

DWX

DEVELOPER WEEK '21

Fabulous Functional Frontends

A dark, atmospheric photograph of a snow-covered mountain range with a prominent peak in the center. A body of water is visible in the foreground.

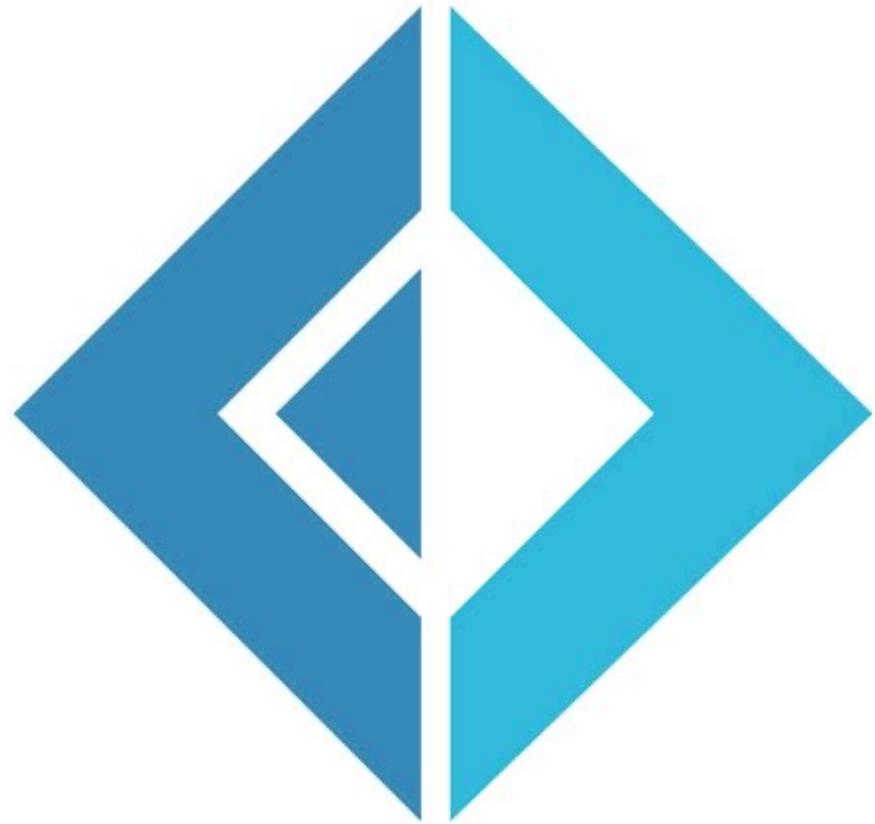
Mark Allibone

Mobile Lead

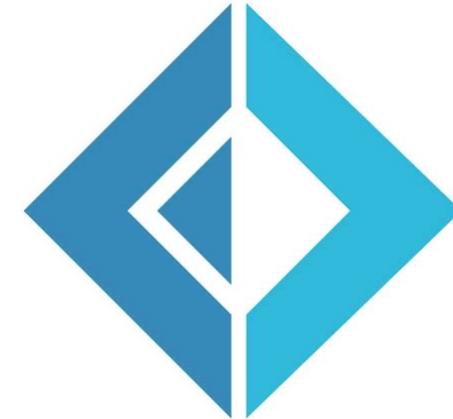
Rey Technology

#dwx21

@mallibone



- No Nulls
- Immutability
- Declarative Programming
- Based on .NET





@mallibone



@mallibone

A screenshot of the Visual Studio 2019 IDE interface, showing a Xamarin.Forms project named "HelloFSharpConf".

The top navigation bar includes File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar for "Search Visual Studio (Ctrl+Q)".

The main editor window displays the file `AppPage.xaml.fs` with the following F# code:

```
1  namespace HelloFSharpConf
2
3  open Xamarin.Forms
4  open Xamarin.Forms.Xaml
5
6  type HelloFSharpConfPage() =
7      inherit ContentPage()
8      let _ = base.LoadFromXaml(typeof<HelloFSharpConfPage>)
9
10 do
11     base.BindingContext <- new MyViewModel()
12 ()
```

The Solution Explorer on the right shows the solution structure:

- Solution 'HelloFSharpConf' (3 of 2 projects)
 - HelloFSharpConf
 - Dependencies
 - AppViewModel.fs
 - AppPage.xaml
 - App.xaml
 - AssemblyInfo.fs
 - HelloFSharpConf.Droid
 - HelloFSharpConf.iOS

The Properties window for the `HelloFSharpConf.Droid` project is open, showing the `HelloFSharpConf.Droid.csproj` file.

The Output window at the bottom shows the following log output:

```
06-14 06:39:05.177 D/EGL_emulation( 4416): eglGetCurrent: 0xa0aa785120: ver 3 0 (tinfo 0xaa783250)
06-14 06:39:05.288 D/Mono   ( 4416): DllImport searching in: '__Internal' ('(null)').
06-14 06:39:05.288 D/Mono   ( 4416): Searching for 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono   ( 4416): Probing 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono   ( 4416): Found as 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.297 D/EGL_emulation( 4416): eglGetCurrent: 0xa0aa785120: ver 3 0 (tinfo 0xaa783250)
```

The status bar at the bottom indicates: "This item does not support previewing".

The footer URL is <https://github.com/jimbobbennett/HelloFSharpConf>.



Live Share



Solution Explorer



Search Solution Explorer (Ctrl+ü)

✓ Solution 'HelloFSharpConf' (3 of 2 projects)

▲ ✓ HelloFSharpConf

▷ Dependencies

+ AppViewModel.fs

▷ AppPage.xaml

▷ App.xaml

+ AssemblyInfo.fs

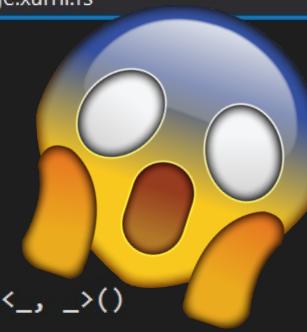
▷ ✓ HelloFSharpConf.Droid

▷ ✓ HelloFSharpConf.iOS

Window Snip

A screenshot of the Visual Studio IDE interface. The top menu bar includes File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar for "Search Visual Studio (Ctrl+Q)". A tab bar at the top shows "HelloFSharpConf" as the active project. Below the menu is a toolbar with various icons for file operations like Open, Save, and Build. The main workspace displays an F# code editor for "AppViewModel.fs". The code defines a type `MyViewModel` with properties `text` and `TapCommand`, and implements the `INotifyPropertyChanged` interface. The code uses F# syntax such as `Event<_, _>` and `System.Windows.Input.ICommand`. The code editor has line numbers on the left and syntax highlighting for F# keywords and types. The status bar at the bottom shows the current file path "HelloFSharpConf.Droid" and the build configuration "my_device (Android 8.1 - API 27)".

```
21  namespace HelloFSharpConf
20
19  type MyViewModel () =
18      let ev = new Event<_,_>()
17      let mutable text = ""
16
15      let createCommand action =
14          let event1 = Event<_, _>()
13          {
12              new System.Windows.Input.ICommand with
11                  member this.CanExecute(obj) = true
10                  member this.Execute(obj) = action(obj)
9                      member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
8                      member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
7              }
6
5      interface System.ComponentModel.INotifyPropertyChanged with
4          []
3              member this.PropertyChanged = ev.Publish
2
1              member this.Text
22                  with get() = text
1                  and set(value) = text <- value
2                      ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
3              member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")
4
5
```



```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search Visual Studio (Ctrl+Q) HelloFSharpConf  
Debug Any CPU HelloFSharpConf.Droid my_device (Android 8.1 - API 27)  
AndroidManifest.xml AppViewModel.fs AppPage.xaml.fs  
21  namespace HelloFSharpConf  
20  
19  type MyViewModel () =  
18      let ev = new Event<_, _>()  
17      let mutable text = ""  
16  
15      let createCommand action =  
14          let event1 = Event<_, _>()  
13          {  
12              new System.Windows.Input.ICommand with  
11                  member this.CanExecute(obj) = true  
10                  member this.Execute(obj) = action(obj)  
9                      member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)  
8                      member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)  
7          }  
6  
5      interface System.ComponentModel.INotifyPropertyChanged with  
4          [<>CLIEvent>]  
3              member this.PropertyChanged = ev.Publish  
2  
1          member this.Text  
2              with get() = tex  
1                  and set(value) = text <- value  
2                      ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))  
3              member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")  
4  
5
```

The screenshot shows the Visual Studio IDE interface with the following details:

- Menu Bar:** File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search Visual Studio (Ctrl+Q) with a magnifying glass icon.
- Toolbox:** On the left side.
- Test Explorer:** On the far left.
- Toolbar:** Includes icons for back, forward, search, and file operations.
- Solution Explorer:** Shows files like AndroidManifest.xml, AppViewModel.fs, AppPage.xaml.fs, and HelloFSharpConf.Droid.
- Properties Explorer:** Shows build configurations (Debug, Any CPU), platform (my_device), and other settings.
- Code Editor:** Displays F# code for `AppViewModel.fs`. A red rectangle highlights the implementation of the `ICommand` interface.

```
21  namespace HelloFSharpConf
20
19  type MyViewModel () =
18      let ev = new Event<_,_>()
17      let mutable text = ""
16
15  let createCommand action =
14      let event1 = Event<_, _>()
13      {
12          new System.Windows.Input.ICommand with
11              member this.CanExecute(obj) = true
10              member this.Execute(obj) = action(obj)
9                  member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
8                  member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
7          }
6
5  interface System.ComponentModel.INotifyPropertyChanged with
4      [<<CLIEvent>]
3      member this.PropertyChanged = ev.Publish
2
1  member this.Text
22      with get() = text
1      and set(value) = text <- value
2          ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
3      member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")
4
5
```

The screenshot shows the Visual Studio IDE interface with the following details:

- Menu Bar:** File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search Visual Studio (Ctrl+Q) with a magnifying glass icon.
- Toolbox:** On the left side.
- Test Explorer:** On the left side.
- Toolbar:** Includes icons for back, forward, search, and file operations.
- Solution Explorer:** Shows the project structure with files like AndroidManifest.xml, AppViewModel.fs, AppPage.xaml.fs, and HelloFSharpConf.Droid.
- Properties Explorer:** Shows the build configuration as "Debug" and platform as "Any CPU".
- Task List:** Shows the current task as "my_device (Android 8.1 - API 27)".
- Code Editor:** Displays F# code for a view model. The code defines a type `MyViewModel` with properties `text` and `ev`, and methods `createCommand` and `PropertyChanged`. A red box highlights the implementation of the `Text` property, which uses a mutable state pattern and triggers a `PropertyChangedEventArgs` event.

```
21  namespace HelloFSharpConf
20
19  type MyViewModel () =
18      let ev = new Event<_,_>()
17      let mutable text = ""
16
15      let createCommand action =
14          let event1 = Event<_, _>()
13          {
12              new System.Windows.Input.ICommand with
11                  member this.CanExecute(obj) = true
10                  member this.Execute(obj) = action(obj)
9                      member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
8                      member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
7
6
5      interface System.ComponentModel.INotifyPropertyChanged with
4          []
3              member this.PropertyChanged = ev.Publish
2
1
1      member this.Text
2          with get() = tex
1          and set(value) = text <- value
2              ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
3      member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")
4
5
```

A screenshot of the Visual Studio IDE interface, showing the development of a Xamarin application named "HelloSharpConf".

The main window displays the code for `AppViewModel.fs` (line numbers 1-27). The code defines a `MyViewModel` type with properties `text` and `TapCommand`, and implements `ICommand` and `INotifyPropertyChanged`.

```
1  namespace HelloSharpConf
2
3  type MyViewModel () =
4      let ev = new Event<_, _>()
5      let mutable text = ""
6
7      let createCommand action =
8          let event1 = Event<_, _>()
9          {
10              new System.Windows.Input.ICommand with
11                  member this.CanExecute(obj) = true
12                  member this.Execute(obj) = action(obj)
13                  member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
14                  member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
15          }
16
17      interface System.ComponentModel.INotifyPropertyChanged with
18          [<<<CLIEvent>>]
19          member this.PropertyChanged = ev.Publish
20
21      member this.Text
22          with get() = text
23          and set(value) = text <- value
24          ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
25      member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")
26
27
```

The Solution Explorer on the right shows the project structure with two projects: `HelloSharpConf` and `HelloSharpConf.Droid`. The `HelloSharpConf` project contains files like `Dependencies`, `AssemblyInfo.fs`, and `App.xaml`.

The Output window at the bottom shows log messages related to EGL emulation and Mono DLL imports.

```
06-14 06:39:05.177 D/EGL_emulation( 4416): eglGetCurrent: 0xa0a785120: ver 3 0 (tinfo 0xa0a783250)
06-14 06:39:05.288 D/Mono   ( 4416): DllImport searching in: '_Internal' ('(null)'). 
06-14 06:39:05.288 D/Mono   ( 4416): Searching for 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono   ( 4416): Probing 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono   ( 4416): Found as 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.297 D/EGL_emulation( 4416): eglGetCurrent: 0xa0a785120: ver 3 0 (tinfo 0xa0a783250)
```



@mallibone



Fabulous – a MVU framework for F# working with Xamarin.Forms

@mallibone

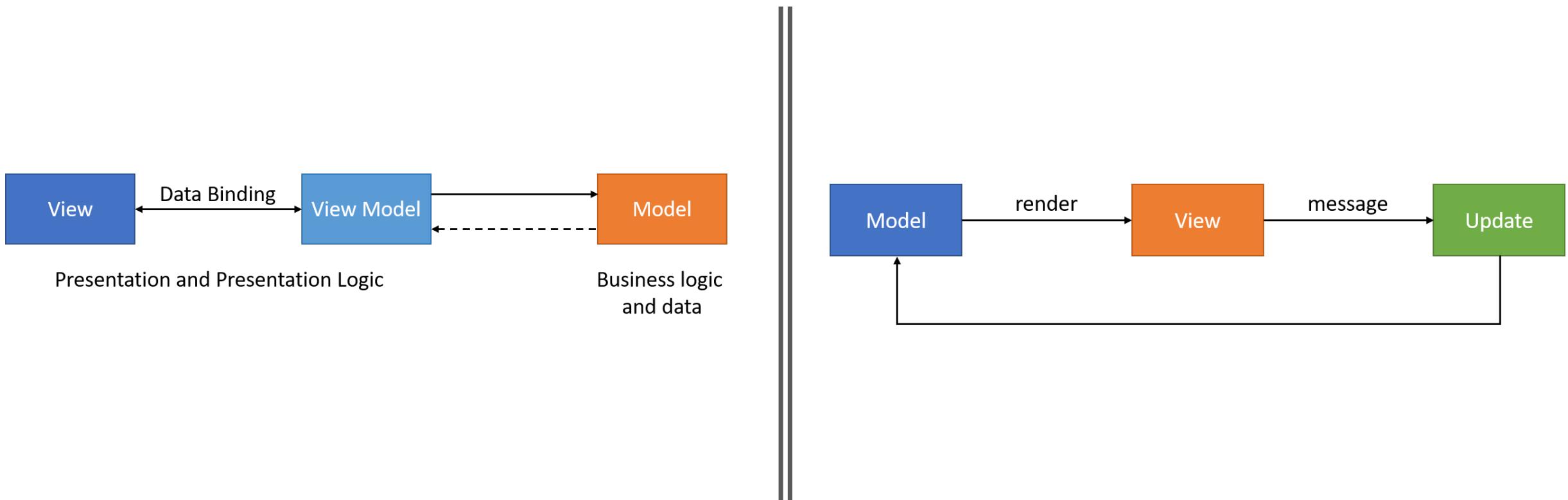


MVU the functional MV* Pattern

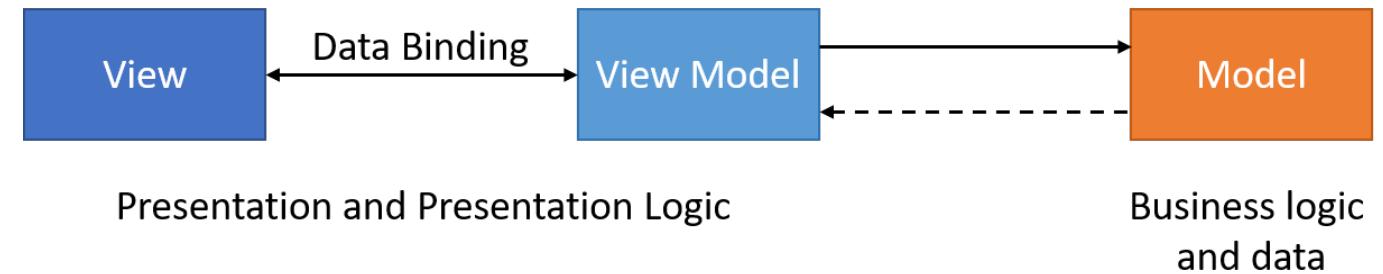


@mallibone

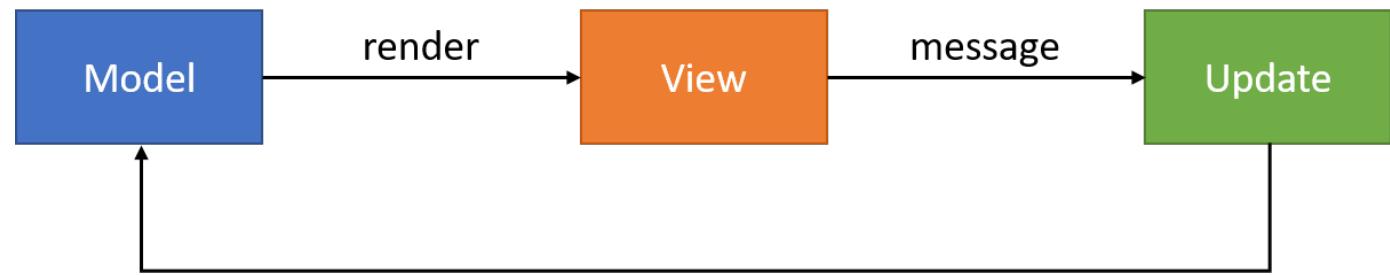
MVVM vs MVU



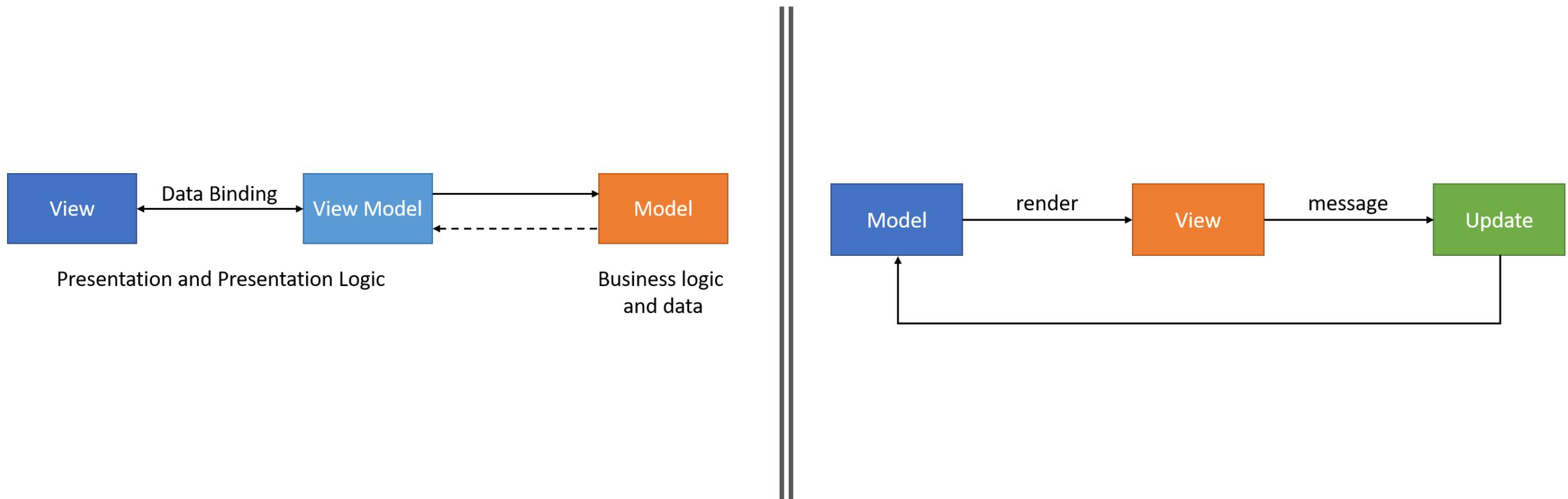
Model View View Model



Model View Update



MVVM vs MVU



A close-up photograph of a pink flamingo's head and neck, facing right. The flamingo has a long, thin, pink neck and a large, curved, black beak. Its head is pink with a small black eye. The background is a soft-focus, warm-toned landscape of water and sky.

NICOTINEBATCH | TUMBLR

FABULOUS

KBS 11
Seoul

Functional Frontends

Getting started

1. Install Visual Studio or Visual Studio for Mac and enable both Xamarin and .NET Core support, these are listed as 'Mobile development with .NET' and '.NET Core Cross-platform development' respectively.
2. Open a command prompt window and install the template pack by entering:

```
dotnet new -i Fabulous.XamarinForms.Templates
```

maal@CHREYNB0085 ➤ C:\Work\Playground

[14:19]

> dotnet new fabulous-xf-app -n Gnabber

Gnabber - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help

iPhoneSimulator Gnabber.iOS Simulator

Toolbox

Gnabber.fs # x

```
open Xamarin.Forms

module App =
    type Model =
        { Count : int
          Step : int
          TimerOn: bool }

    type Msg =
        | Increment
        | Decrement
        | Reset
        | SetStep of int
        | TimerToggled of bool
        | TimedTick

    let initModel = { Count = 0; Step = 1; TimerOn=false }

    let init () = initModel, Cmd.none

    let timerCmd = ...

    let update msg model =
        match msg with...

    let view (model: Model) dispatch =
        View.ContentPage(
            content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))

    // Note, this declaration is needed if you enable LiveUpdate
    let program = Program.mkProgram init update view

type App () as app =
    inherit Application ()

    let runner =
        App.program
    #if DEBUG
    |> Program.withConsoleTrace
    #endif
    |> Program.runWithDynamicView app

    #if DEBUG
    |> Program.enableLiveUpdate()
    #endif

    // Uncomment this code to save the application state to app.Properties using Newtonsoft.Json
    // See https://fsprojects.github.io/Fabulous/tools.html for further instructions.

```

110 %

Error List

Entire Solution 0 Errors 0 Warnings 0 Messages Build + IntelliSense

Solution Explorer

Search Solution Explorer (Ctrl+U)

Solution 'Gnabber' (3 projects)

- GNabber
- Dependencies
- NuGet
- SDK
- Gnabber.fs

Gnabber.Android

Gnabber.iOS

Solution Explorer Team Explorer

Properties

Ready Data Tools Operations Package Manager Console Error List Output F# Interactive

Ln 37 Col 26 Ch 57 INS

Add to Source Control

@mallibone

Gnabber - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help

iPhoneSimulator Gnabber.iOS Simulator

Toolbox

Gnabber.fs # x

```
open Xamarin.Forms

module App =
    type Model =
        { Count : int
          Step : int
          TimerOn: bool }

    type Msg =
        | Increment
        | Decrement
        | Reset
        | SetStep of int
        | TimerToggled of bool
        | TimedTick

    let initModel = { Count = 0; Step = 1; TimerOn=false }

    let init () = initModel, Cmd.none

    let timerCmd = ...

    let update msg model =
        match msg with...

    let view (model: Model) dispatch =
        View.ContentPage(
            content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))

    // Note, this declaration is needed if you enable LiveUpdate
    let program = Program.mkProgram init update view

type App () as app =
    inherit Application ()

    let runner =
        App.program
    #if DEBUG
    |> Program.withConsoleTrace
    #endif
    |> Program.runWithDynamicView app

    #if DEBUG
    |> Program.enableLiveUpdate()
    #endif

    // Uncomment this code to save the application state to app.Properties using Newtonsoft.Json
    // See https://fsprojects.github.io/Fabulous/tools.html for further instructions.

```

110 %

Error List

Entire Solution 0 Errors 0 Warnings 0 Messages Build + IntelliSense

Solution Explorer

Search Solution Explorer (Ctrl+U)

Solution 'Gnabber' (3 projects)

- GNabber
- Dependencies
- NuGet
- SDK

F# Gnabber.fs

Gnabber.Android

Gnabber.iOS

Solution Explorer Team Explorer

Properties

Ready Data Tools Operations Package Manager Console Error List Output F# Interactive

Ln 37 Col 26 Ch 57 INS

Add to Source Control

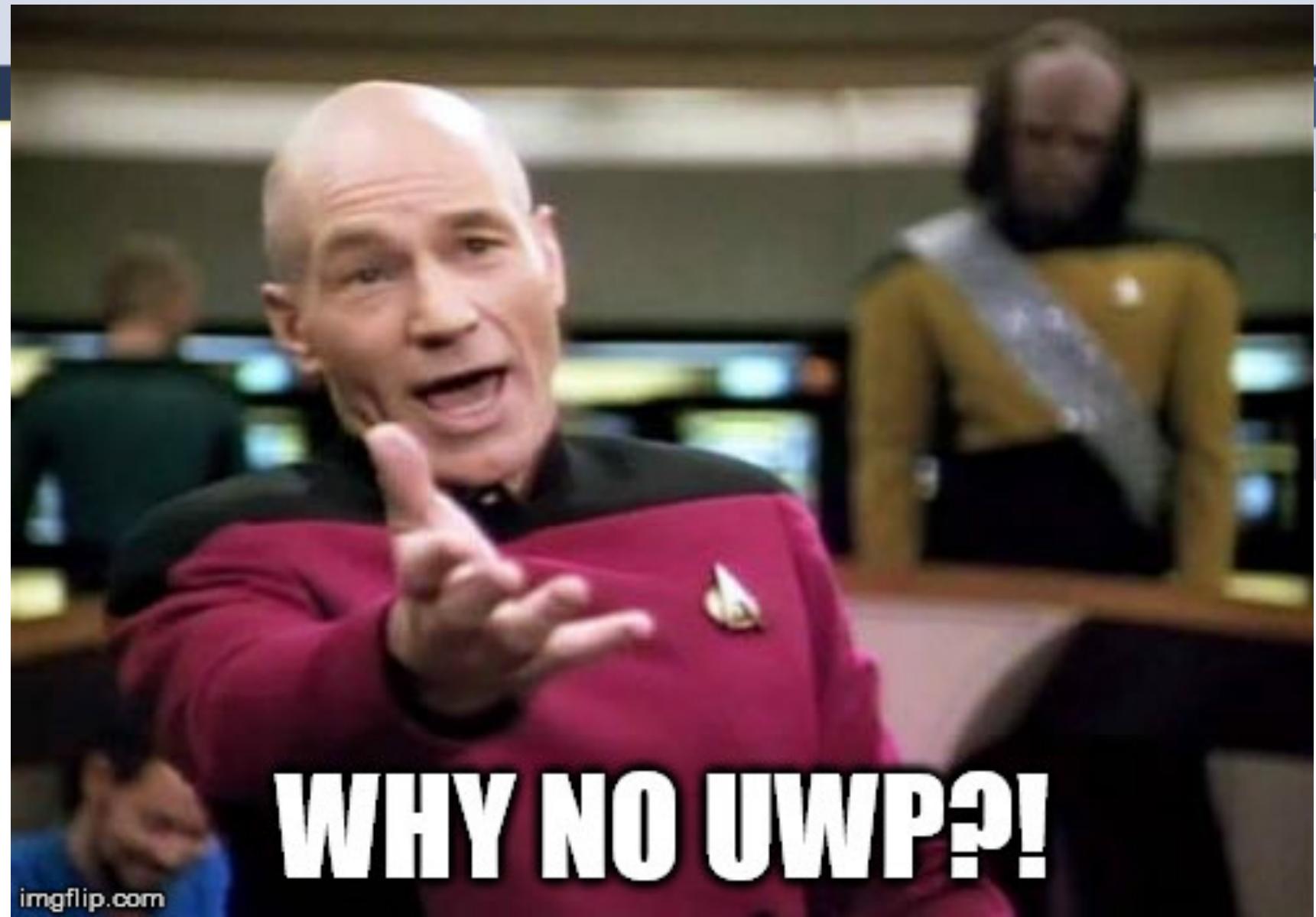


Quick Launch



Mark Allibone

MA



imgflip.com



Notifications



Gnabber.fs X

```
7  23 [open Xamarin.Forms
8  22
9  21 [module App =
10 20 [type Model =
11 19 [{ Count : int
12 18 [Step : int
13 17 [TimerOn: bool }
14 16
15 15 [type Msg =
16 14 [| Increment
17 13 [| Decrement
18 12 [| Reset
19 11 [| SetStep of int
20 10 [| TimerToggled of bool
21 9 [| TimedTick
22 8
23 7 [let initModel = { Count = 0; Step = 1; TimerOn=false }
24 6
25 5 [let init () = initModel, Cmd.none
26 4
27 3 [let timerCmd = ...
28 2
29 1 [let update msg model =
30 0 [match msg with ...]
31 1
32 2 [let view (model: Model) dispatch =
33 3 [View.ContentPage(content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
34 4 [5
35 6 [// Note, this declaration is needed if you enable LiveUpdate
36 7 [let program = Program.mkProgram init update view
37 8
38 9 [type App () as app =
```



Gnabber.fs X

```
7 23 [open Xamarin.Forms
8 22
9 21 [module App =
10 20 [type Model =
11 19 { Count : int
12 18 | Step : int
13 17 | TimerOn: bool }
14 16
15 15 [type Msg =
16 14 | Increment
17 13 | Decrement
18 12 | Reset
19 11 | SetStep of int
20 10 | TimerToggled of bool
21 9 | TimedTick
22 8
23 7 let initModel = { Count = 0; Step = 1; TimerOn=false }
24 6
25 5 let init () = initModel, Cmd.none
26 4
27 3 [let timerCmd = ...
28 2
29 1 [let update msg model =
30 0 [match msg with...]
31 1
32 2 [let view (model: Model) dispatch =
33 3 View.ContentPage(
34 4 content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
35 5
36 6 // Note, this declaration is needed if you enable LiveUpdate
37 7 let program = Program.mkProgram init update view
38 8
39 9 [type App () as app =
```



Gnabber.fs X

```
7  23 [open Xamarin.Forms
8  22
9  21 [module App =
10 20 [type Model =
11 19 [{ Count : int
12 18 [    Step : int
13 17 [    | TimerOn: bool }
14 16
15 15 [type Msg =
16 14 [| Increment
17 13 [| Decrement
18 12 [| Reset
19 11 [| SetStep of int
20 10 [| TimerToggled of bool
21 9 [| TimedTick
22 8
23 7 [let initModel = { Count = 0; Step = 1; TimerOn=false }
24 6
25 5 [let init () = initModel, Cmd.none
26 4
27 3 [let timerCmd = ...
28 2
29 1 [let update msg model =
30 0 [    match msg with ...
31 1
32 2
33 1 [let view (model: Model) dispatch =
34 0 [    View.ContentPage(
35 1 [        content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
36 0
37 1 [// Note, this declaration is needed if you enable LiveUpdate
38 0
39 1 [let program = Program.mkProgram init update view
40 0
41 1 [type App () as app =
```



Gnabber.fs X

```
7  23 [open Xamarin.Forms
8  22
9  21 [module App =
10 20 [type Model =
11 19 [{ Count : int
12 18 [    Step : int
13 17 [    | TimerOn: bool }
14 16
15 15 [type Msg =
16 14 [| Increment
17 13 [| Decrement
18 12 [| Reset
19 11 [| SetStep of int
20 10 [| TimerToggled of bool
21 9 [| TimedTick
22 8
23 7 [let initModel = { Count = 0; Step = 1; TimerOn=false }
24 6
25 5 [let init () = initModel, Cmd.none
26 4
27 3 [let timerCmd = ...
28 2
29 1 [let update msg model =
30 0 [match msg with ...
31 1
32 2 [let view (model: Model) dispatch =
33 1 [    View.ContentPage(
34 0 [        content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
35 5
36 6 [// Note, this declaration is needed if you enable LiveUpdate
37 7 [let program = Program.mkProgram init update view
38 8
39 9 [type App () as app =
```



Gnabber.fs X

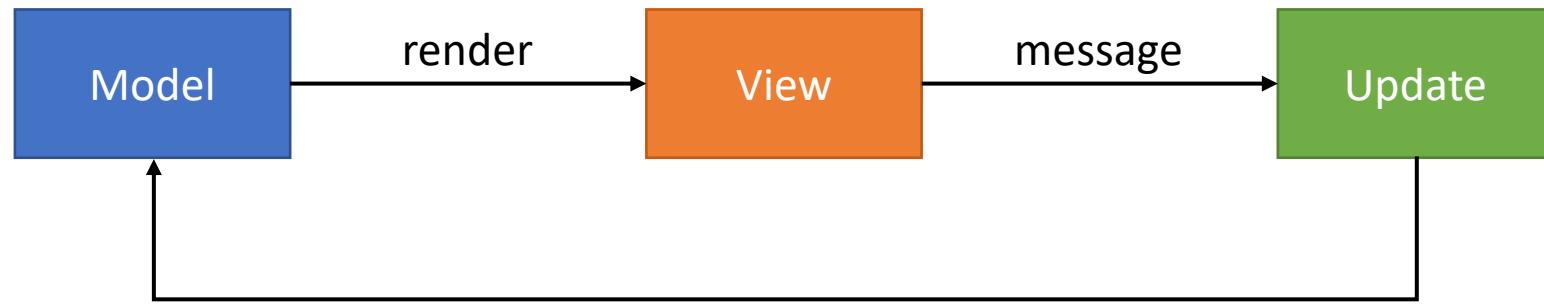
```
7  23 [open Xamarin.Forms
8  22
9  21 [module App =
10 20 [type Model =
11 19 [{ Count : int
12 18 [Step : int
13 17 [TimerOn: bool }
14 16
15 15 [type Msg =
16 14 [| Increment
17 13 [| Decrement
18 12 [| Reset
19 11 [| SetStep of int
20 10 [| TimerToggled of bool
21 9 [| TimedTick
22 8
23 7 [let initModel = { Count = 0; Step = 1; TimerOn=false }
24 6
25 5 [let init () = initModel, Cmd.none
26 4
27 3 [let timerCmd = ...
28 2
29 1 [let update msg model =
30 0 [| match msg with ...
31 1
32 2 [let view (model: Model) dispatch =
33 3 [| View.ContentPage(
34 4 [| content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
35 5
36 6 [// Note, this declaration is needed if you enable LiveUpdate
37 7 [let program = Program.mkProgram init update view
38 8
39 9 [type App () as app =
```

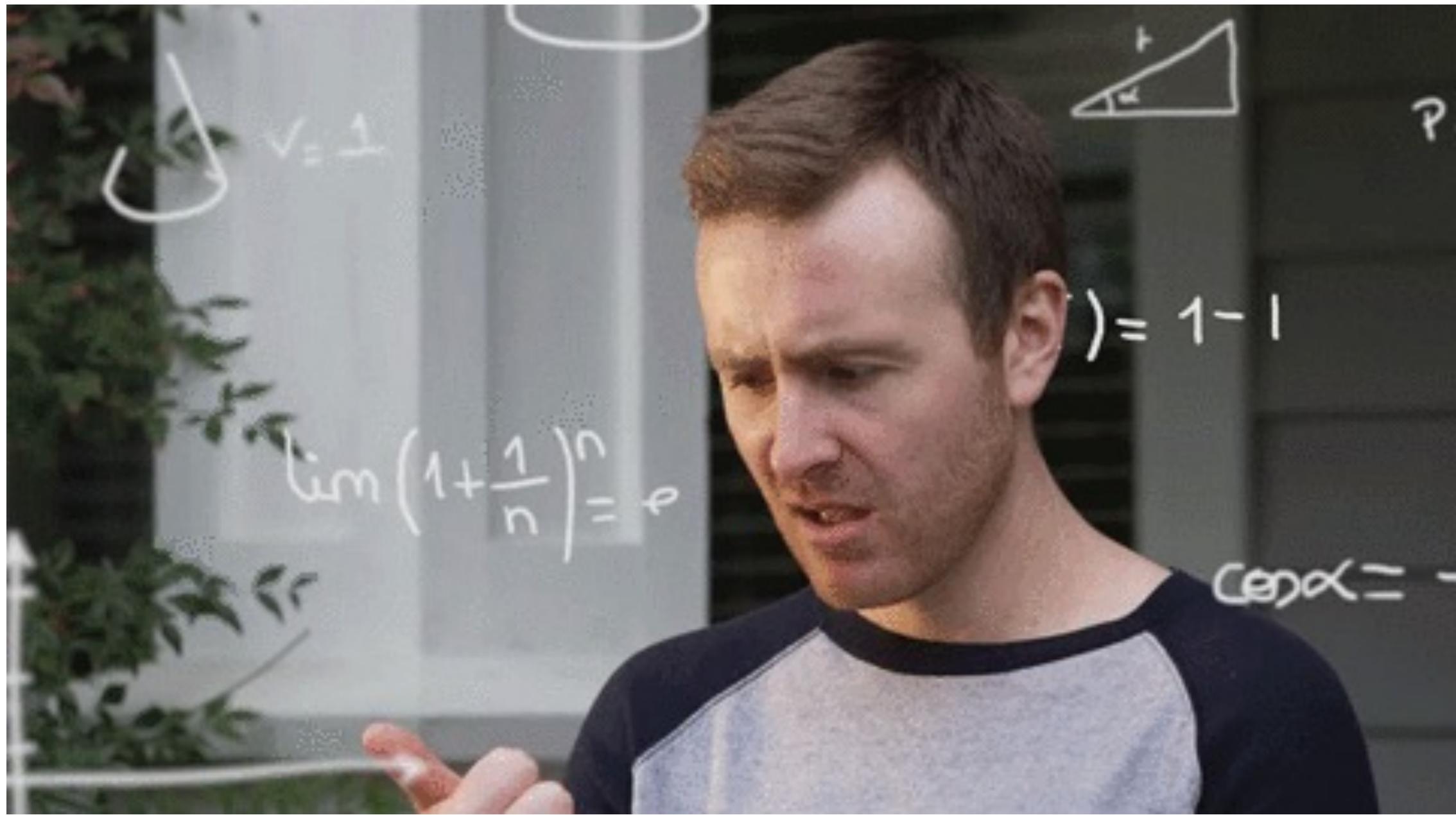


Gnabber.fs X

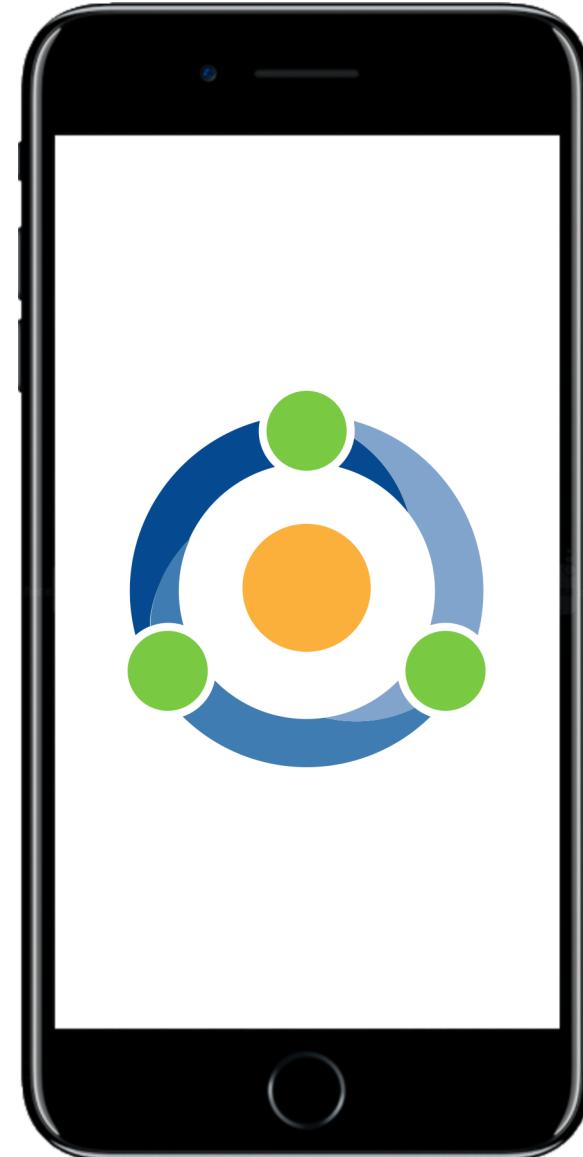
```
7  23 [open Xamarin.Forms
8  22
9  21 [module App =
10 20 [type Model =
11 19 [{ Count : int
12 18 [    Step : int
13 17 [    TimerOn: bool }
14 16
15 15 [type Msg =
16 14 [    | Increment
17 13 [    | Decrement
18 12 [    | Reset
19 11 [    | SetStep of int
20 10 [    | TimerToggled of bool
21 9 [    | TimedTick
22 8
23 7 [let initModel = { Count = 0; Step = 1; TimerOn=false }
24 6
25 5 [let init () = initModel, Cmd.none
26 4
27 3 [let timerCmd = ...
28 2
29 1 [let update msg model =
30 0 [    match msg with... ]
31 1
32 2 [let view (model: Model) dispatch =
33 3 [    View.ContentPage(
34 4 [        content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
35 5
36 6 [// Note, this declaration is needed if you enable LiveUpdate
37 7 [let program = Program.mkProgram init update view
38 8
39 9 [type App () as app =
```

Model View Update





- Driven from the model
- All state changes are defined with messages
- Single place for change - the update method
- Uses Xamarin Forms 



FASCINATING

NOW MAKE IT PRETTY

imgflip.com

@mallibone



CSS

Doing it with style



@mallibone



XAML vs Coded UI

The background of the image is a close-up photograph of ink swirling in water against a black background. The ink forms large, billowing clouds in shades of orange, yellow, and purple. The colors are most concentrated on the left side of the frame, creating a sense of motion and depth.

Demo

@mallibone

- Interactive Development and Designing
- You can create good looking UIs
- Create reusable UI Components





The background of the slide features a vibrant, abstract pattern of ink swirling in water against a black background. The colors include deep purple, bright orange, yellow, and white, creating a dynamic and organic feel.

Live Update ❤️ Coded UI

<https://fsprojects.github.io/Fabulous/tools.html>

@mallibone



Photo by [Josh Hild](#) from [Pexels](#)

@mallibone



```
let timerCmd =
    async { do! Async.Sleep 200
            return TimedTick }
|> Cmd.ofAsyncMsg

let update msg model =
    match msg with
    // other messages
    | TimerToggled on -> { model with TimerOn = on }, (if on then timerCmd else Cmd.none)
    | TimedTick ->
        if model.TimerOn then
            { model with Count = model.Count + model.Step }, timerCmd
        else
            model, Cmd.none
```



```
let timerCmd =
    async { do! Async.Sleep 200
            return TimedTick }
|> Cmd.ofAsyncMsg
```

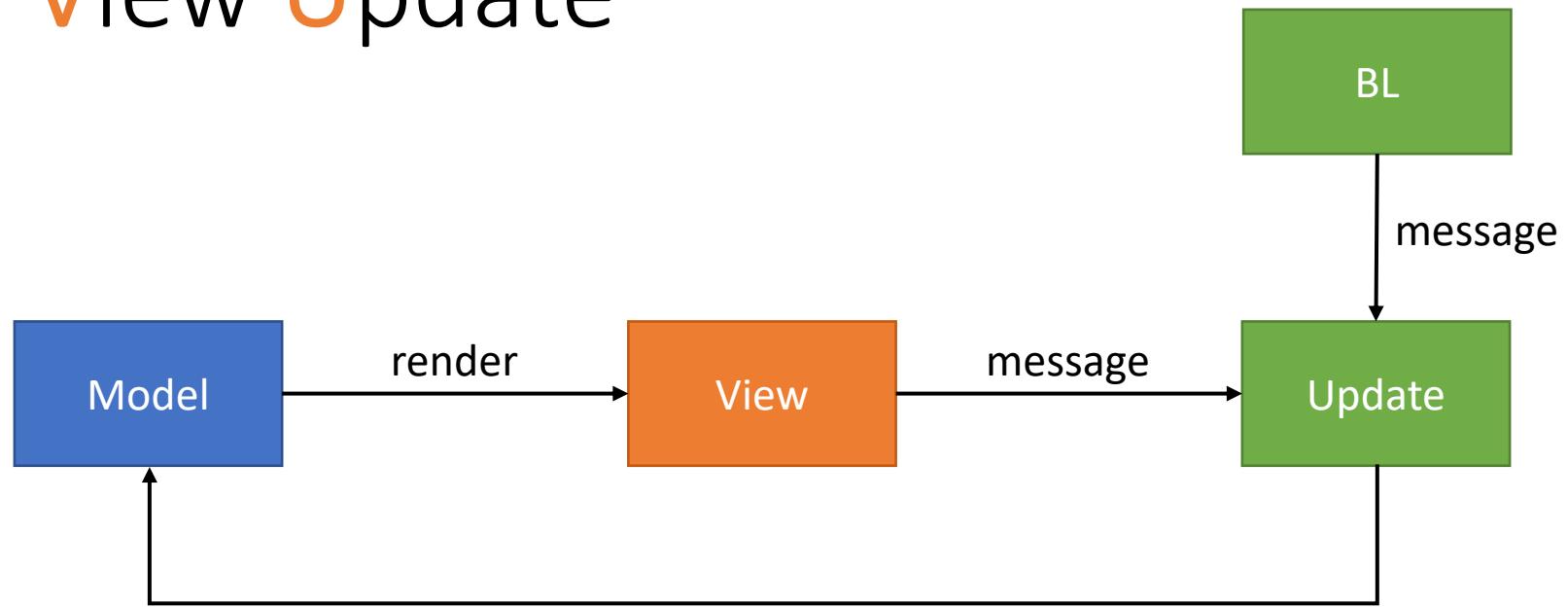
```
let update msg model =
    match msg with
    // other messages
    | TimerToggled on -> { model with TimerOn = on }, (if on then timerCmd else Cmd.none)
    | TimedTick ->
        if model.TimerOn then
            { model with Count = model.Count + model.Step }, timerCmd
        else
            model, Cmd.none
```



```
let timerCmd =
    async { do! Async.Sleep 200
            return TimedTick }
|> Cmd.ofAsyncMsg

let update msg model =
    match msg with
    // other messages
    | TimerToggled on -> { model with TimerOn = on }, (if on then timerCmd else Cmd.none)
    | TimedTick ->
        if model.TimerOn then
            { model with Count = model.Count + model.Step }, timerCmd
        else
            model, Cmd.none
```

Model View Update



Demo



@mallibone

- Every state change is defined
- All changes run through the update method
- Update processes messages in sequence



Camera



Google



Mail

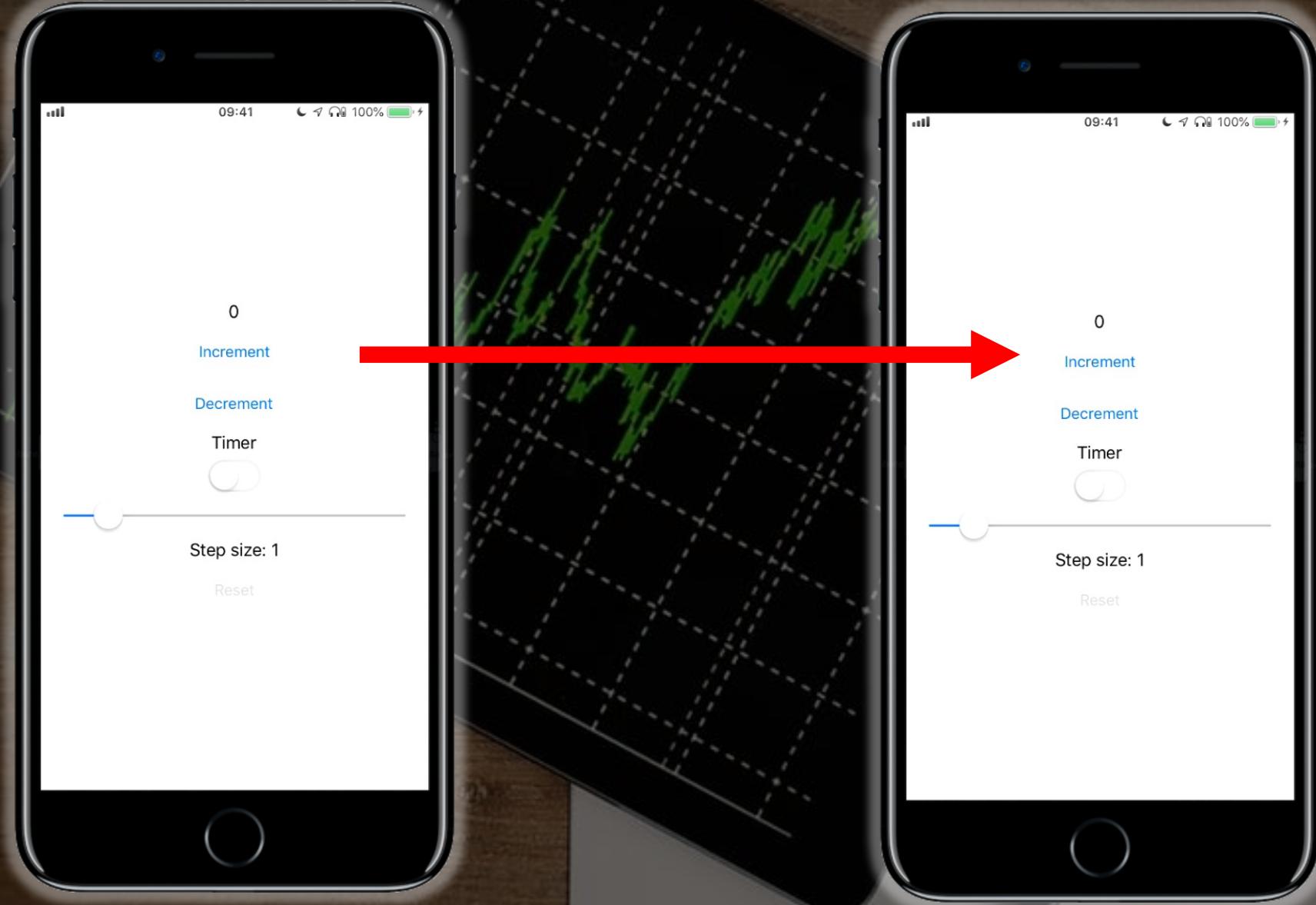


Phone



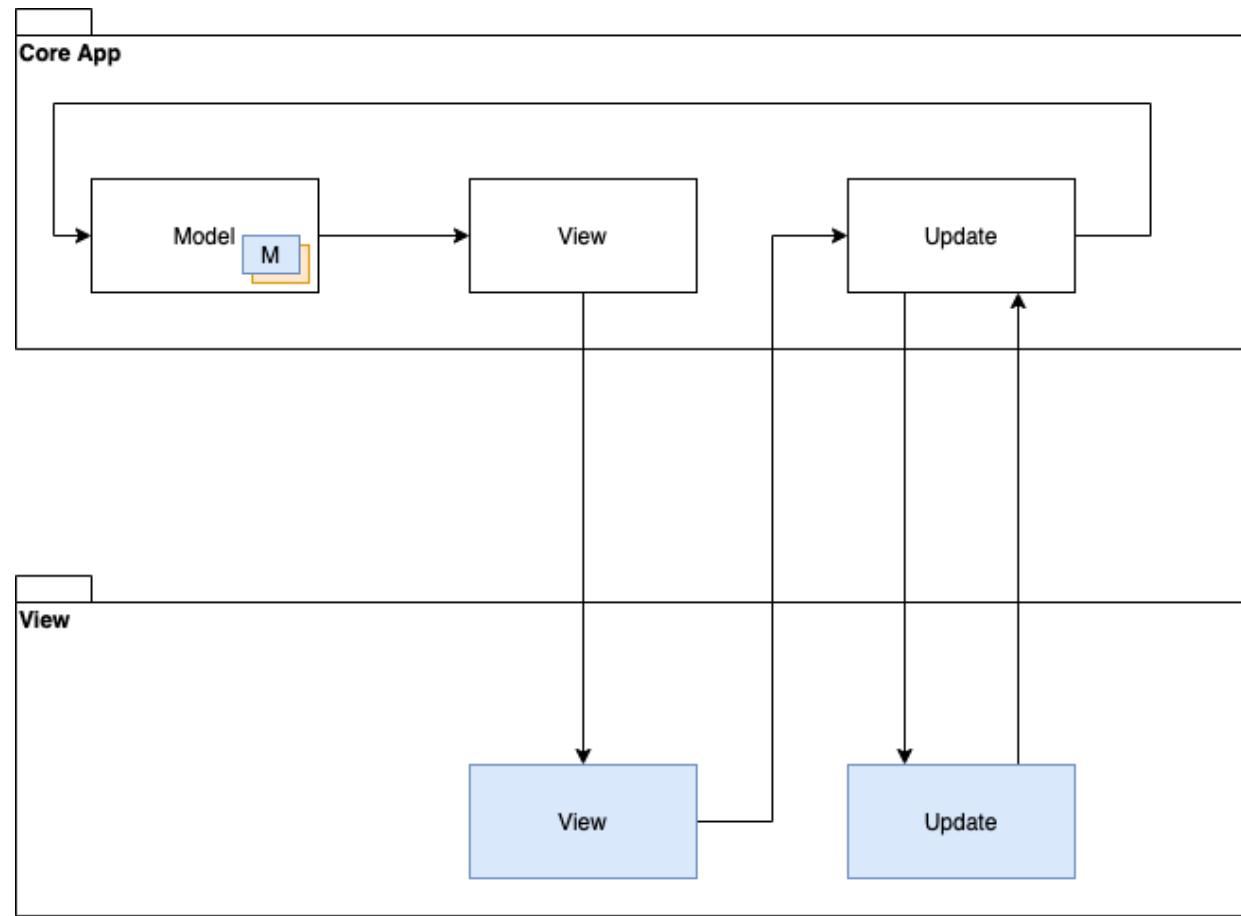
W

@mallibone



@mallibone

MVU with multiple pages



Demo



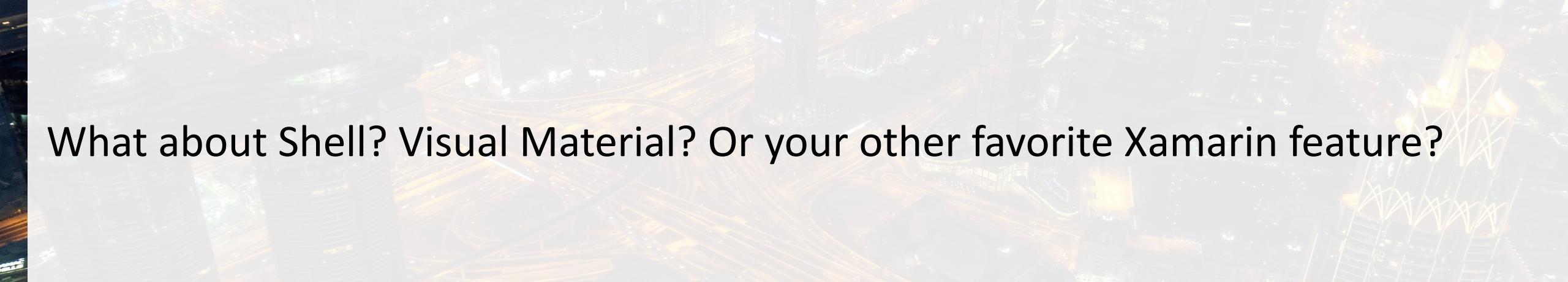
@mallibone

- The state of the app is stored in the model of the app
- Navigation Pages are updated in the core app view
- All updates pass through the core app update method





What about Shell? Visual Material? Or your other favorite Xamarin feature?



@mallbone



NuGet

@mallibone

Xamarin.Essentials

02/26/2020 • 2 minutes to read • +3

Xamarin.Essentials provides developers with cross-platform APIs for their mobile applications.

Android, iOS, and UWP offer unique operating system and platform APIs that developers have access to all in C# leveraging Xamarin. Xamarin.Essentials provides a single cross-platform API that works with any Xamarin.Forms, Android, iOS, or UWP application that can be accessed from shared code no matter how the user interface is created.

Get Started with Xamarin.Essentials

Follow the [getting started guide](#) to install the **Xamarin.Essentials** NuGet package into your existing or new Xamarin.Forms, Android, iOS, or UWP projects.

Feature Guides

Follow the guides to integrate these Xamarin.Essentials features into your applications:

- [Accelerometer](#) – Retrieve acceleration data of the device in three dimensional space.
- [App Actions](#) – Get and set shortcuts for the application.
- [App Information](#) – Find out information about the application.
- [App Theme](#) – Detect the current theme requested for the application.
- [Barometer](#) – Monitor the barometer for pressure changes.
- [Battery](#) – Easily detect battery level, source, and state.
- [Clipboard](#) – Quickly and easily set or read text on the clipboard.
- [Color Converters](#) – Helper methods for System.Drawing.Color.
- [Compass](#) – Monitor compass for changes.
- [Connectivity](#) – Check connectivity state and detect changes.
- [Contacts](#) – Retrieve information about a contact on the device.
- [Detect Shake](#) – Detect a shake movement of the device.

Is this page helpful?

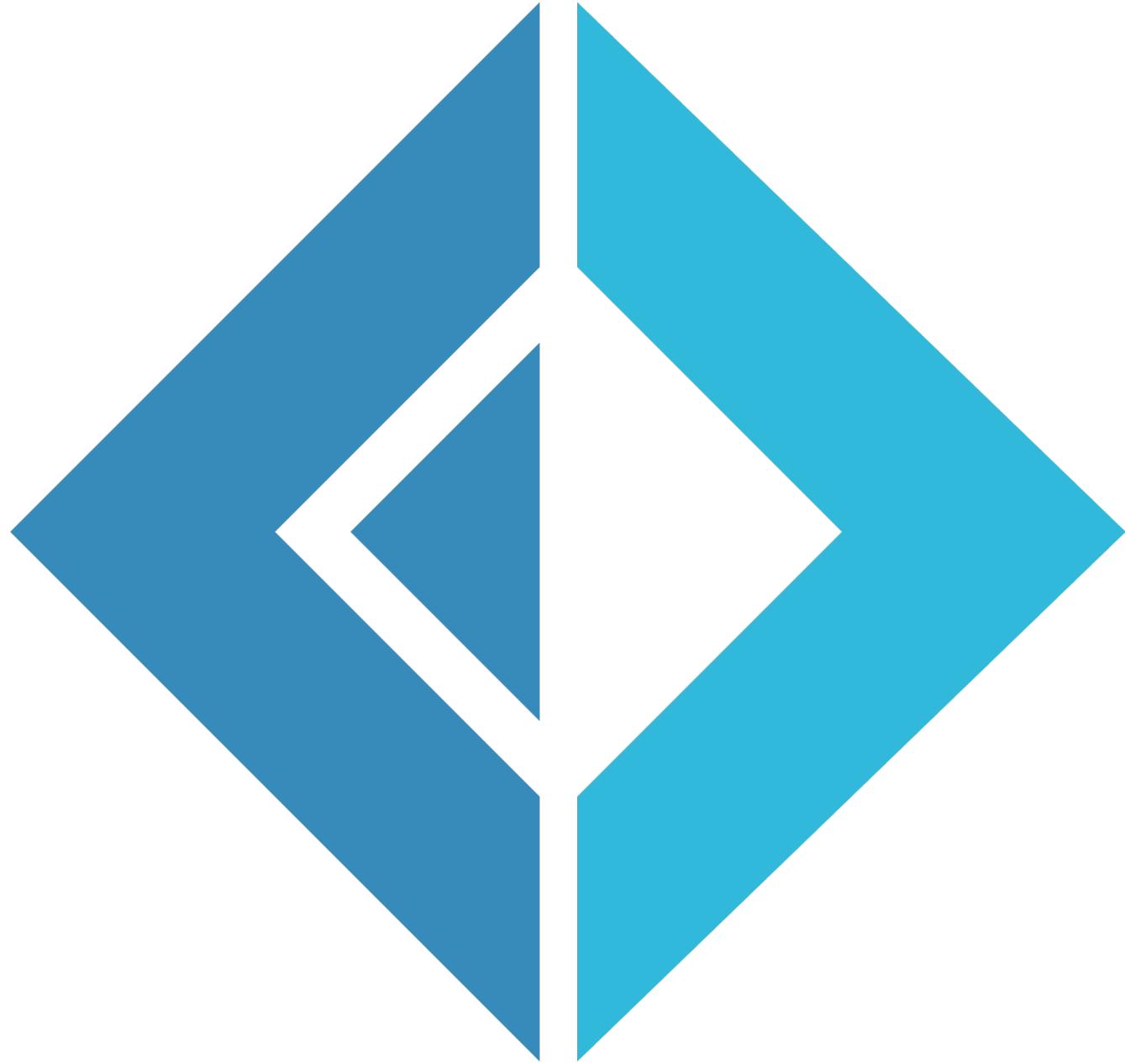
Yes No

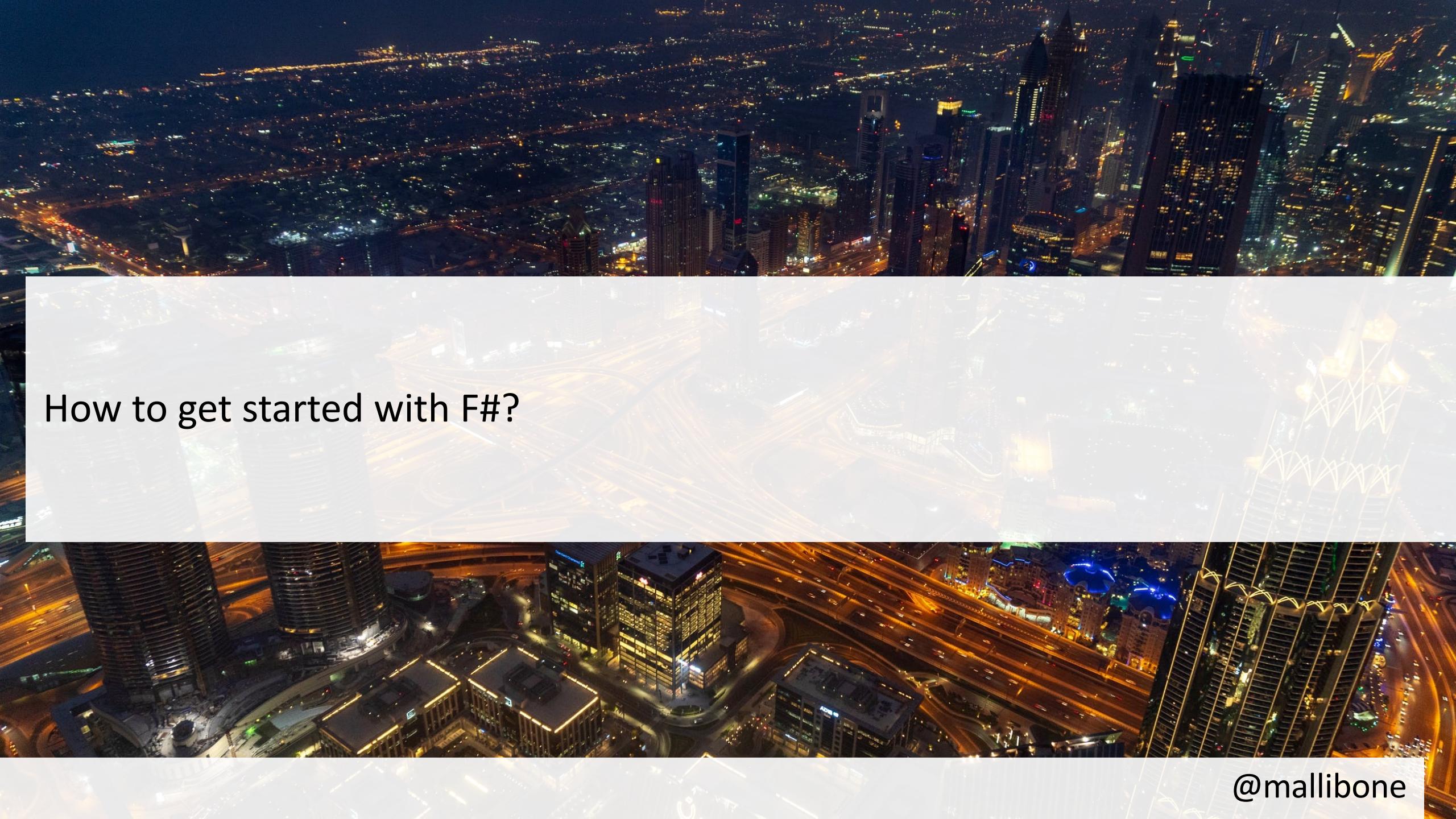
In this article

[Get Started with Xamarin.Essentials](#)
[Feature Guides](#)
[Troubleshooting](#)
[Xamarin.Essentials on Q&A](#)
[Release Notes](#)
[API Documentation](#)

[Download PDF](#)

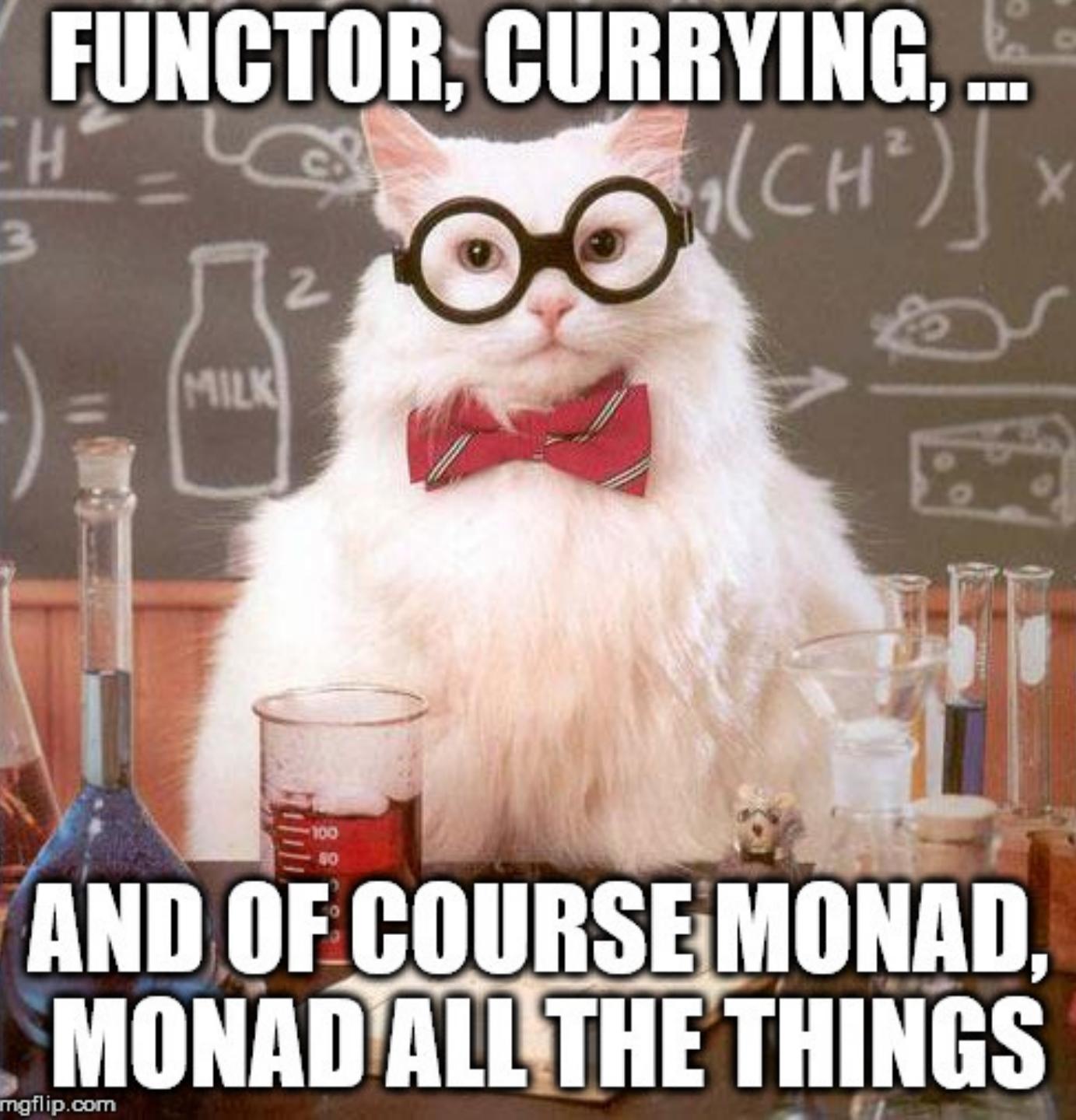
@mallibone





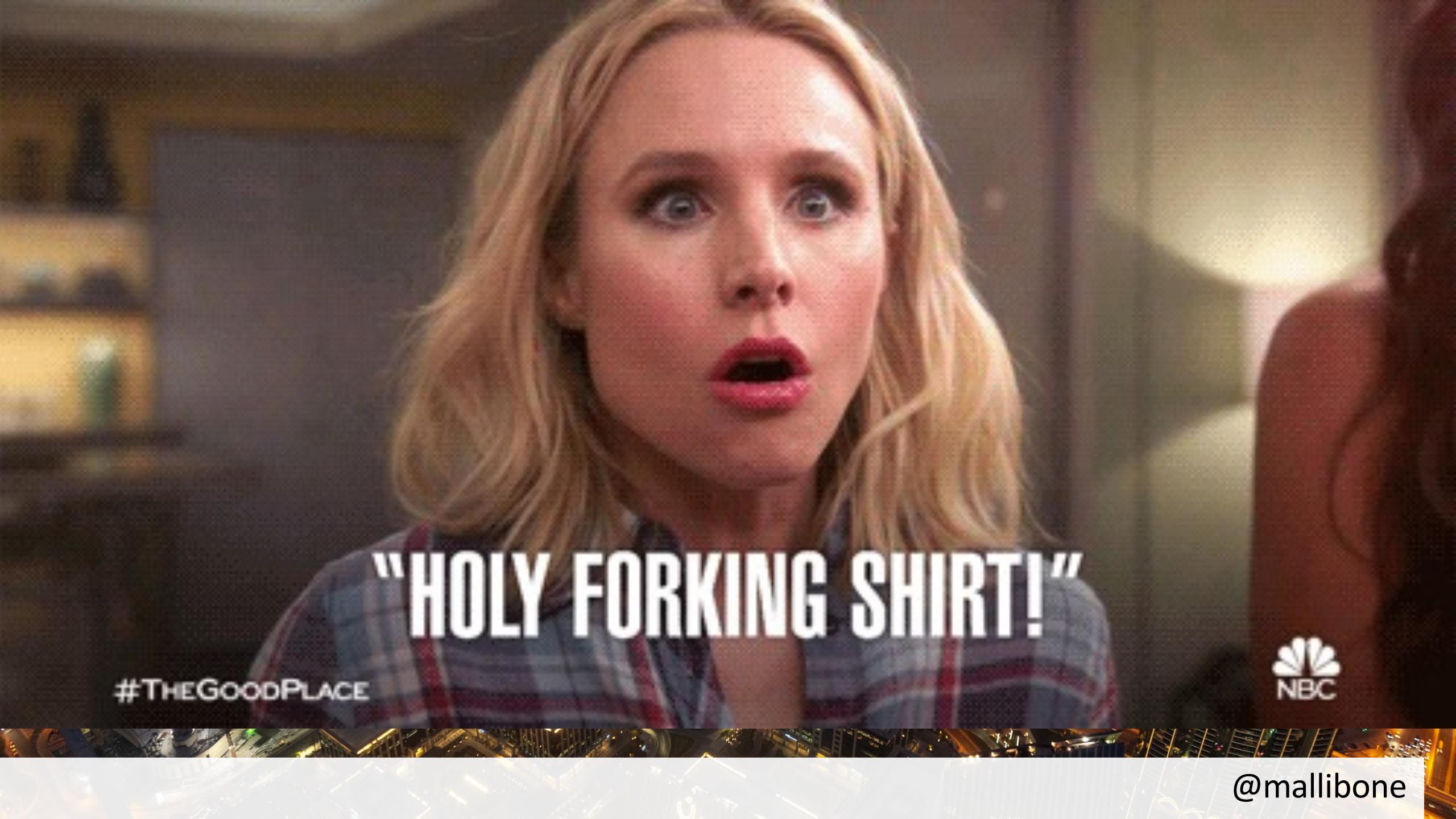
How to get started with F#?

@mallbone



FUNCTION, CURRYING,...

AND OF COURSE MONAD, MONAD ALL THE THINGS



"HOLY FORKING SHIRT!"

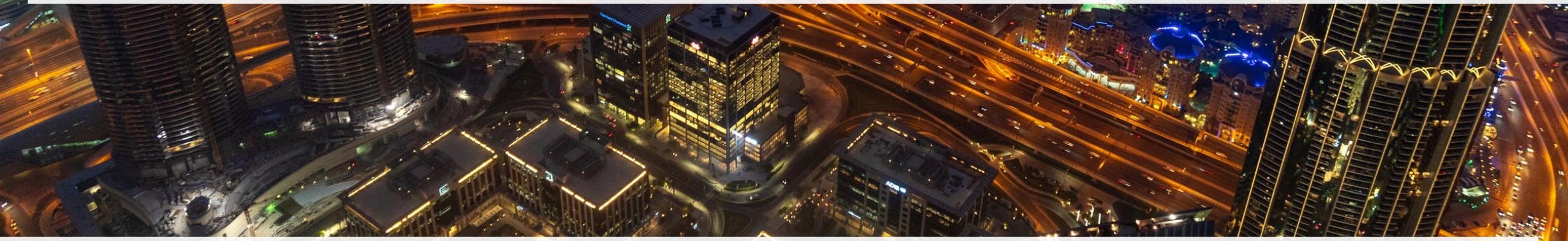
#THEGOODPLACE



@mallibone

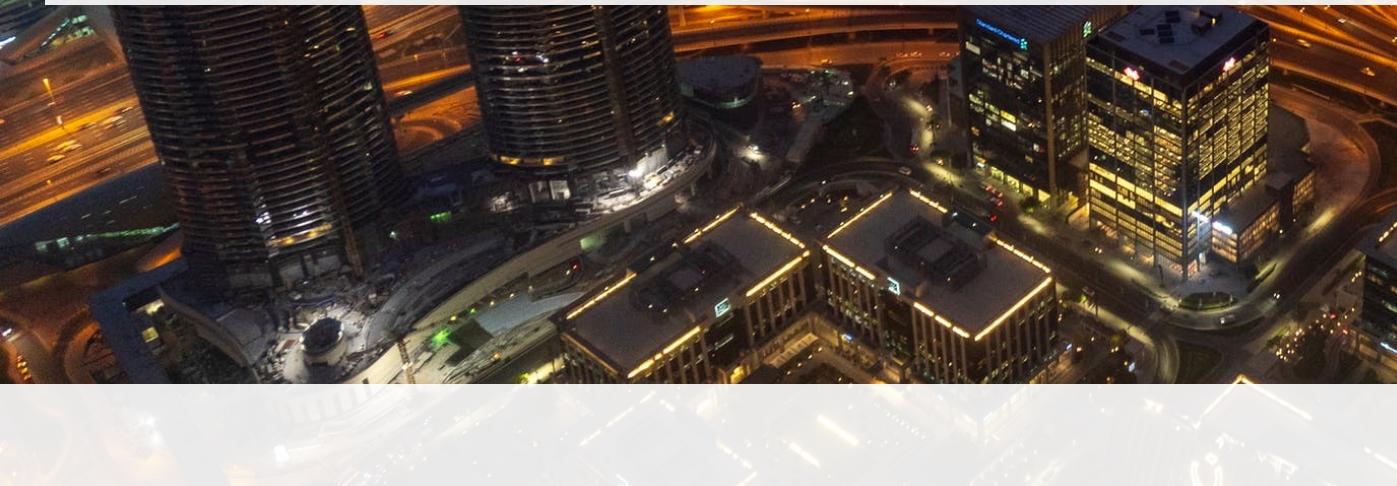


Learn Design patterns after learning the language



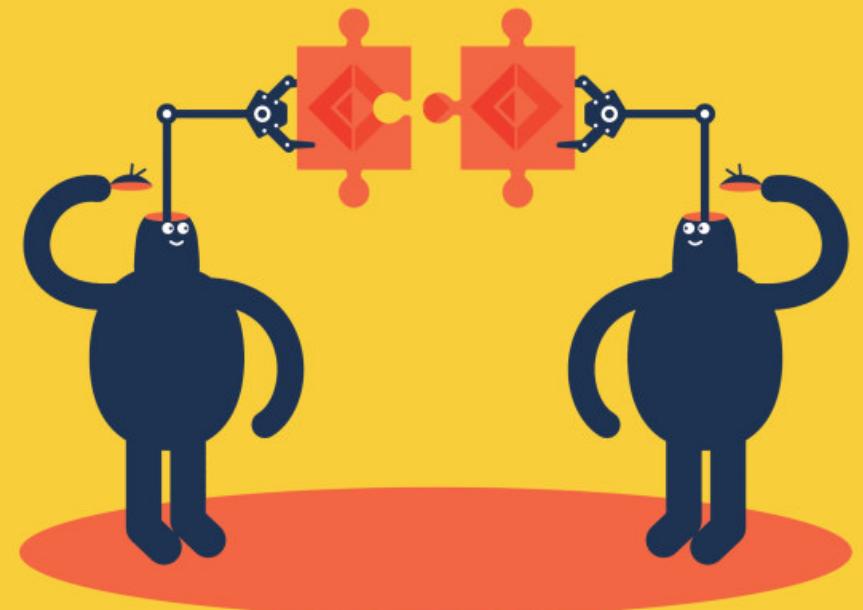


How to get started with F#?



GET PROGRAMMING WITH **F#**

A guide for .NET developers

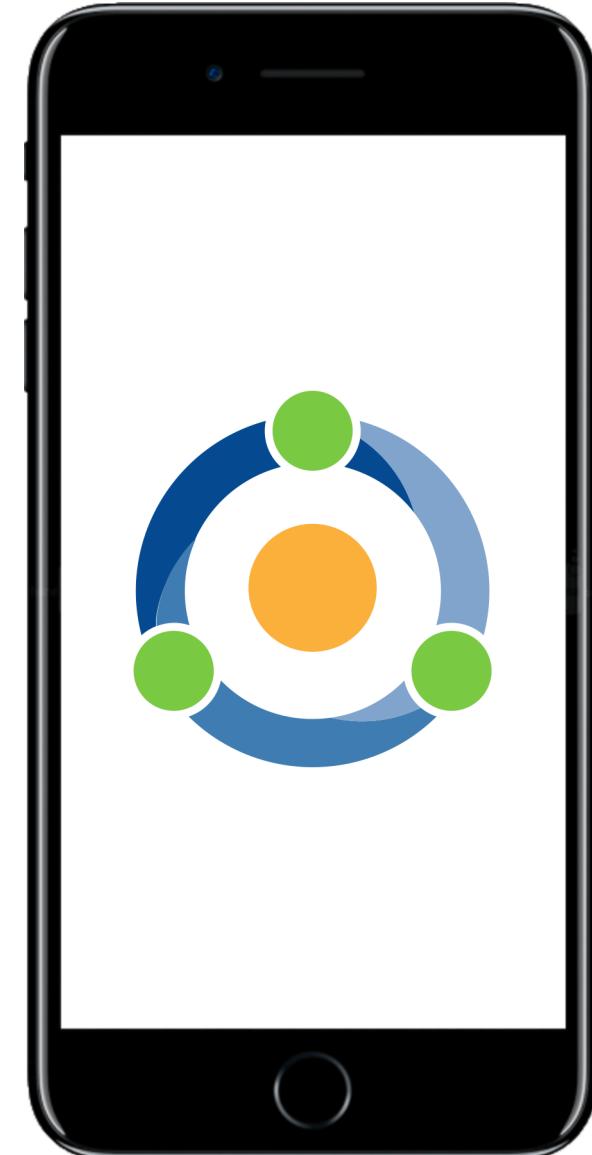


Isaac Abraham

Forewords
by Dustin Campbell
and Tomas Petricek

 MANNING

Takeaways



Takeaways

- F# is not scary it's just your friendly functional first .NET language
- Model View Update simplifies state handling in UI
- Fabulous is everywhere
- Have a Fabulous Day!





Thank you for your time!



Mark Allibone



@mallibone



Rey Technology



FEATURED
COMMUNITY
BLOG

<https://mallibone.com>



<https://nullpointers.io>



<https://fsprojects.github.io/Fabulous>



<https://fsharpforfunandprofit.com/>



<https://bit.ly/36kAYVv>



Microsoft®
Most Valuable
Professional