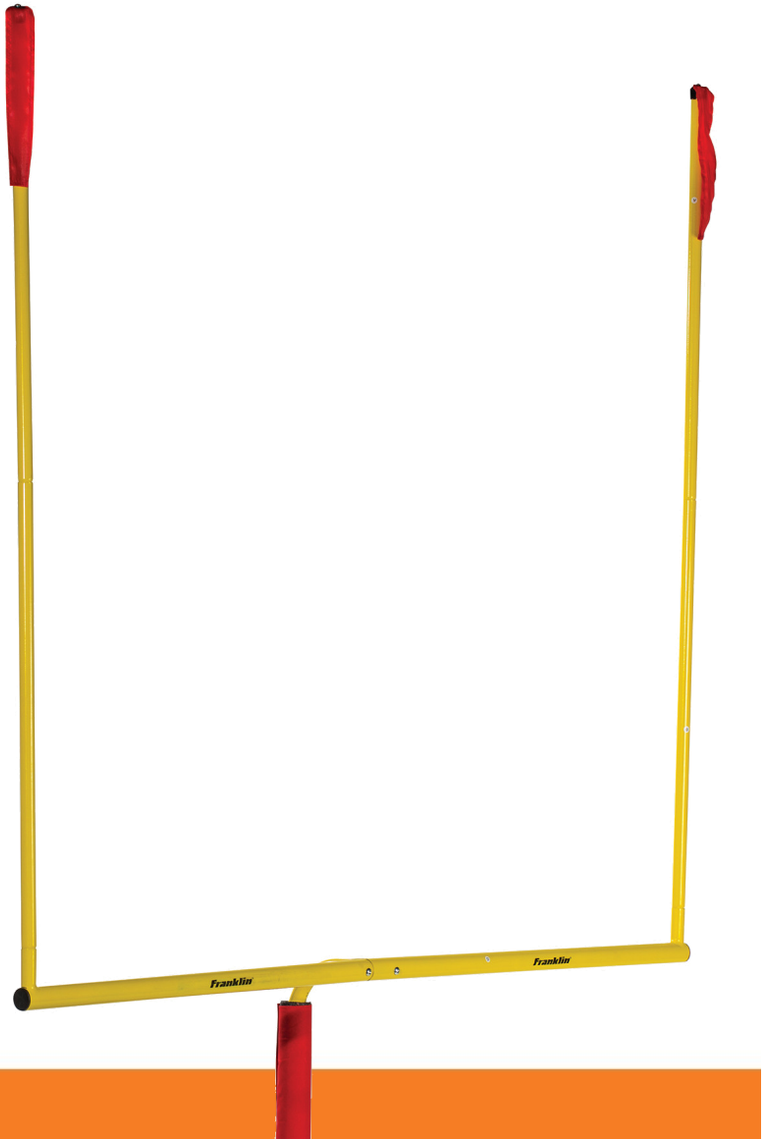
Modules

Develop Intelligence

Goals

1. Explain the benefits of PowerShell modules
2. Explain the module path
3. List the 2 uses of keyword `using`



Roadmap

1. Scripts
2. Simple Modules
3. Fancy Modules

Develop
Intelligence

Scripts

Develop Intelligence

Overview

- Rules
  - Plain text
  - Extension '.ps1'
  - 1 or more PowerShell commands
- Run via
  - Terminal
  - Some other script

Example

Here's MyScript.ps1

```
Write-Output "I do nothing"

function Get-Self($Self){
  return $Self
}
```

Invoke via Terminal

```
$ ./MyScript.ps1
```

Invoke via script

```
# Run directly
./MyScript.ps1
```

Script Parameters

Contents of `Get-Self.ps1`

```
1 Param($Self)
2 return $Self
```

Invoke via Terminal

```
1 $ ./Get-Self.ps1 -Self 'There is no spoon.'
```

Dot-Sourcing

- Imports *everything* into the current session
- Simple, old-school
- Only works for '.ps1' extension
- Might clobber
  - Variables
  - Functions
  - Aliases

Ignoring Output

- Avoid pipeline pollution with `Out-Null`

```
function Start-Enfarculation{
    Write-Host 'Commencing Enfarculation Procedure'
    # Dot-source and ignore output
    . ./Get-Greeting.ps1 | Out-Null
    Get-Greeting
    Write-Host 'Initialization Complete...'
}
```

Benefits (often) Outweighed

- Advantages
  - Do it anywhere
  - Works with variables
- Costs
  - No way to select what's being imported
  - Side effects

Develop
Intelligence

Simple Modules

Develop Intelligence

Overview

- A module is a package of:
  - Cmdlets
  - Functions
  - Variables
  - Etc
- Turn a script into a module: Change the extension to **psm1**
- Modules can't be dot-sourced

Auto-Loading

PowerShell imports modules automatically the first time that you run any command in an installed module. You can now use the commands in a module without any set-up or profile configuration, so there's no need to manage modules after you install them on your computer.

- Installation path `$env:PSModulePath`
- Disable with `$PSModuleAutoloadingPreference = 'None'`

Export-ModuleMember

- By default, everything in your module is available to consumers
- **Best Practice**: Explicitly export the public interface

```powershell
# Get-Self.psm1

$secret = 'i like bunnies'

function Get-Self{
  #...
}

Export-ModuleMember -Function Get-Self
```

Using `Import-Module`

- Adds a module to the current session
- By default imports all public members
  - Aliases
  - Functions
  - Variables
- Lets you import only the things you need

Examples

Develop Intelligence

Import Everything

```
Import-Module $PSScriptRoot/Admin.psm1
```

Import `Get-Users` Only

```
Import-Module $PSScriptRoot/Admin.psm1 -Function Get-Users
$users = Get-Users
```

Prefix to Avoid Collisions

```
Import-Module $PSScriptRoot/Admin.psm1 `
  -Function Get-Users `
  -Prefix Windows
$users = Get-WindowsUsers
```

Using `using`

- Lets you to specify which namespaces are used in the session
- Good for 'importing'
  - .NET namespaces
  - Powershell modules
- Rules:
  - Must be at the top of the file
  - No variables allowed

Motivation

```powershell
$changeTypes = [System.IO.WatcherChangeTypes]::Created, [System.IO.Watc
$watcher = [System.IO.FileSystemWatcher]::new('c:/temp/');

while($true){
  $result = $watcher.WaitForChanged($changeTypes, 5000)
  if(-not $result.TimedOut){
    "Got a change: $($result.ChangeType), $($result.Name)"
  }
}
```

Refactored

Develop Intelligence

```
using namespace System.IO

[WatcherChangeTypes]$changeTypes = 'Created', 'Deleted'
[FileSystemWatcher]$watcher = [FileSystemWatcher]::new('c:/temp/');

while($true){
  $result = $watcher.WaitForChanged($changeTypes, 5000)
  if(-not $result.TimedOut){
    "Got a change: $($result.ChangeType), $($result.Name)"
  }
}
```

Develop
Intelligence

Fancy Modules

Develop Intelligence

The Manifest

- Manifests aren't required
- BUT they're good for
  - Metadata
  - Packaging
  - Sharing - e.g. Powershell Gallery
- Generate with New-ModuleManifest

Example

```
@{
  RootModule = 'SampleModule.psm1'
  ModuleVersion = '1.0.0'
  GUID = '3ff4d4b6-ade9-46f3-a4f2-2ad6f5508388'
  Author = '<name>'
  CompanyName = 'Unknown'
  Copyright = '(c) 2021 <name>.'
  Description = 'Some description.  This is required by the PowerShell
  FunctionsToExport = @('New-File','Import-FileNoWildcard','Import-Fil
}
```

Develop
Intelligence

Review

1. Explain the benefits of PowerShell modules
2. Explain the module path
3. List the 2 uses of keyword `using`