PowerShell Conference Europe 2019

Hannover, Germany

June 4-7, 2019

A better way to do WPF in PowerShell 5+

JAKUB JARES

This Session

- Learn how to write a reactive application in WPF without naming all your components and explicitly updating all of them.

nohwnd

The agenda

- Traditional approach to WPF
- MVVM and WPF primer
- Example in C#
- Example in PowerShell
- How it all works

nohwnd

Traditional approach to WPF in PowerShell

nohwnd

Traditional approach to WPF in PowerShell

- Name all components in the View
- Lookup all components by name in code-behind
- Add Click handlers to buttons
- Set and update data explicitly

DEMO

Traditional approach

nohwnd

Traditional approach

Upsides

- Very explicit
- Easy to grasp
- Easy to do in PowerShell
- Similar to WinForms

Downsides

- Naming everything is annoying
- Updating data has to be done manually
- Disabling actions based on validation is complicated
- Not how WPF is meant to use

New approach

nohwnd

DEMO

MVVM

nohwnd

MVVM

Upsides

- The way WPF is meant to use
- Data update automatically
- Disabling actions based on validation is easy
- Testable

Downsides

- Impossible to do properly in PowerShell.

Or, not? 😀

DEMO

PokeBrowser

Piece of Pokemon trivia

nohwnd

Pichu can evolve to Pikachu when it has a good friend

Pichu
Electric

→

(high Friendship)

Pikachu
Electric

#awesomeJaap

Pikachu

Electric

→

(awesomeness + PowerShell
+ whiskey / Panzerwagons)

Jaapchu

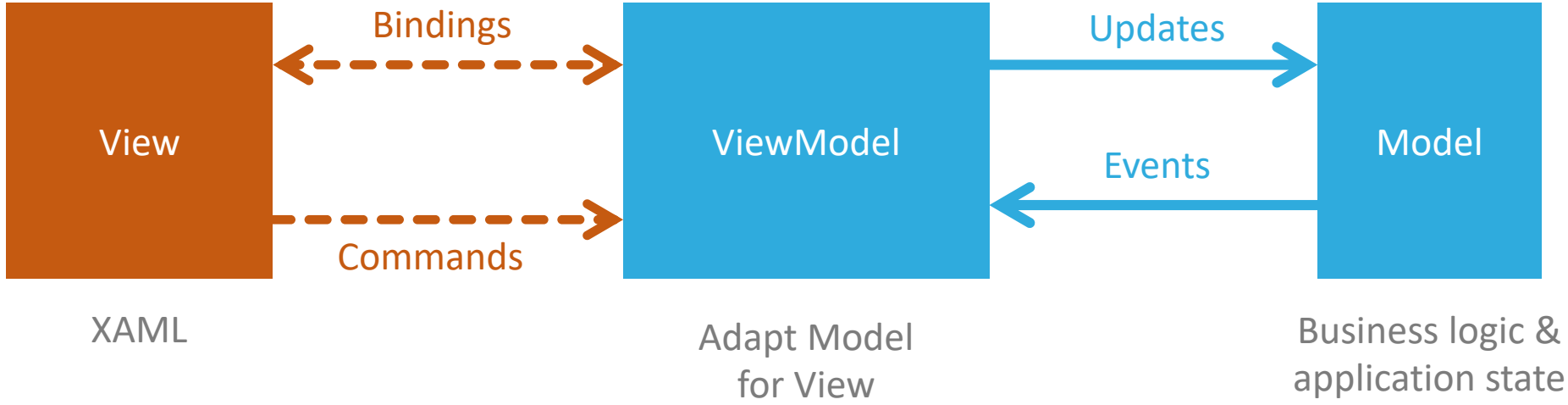
Electric

#awesomeJaap

Introduction to MVVM

nohwnd

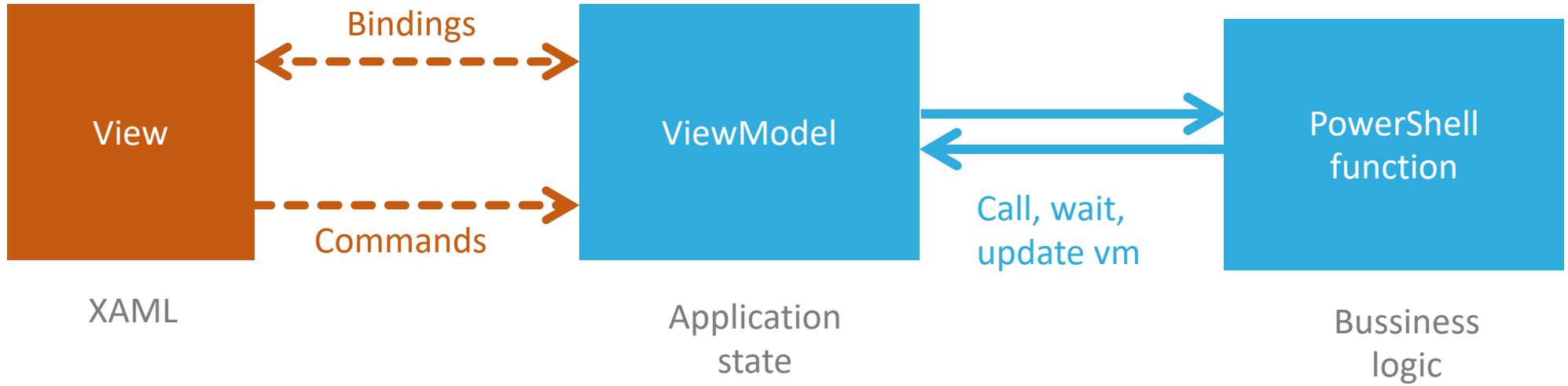
MVVM


View
Bindings
Commands
ViewModel
Updates
Events
Model
XAML
Adapt Model
for View
Business logic &
application state


PSCONF.EU

nohwnd

MVVM in PowerShell



nohwnd

Binding

nohwnd

Binding

- Connects View to a property on the ViewModel
- One way or two way

- Works for top-level properties but also for child properties

```
<Label Content="{Binding Name}" />
<Label Content="{Binding SelectedUser.Role.Name}" />
<TextBox Text="{Binding Path=Name, Mode=TwoWay}" />
```

Updates

- Update from View is written directly to the bound property
- Update from ViewModel is written into the property + PropertyChanged notification

```
PropertyChanged(new PropertyChangedEventArgs("Name"));
```

```
INotifyPropertyChanged
```

Commands

nohwnd

Command

- Connects View to an action on the ViewModel
- Easily invoked from multiple places, like menu, button, context menu and keyboard shortcut
- Can be protected by a guard, that disables the component when condition is not met

```
<Button Command="{Binding Refresh}">Refresh</Button>
```

Command

- Implemented as ICommand **property** on the type
- Typically implemented as RelayCommand class that takes the action to do, and filter that returns true when the action is enabled
- Filter is evaluated repeatedly so it should be cheap to run

Async

nohwnd

Async

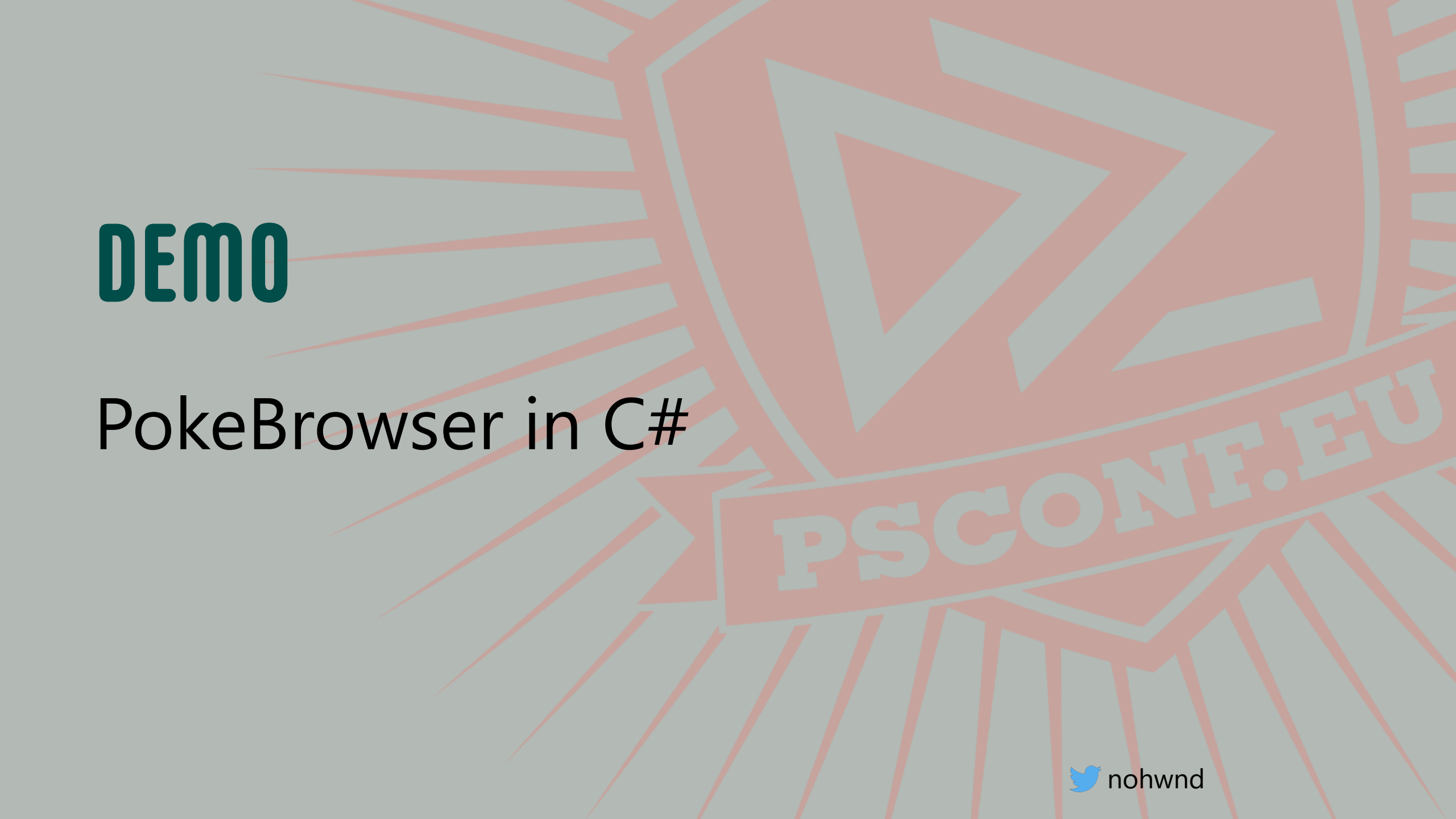
- IO work is easy to do asynchronously using Task
- Heavy work is easily offloaded to background using Task

- The goal is to not block the UI thread

- Callbacks to UI can be done using Dispatcher

DEMO

PokeBrowser in C#



nohwnd

Summary

- ViewModel is a class implementing **`INotifyPropertyChanged`**
- Properties raise **`PropertyChanged`** event when they are **`set`** to a new value
- Commands use **`RelayCommand`** that refers to **`this`**
- **`Task`** and **`async`** is used to do async and heavy work

PowerShell

nohwnd

Problems

There are a few

nohwnd

ViewModel

❌ Problem:

- PowerShell classes can't easily implement events

✔ Solution:

- Derive from a C# class **`ViewModelBase`** that implements INotifyPropertyChanged

Binding

❌ Problem:

- PowerShell classes can't define custom setters

✓ Solution:

- Add **`Init`** method that will add a **`ScriptMethod`** that sets the value and notifies

```
$this.Init('Selected')
$this.SetSelected(($pokemon | Select -First 1))
```

Commands

❌ Problem:

- Commands are easy to implement as scriptblocks but they miss the **`this`** reference

✓ Solution:

- Write a **`RelayCommand`** that takes **`this`** reference and pass that by param

Background work

❌ Problem:

- We don't have async or any other easy way to run background work

✓ Solution:

- Use runspaces and a synced hash table to pass state to it (Thanks BoeProx @ProxB! https://learn-powershell.net/2012/10/14/powershell-and-wpf-writing-data-to-a-ui-from-a-different-runspace/)
- Define helper Dispatcher function to make callbacks easier

Other

❌ The code is complex -> ✓ hide the complexity from the user by providing a handy method on **`this`**

❌ Methods on ViewModelBase are in C# -> ✓ expose methods on the object, but internally call back to powershell scriptblocks where possible

❌Classes need to be defined before use -> ✓ define them in a separate file and dot-source it

nohwnd

DEMO

PokeBrowser in PowerShell

nohwnd

DEMO

Command guards

nohwnd

DEMO

Converters

nohwnd

Approaches that did not work

- writing all the code in powershell including the events - it gets stuck

- binding psobject - cannot add events to it

- binding empty Notifiable object decorated with noteproperties taken from psobject - it cannot bind anything (because IDynamicMetaObjectProvider is missing, thanks Steve Lee! @Steve_MSFT! https://twitter.com/nohwnd/status/1126241707698327552) <- this is probably the best way forward

- marshalling the scriptblock via the sync hash and invoking it directly, it needs to be re-created - it gets stuck

Possible improvements

- automatic init for all properties, unless they are excluded

- Init with dependent properties (so one property can notify about change in two or more)

- make turn on binding debugging by function

- research converters (would be useful for the visibility)

- make it into a full framework using wpf extras, with stuff like built in progress etc

- invoke action with progress instead of try finally

Wanna adopt this?

Get in touch :)

nohwnd

Summary

- MVVM is possible in PowerShell and makes making reactive UI reasonably simple
- You can choose between traditional and new style based on the app you are creating
- I've shown the absolute minimum to get the points across, WPF can do way more than this

I have Pester stickers come say hi and grab some.

nohwnd

Slides and demo code

```
Start-Process -FilePath https://github.com/psconfeu/2019
```

https://github.com/nohwnd/WpfToolkit

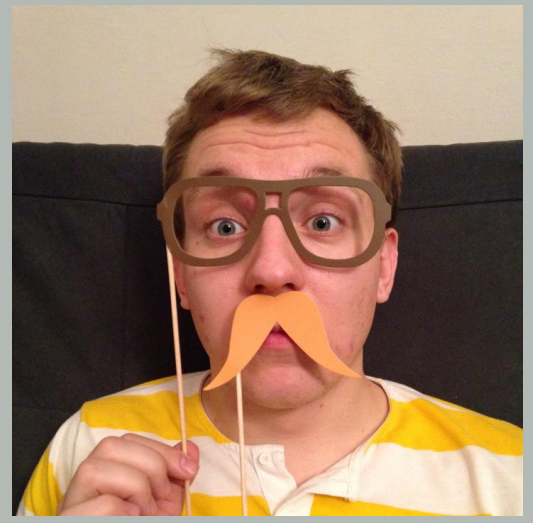
nohwnd

Questions?

Use the conference app to vote for this session:
https://my.eventraft.com/psconfeu

nohwnd

about_Speaker



Jakub Jares

@nohwnd

me@jakubjares.com

I am Jakub Jares and enjoy coding and architecting solutions. For the past few years I work as a .NET Developer for cngroup.dk in Prague, Czech Republic. I mainly work with C# and F#, during the day. But in the evenings I often go back to PowerShell and improve and maintain Pester, or try to find out if something crazy, like doing MVVM in PowerShell, is possible.