" $args[1]
}

if (Test-Path $args[0]) {
    if (!(Get-Item $args[0]).PSIsContainer) {
        Write-Host $args[0] "is a file"
    }
}
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

Boolean example

```
# if-bool.ps1

if ((1 -eq 2) -and (1 -eq 1) -or (1 -eq 1)) {
    Write-Host "And has precedence"
} else {
    Write-Host "Or has precedence"
}

# force OR precedence:

if ((1 -eq 2) -and ((1 -eq 1) -or (1 -eq 1))) {
    Write-Host "And has precedence"
} else {
    Write-Host "Or has precedence"
}
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

Switch/Case

```
# switch.ps1

$short = @{ yes="y"; nope="n" }
$ans = Read-Host
switch ($ans) {
    yes { Write-Host "yes" }
    nope { Write-Host "nope"; break }
    { $short.ContainsKey("$ans") } `
        { Write-Host $short[$ans] }
    default { Write-Host "$ans `???" }
}
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Where/Where-Object

```
# where.ps1
```

```
Get-ChildItem | Where-Object {$_ .Length -gt 1KB}
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

ForEach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

```
# for.ps1

for ($i=1;$i-le3;$i++) {
    Write-Host "$i"
}

# something more useful:

$file=Get-ChildItem
for ($i=0;$i-lt$file.Count;$i++) {
    if (!(Get-Item $file[$i]).PSIsContainer) {
        Write-Host $file[$i].Name "is a file"
    } else {
        Write-Host $file[$i].Name "is a directory"
    }
}
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

```
# while.ps1

while ($i -le 3) {
    Write-Host $i
    $i++
}

# something more useful:

$file=Get-ChildItem
$i=0
while ($i -lt $file.Count) {
    if (!(Get-Item $file[$i]).PSIsContainer) {
        Write-Host $file[$i].Name "is a file"
    } else {
        Write-Host $file[$i].Name "is a directory"
    }
    $i++
}
```


Foreach loop

Variables

Arrays
Structures/Classes
Command-line args

Input

Input
System commands

Conditions

if/else
Operators
Switch/case
Where

Iteration

For
While
Foreach

Math

Functions

RegExp
PowerShell example

PowerShell
only

Credits

```
# foreach.ps1

foreach ($i in Get-ChildItem) {
    Write-Host $i.Name
}

# with associative arrays

$user=@{
    "frodeh" = "Frode Haug";
    "monicas" = "Monica Strand";
    "ivarm" = "Ivar Moe"
}

foreach ($key in $user.Keys) {
    Write-Host $user[$key]
}
```

Foreach

If we want to read from the pipeline and do stuff object by object:

```
# foreach-pipe.ps1

foreach ($i in $input) {
    $foo += @($i)
}

Write-Host "size of foo is" $foo.Count
```

or

```
# foreach-object-pipe.ps1

$input | ForEach-Object {
    $foo += @($_)
}

Write-Host "size of foo is" $foo.Count
```

```
$ Get-ChildItem | ./foreach-object-pipe.ps1
size of foo is 20
```

Operators

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

If/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Operator	Meaning
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulus

```
# math.ps1
```

```
Write-Host "3+5 is" (3+5)
```

Functions

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

If/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

```
# func.ps1

# declare:
function add($a, $b) {
    Write-Host "$a+$b is" ($a+$b)
}

# use:
add 5.12 2.56
```

Regular expressions intro 1/5

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

Special/Meta-characters:

`\ | () [] { } ^ $ * + ? .`

These have to be protected with `\`, e.g.

`http://www\.hig\.no`

To match `c:\temp`, you need to use the regex `c:\\temp`. As a string in C++ source code, this regex becomes "`c:\\\\temp`". Four backslashes to match a single one indeed.

(from `http:`

`//www.regular-expressions.info/characters.html`):

Regular expressions intro 2/5

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Describing characters:

Operator	Meaning
.	Any single character
[abcd]	One of these characters
[^abcd]	Any one but these characters
[a-zA-Z0-9]	A character in these ranges

Regular expressions intro 3/5

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

Grouping:

Operator	Meaning
()	Group
	OR

Anchoring:

Operator	Meaning
^	Beginning of line
\$	End of line

Regular expressions intro 4/5

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell

only

Credits

Repetition operators/Modifiers/Quantifiers:

Operator	Meaning
?	0 or 1 time
*	0 or more times
+	1 or more times
{N}	N times
{N, }	At least N times
{N, M}	At least N but not more than M

Regular expressions intro 5/5

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell

only

Credits

Finding URLs in HTML:

```
(mailto|http)://[~"]*
```

Each line should be an email address:

```
^[A-Za-z0-9._-]+@[A-Za-z0-9.-]+$
```

PowerShell example

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

```
# regex.ps1

$input | ForEach-Object {
    if ($_ -match
        "^[A-Za-z0-9._-]+@([A-Za-z0-9.-]+)$") {
        Write-Host "Valid email", $matches[0]
        Write-Host "Domain is", $matches[1]
    } else {
        Write-Host "Invalid email address!"
    }
}
```

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

Foreach

Math

Functions

RegExp

PowerShell example

PowerShell
only

Credits

Advanced stuff

See the complete Mastering PowerShell book at

<http://powershell.com/cs/blogs/ebook/>

for much more of what you can do with PowerShell

Variables

Arrays

Structures/Classes

Command-line args

Input

Input

System commands

Conditions

if/else

Operators

Switch/case

Where

Iteration

For

While

ForEach

Math

Functions

RegExp

PowerShell example

PowerShell only

Credits

<http://refcardz.dzone.com/refcardz/windows-powershell>

<http://powershell.com/cs/blogs/ebook/>

<http://technet.microsoft.com/en-us/library/ee692948.aspx>

http://www.techotopia.com/index.php/Windows_PowerShell_1.0_String_Quoting_and_Escape_Sequences

<http://dmitrysotnikov.wordpress.com/2008/11/26/input-gotchas/>

<http://stackoverflow.com/questions/59819/>

[how-do-i-create-a-custom-type-in-powershell-for-my-scripts-to-use](http://www.powershellpro.com/powershell-tutorial-introduction/)

<http://www.powershellpro.com/powershell-tutorial-introduction/>

http://en.wikipedia.org/wiki/Windows_PowerShell

<http://www.johndcook.com/powershell.html>

<http://www.regular-expressions.info/>

OUSTERHOUT, J., "Scripting: Higher-Level Programming for the 21st Century", IEEE Computer, Vol. 31, No. 3, March 1998, pp. 23-30.)