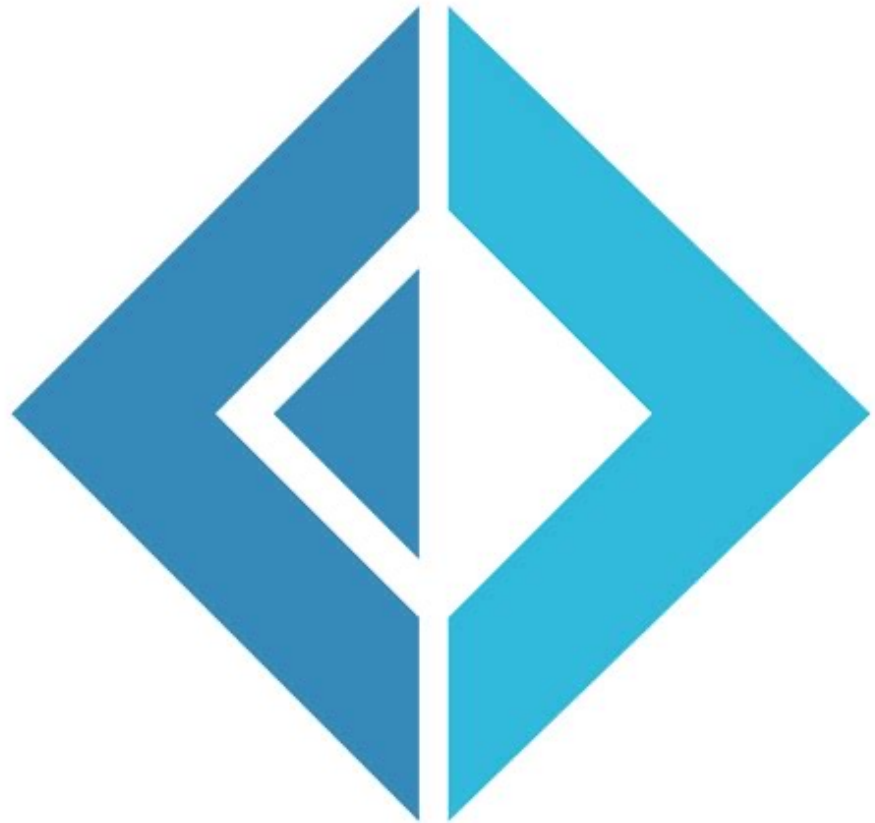
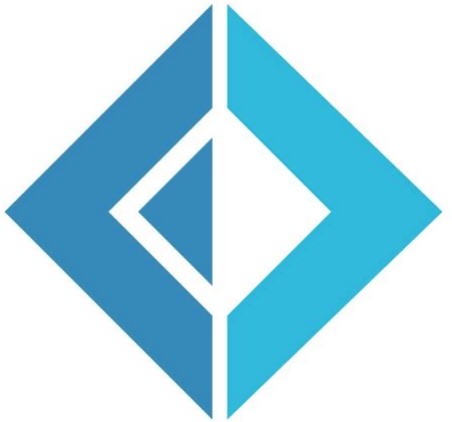

DWX
DEVELOPER WEEK '21
Fabulous Functional Frontends
Mark Allibone
Mobile Lead
Rey Technology
#dwx21
@mallibone



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

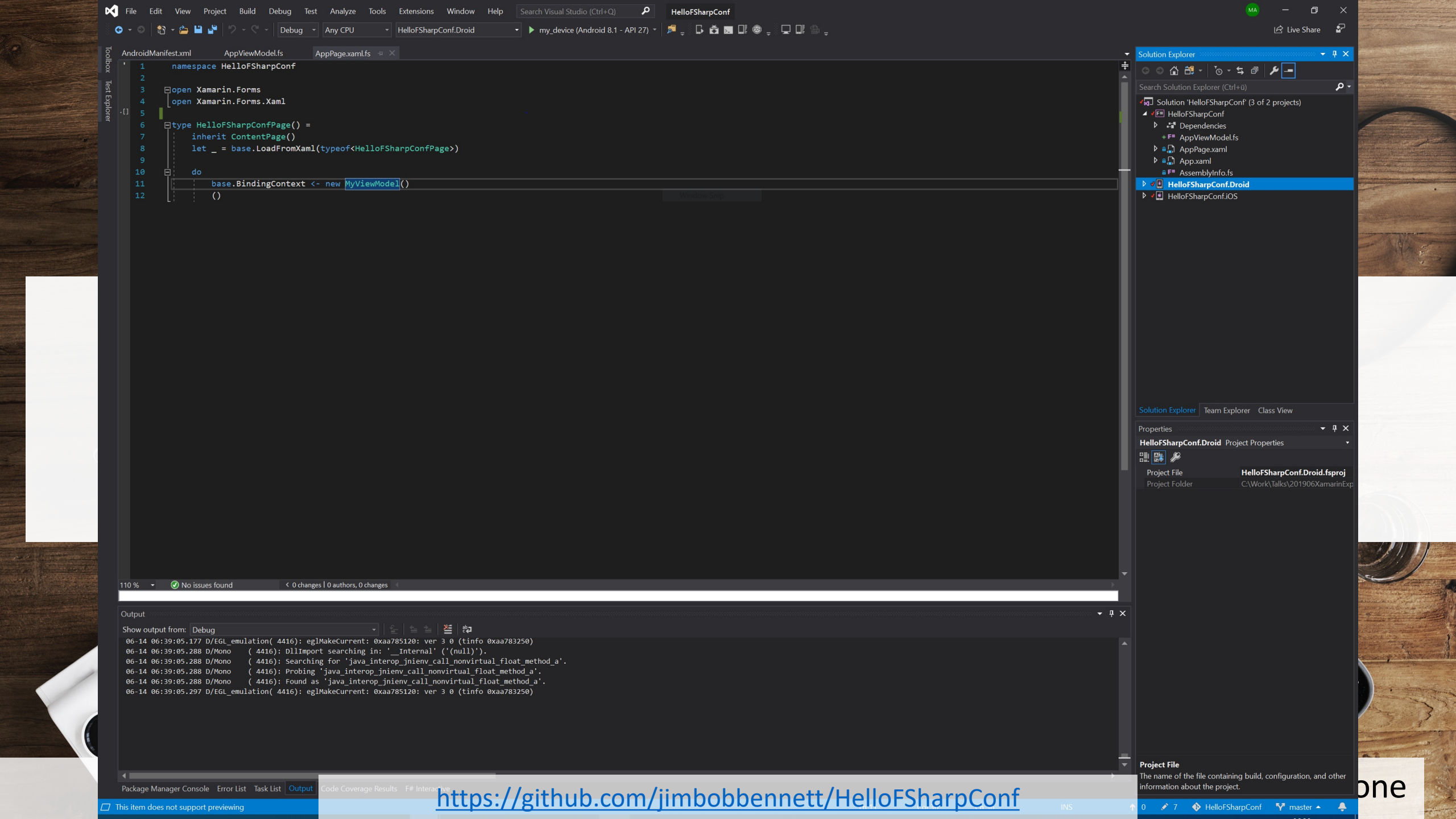


F# is great for BLOBs – overall a great general-purpose .NET language

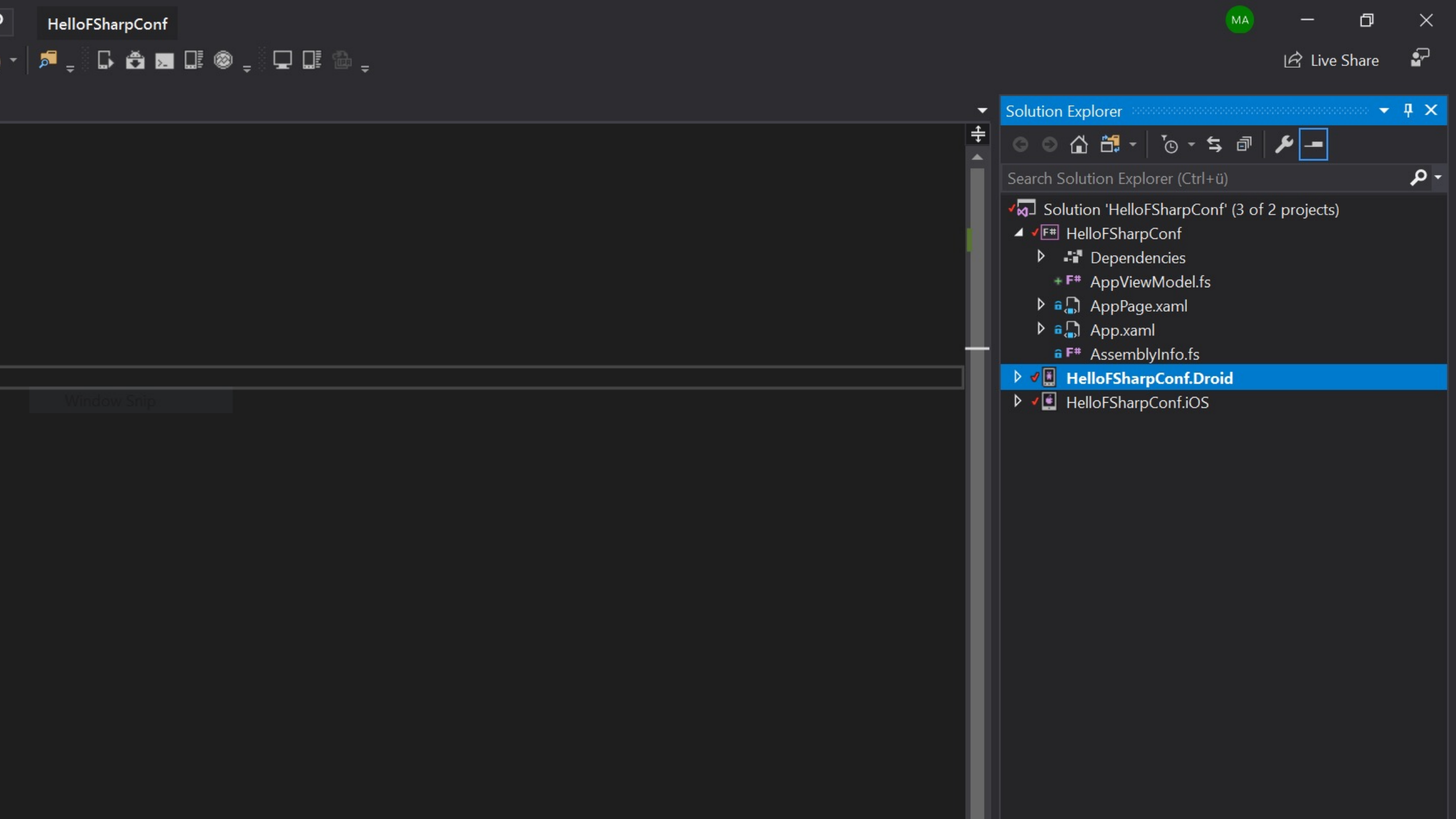
@mallibone

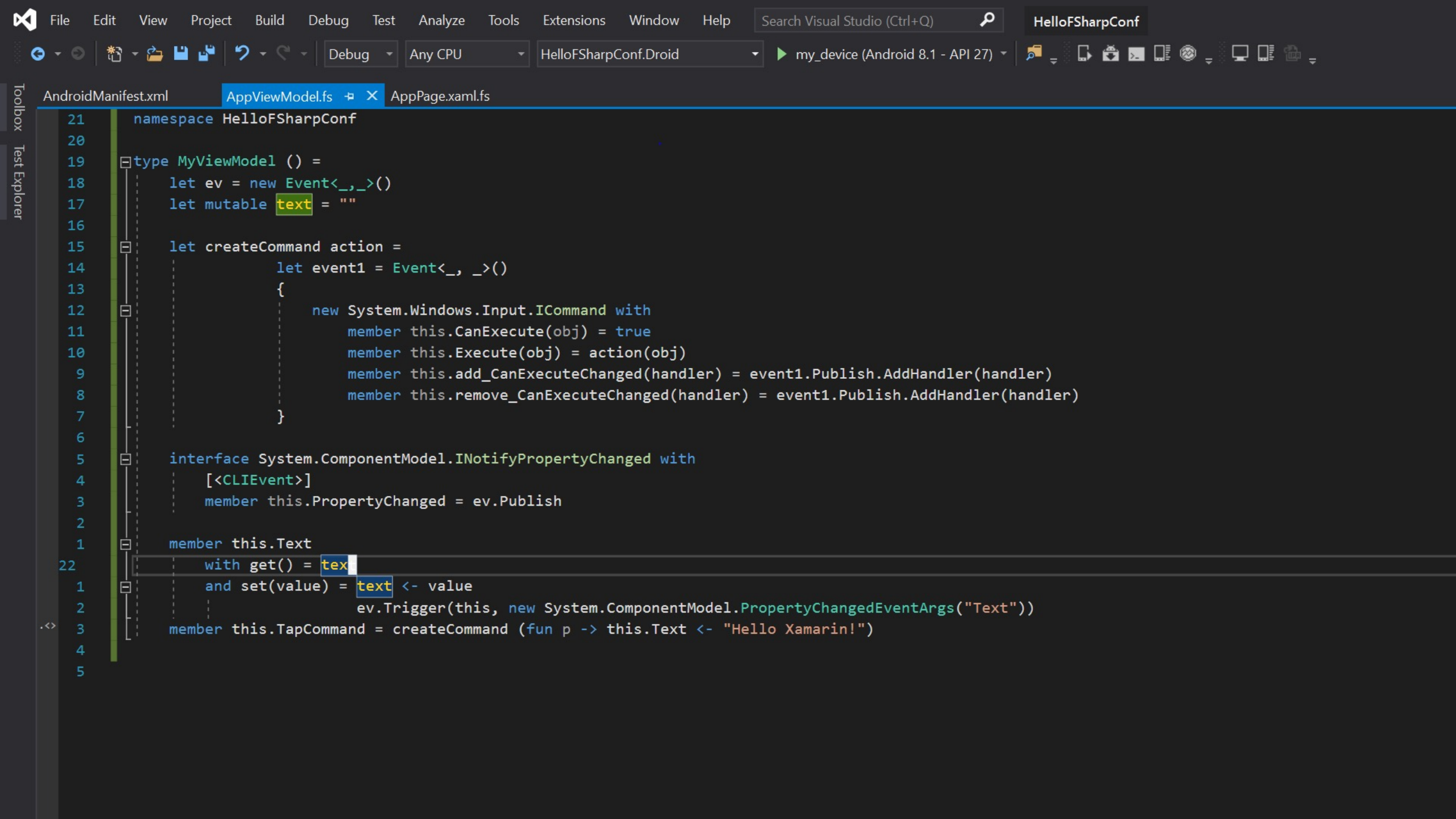
Why not just use Xamarin Forms?!

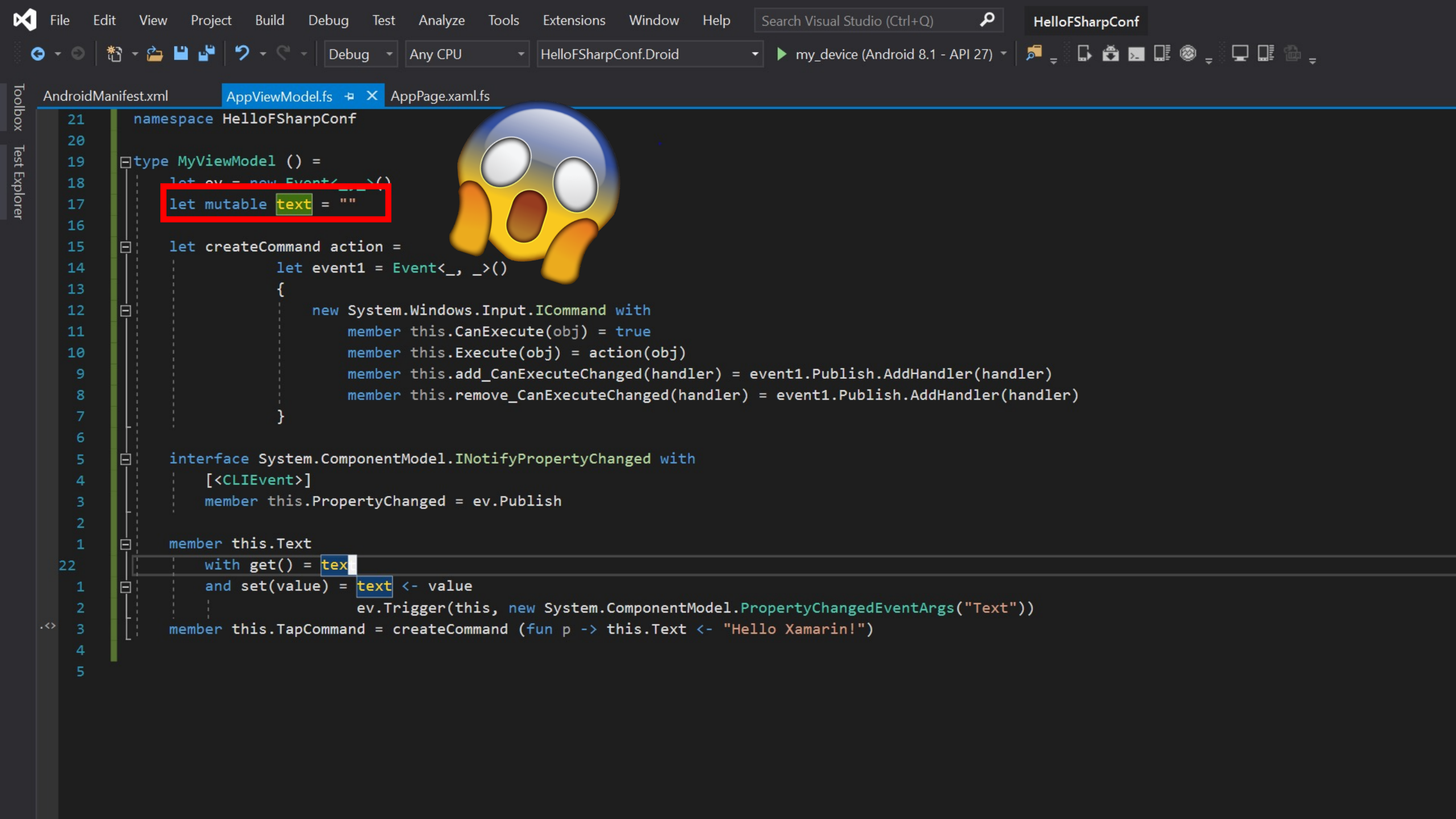
@mallibone

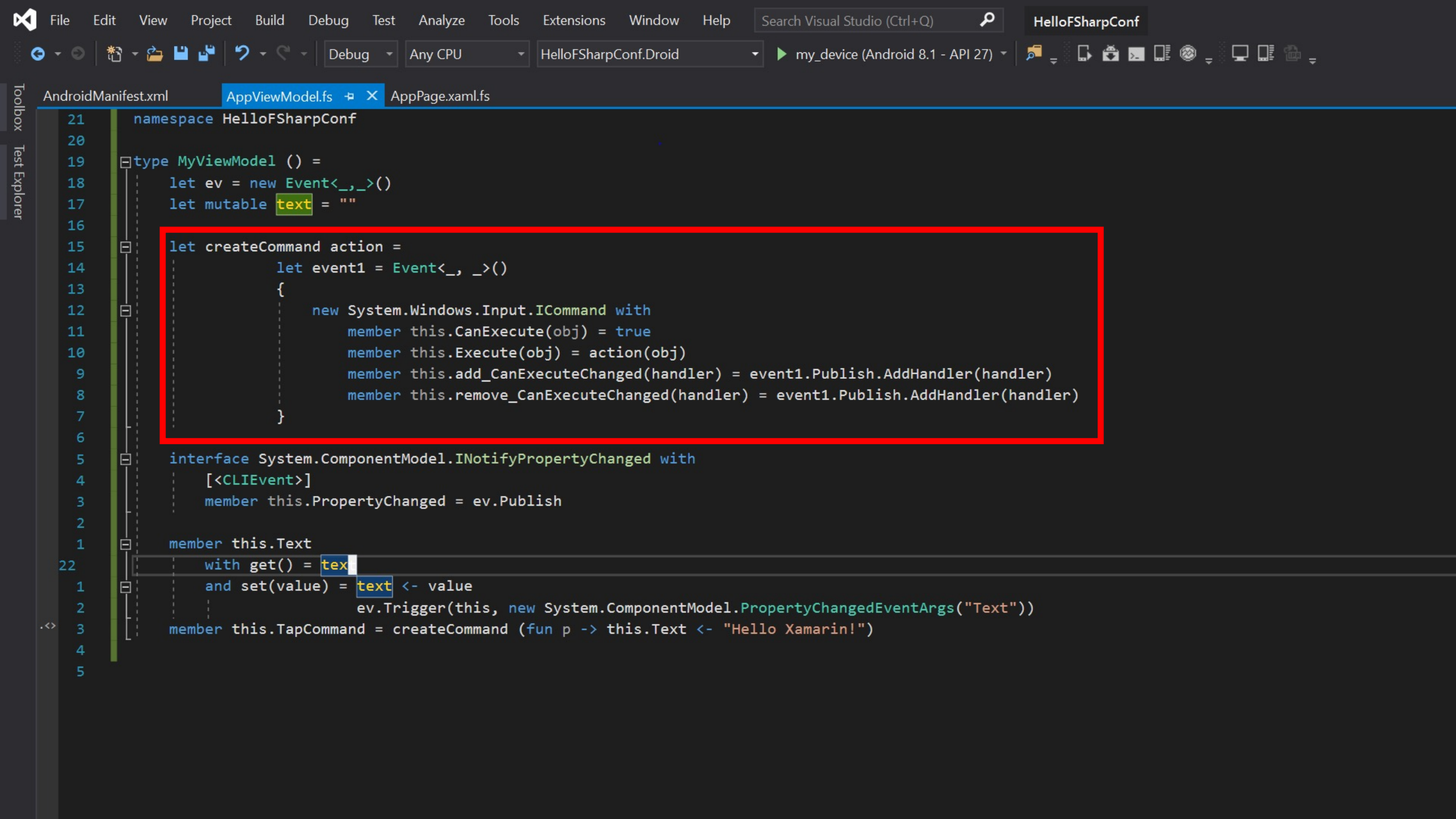


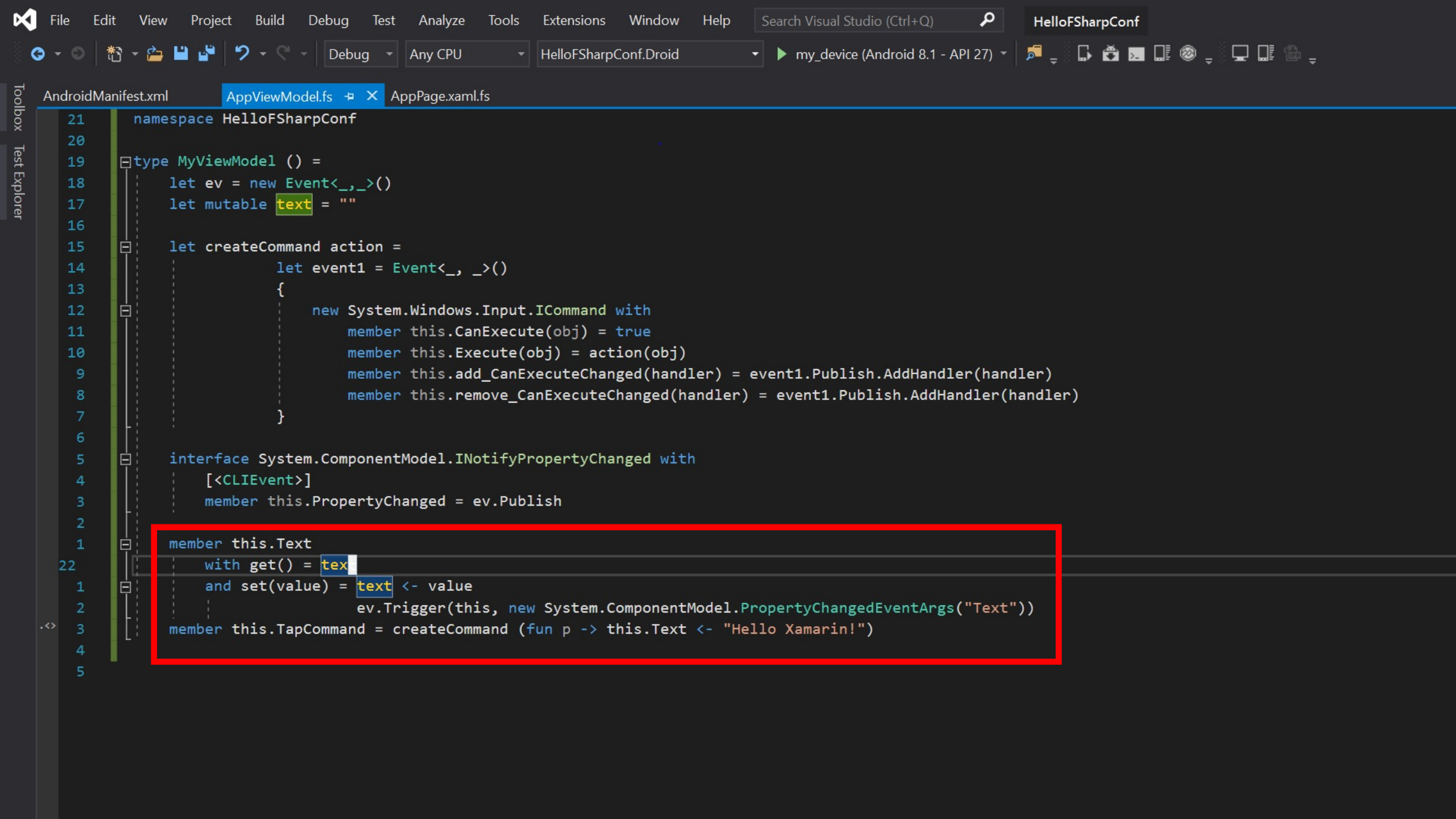
https://github.com/jimbobbennett/HelloFSharpConf

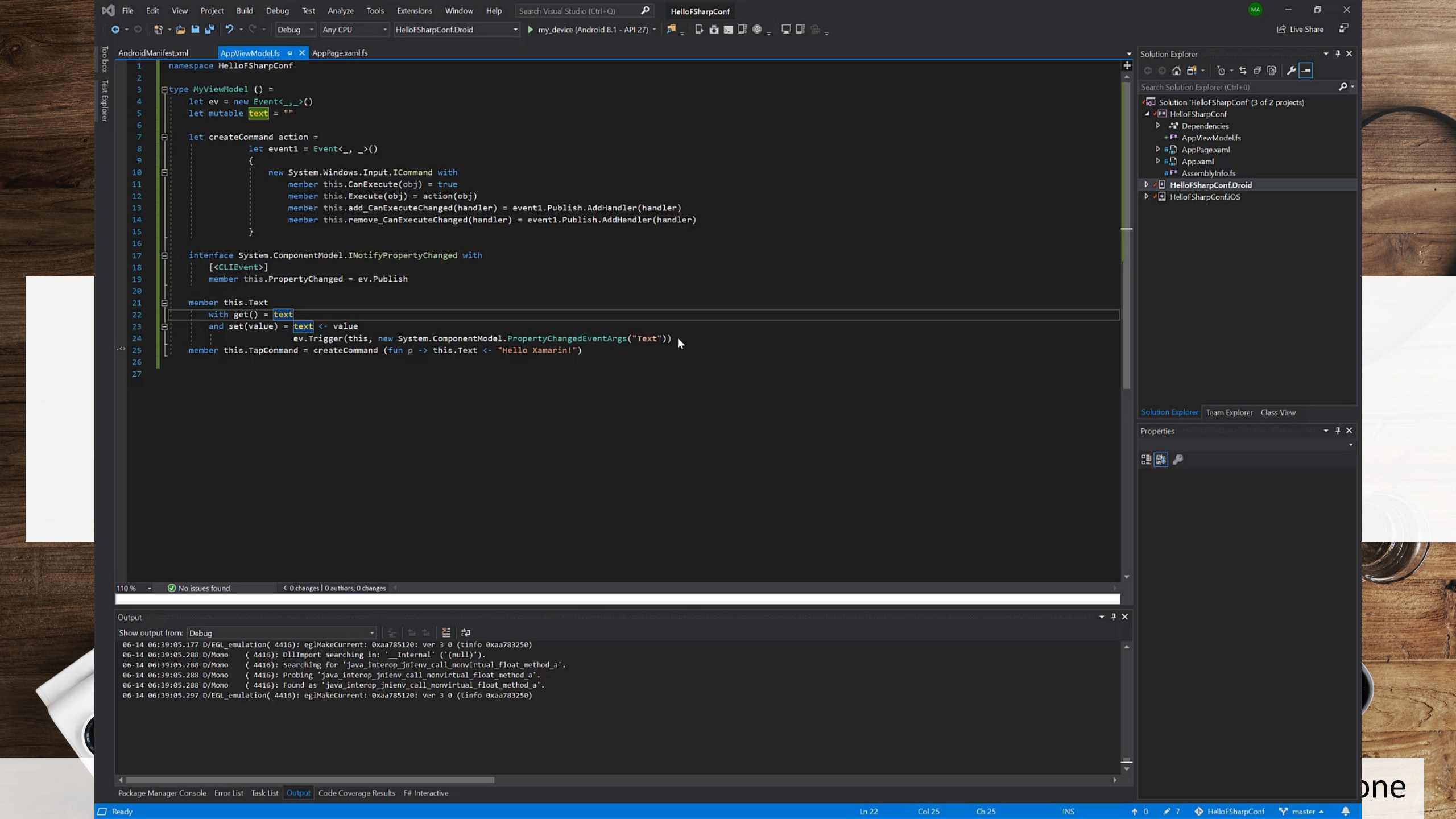

HelloFSharpConf
MA
Live Share
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'HelloFSharpConf' (3 of 2 projects)
HelloFSharpConf
Dependencies
AppViewModel.fs
AppPage.xaml
App.xaml
AssemblyInfo.fs
HelloFSharpConf.Droid
HelloFSharpConf.iOS


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
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = tex
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")


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
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = tex
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")

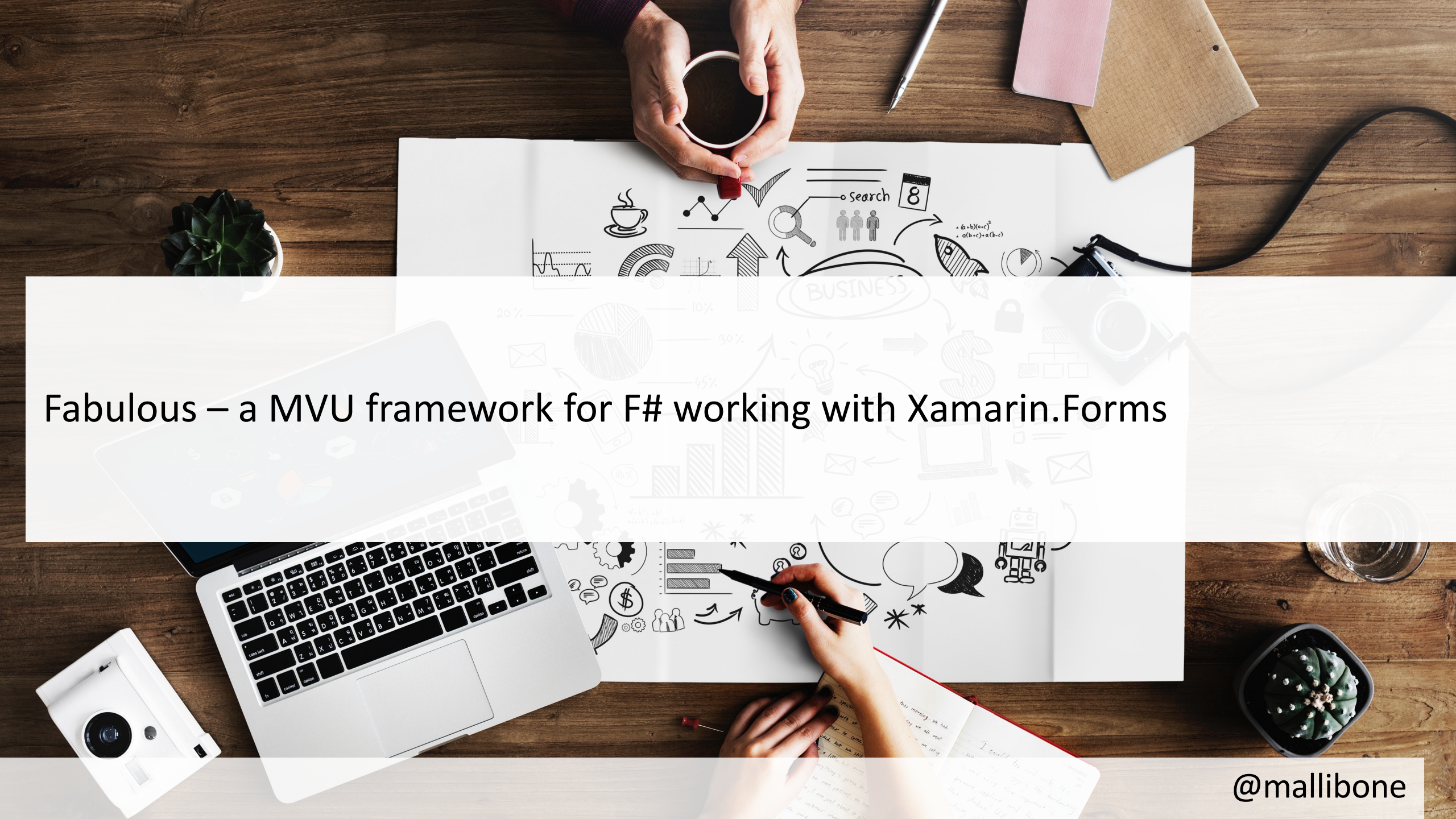

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = tex
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")


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
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = text
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")


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)
Live Share
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = text
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")
110 %
No issues found
0 changes | 0 authors, 0 changes
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
Solution Explorer
Team Explorer
Class View
Properties
Output
Show output from: Debug
06-14 06:39:05.177 D/EGL_emulation(4416): eglMakeCurrent: 0xaa785120: 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): eglMakeCurrent: 0xaa785120: ver 3 0 (tinfo 0xaa783250)
Package Manager Console
Error List
Task List
Output
Code Coverage Results
F# Interactive
Ready
Ln 22
Col 25
Ch 25
INS
HelloFSharpConf
master

F# can do OO programming but it is functional first

@mallibone



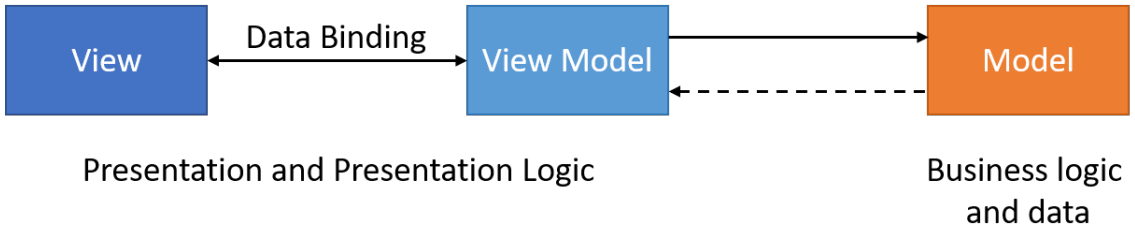
Fabulous – a MVU framework for F# working with Xamarin.Forms

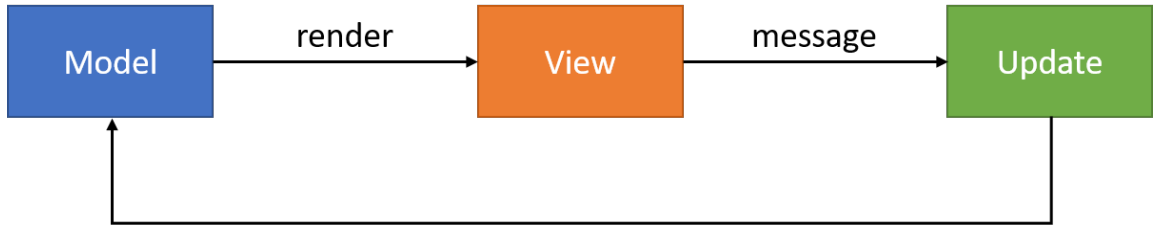
@mallibone

MVU the functional MV* Pattern

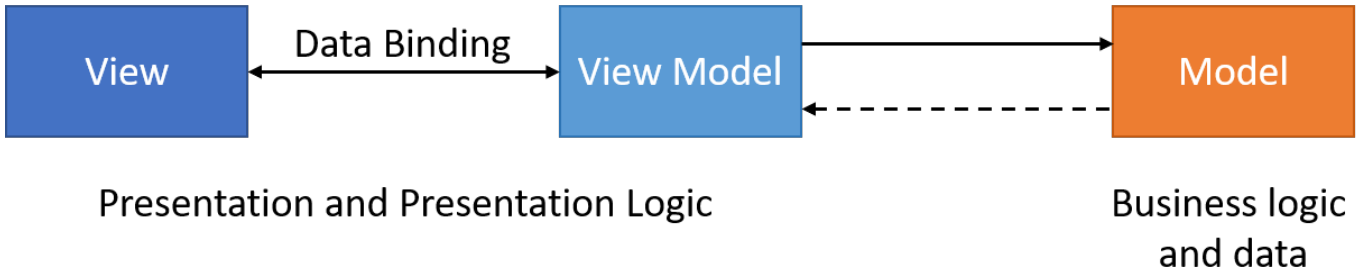
@mallibone

MVVM vs MVU

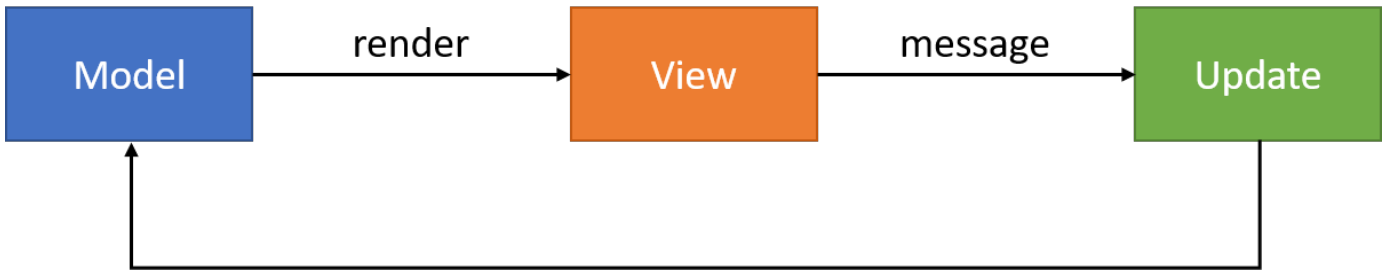

View
Data Binding
View Model
Model
Presentation and Presentation Logic
Business logic
and data


Model
render
View
message
Update

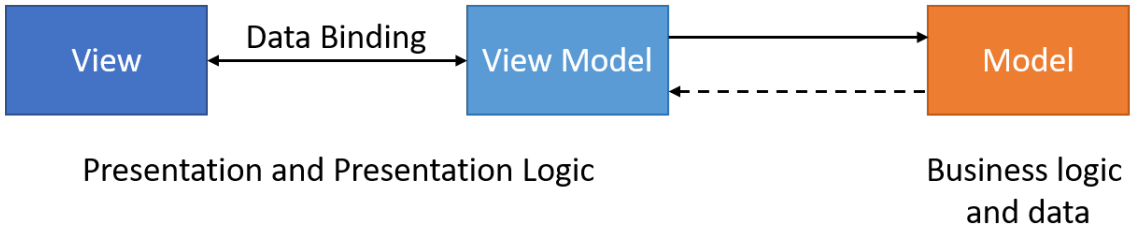
Model View View Model

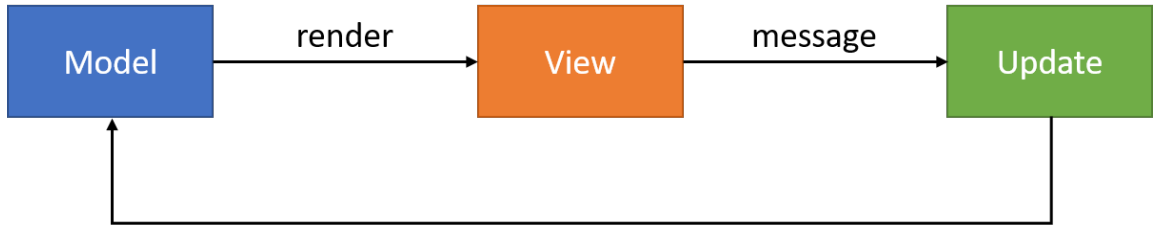


Model View Update



MVVM vs MVU


View
Data Binding
View Model
Model
Presentation and Presentation Logic
Business logic
and data


Model
render
View
message
Update

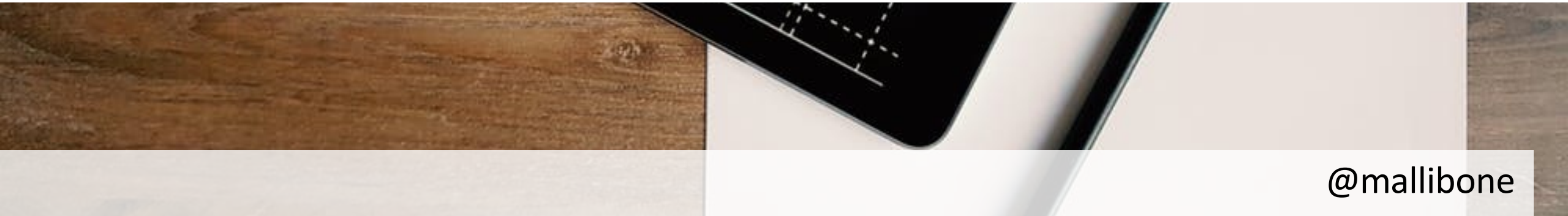
Functional Frontends

Getting started

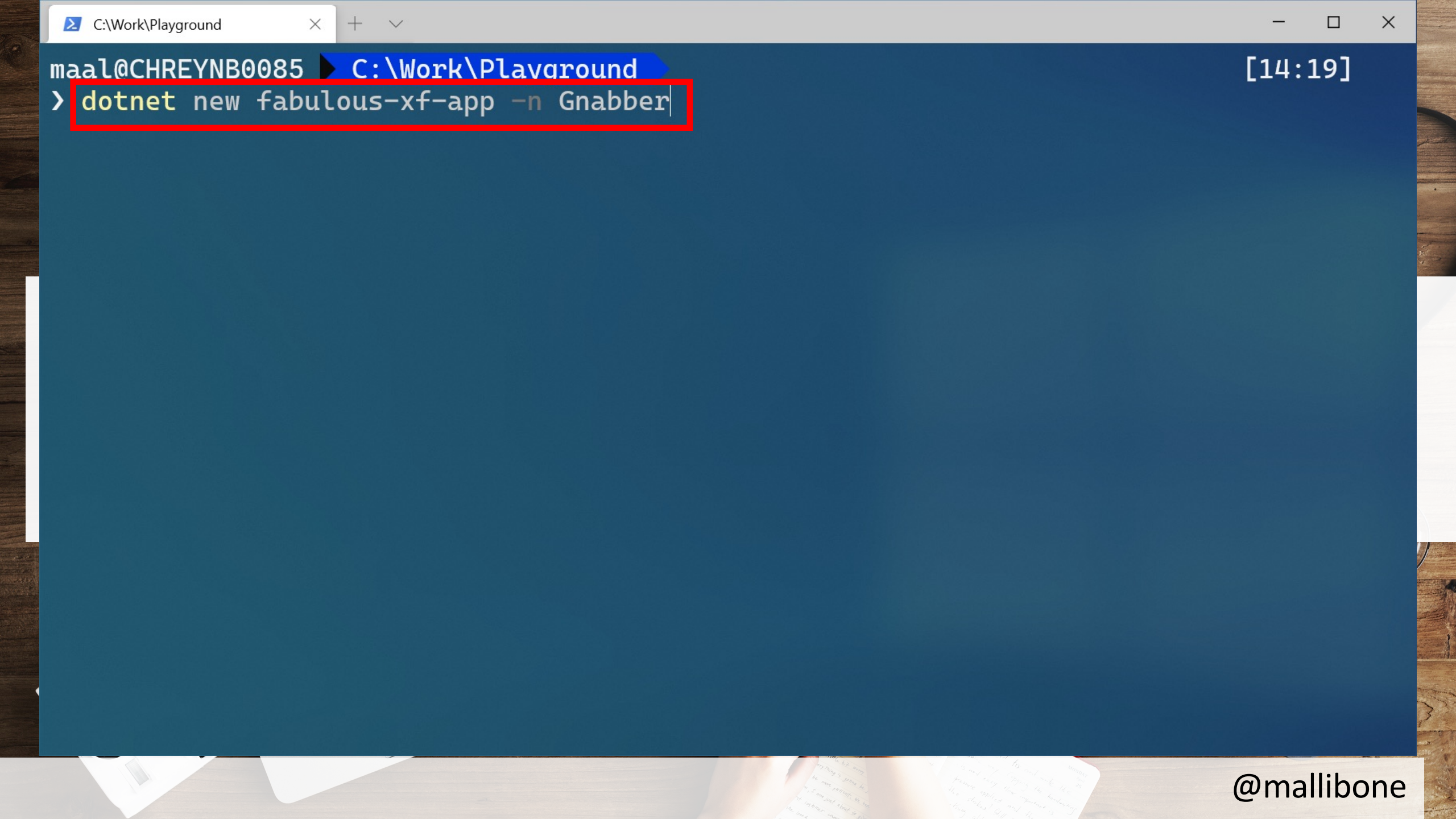
1. Install Visual Studio or Visual Studio for Mac and enable both Xamarin and .NET Core support, these are listed as 'Mobile development with .NET' and '.NET Core Cross-platform development' respectively.

2. Open a command prompt window and install the template pack by entering:

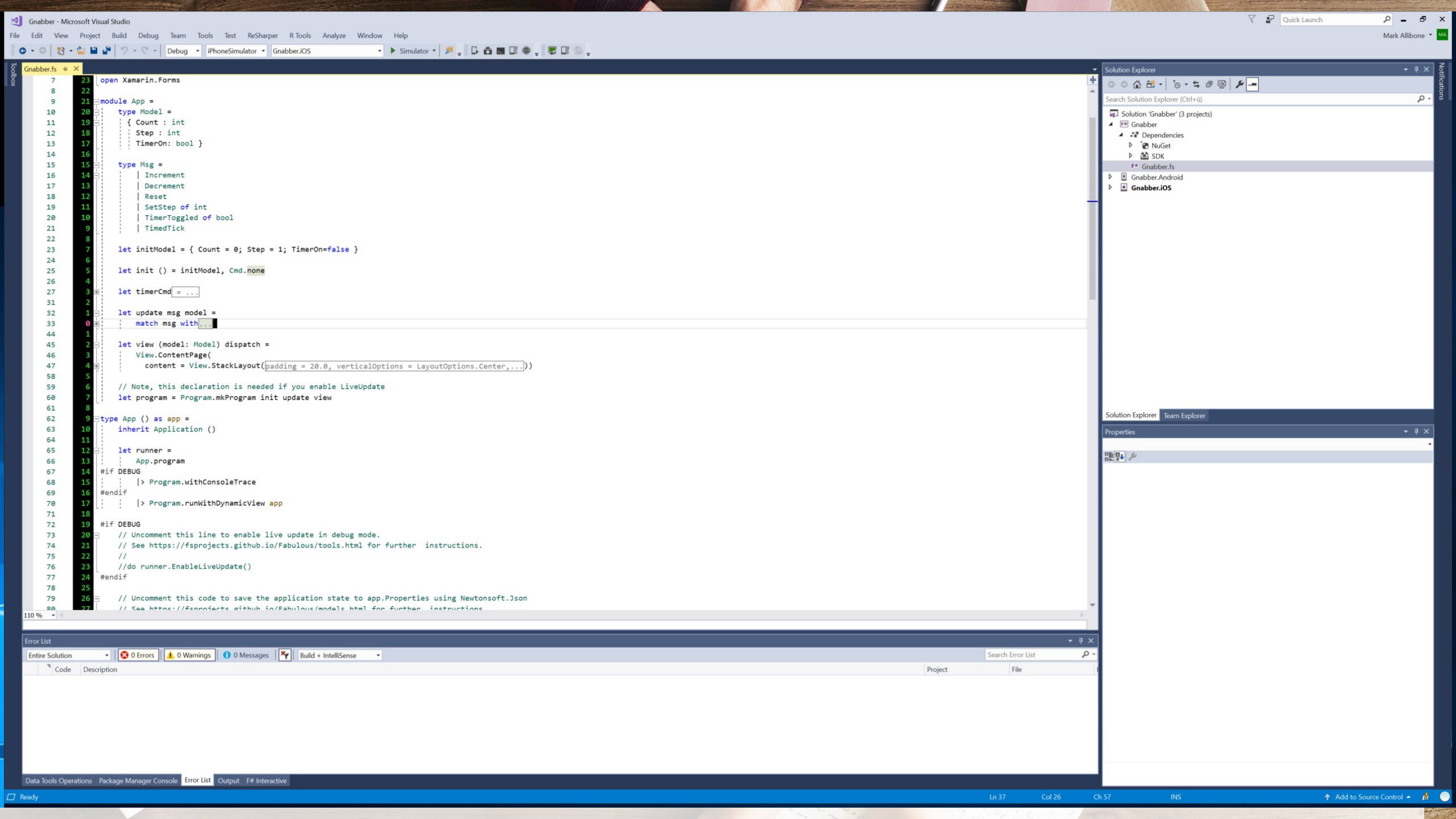
```
dotnet new -i Fabulous.XamarinForms.Templates
```

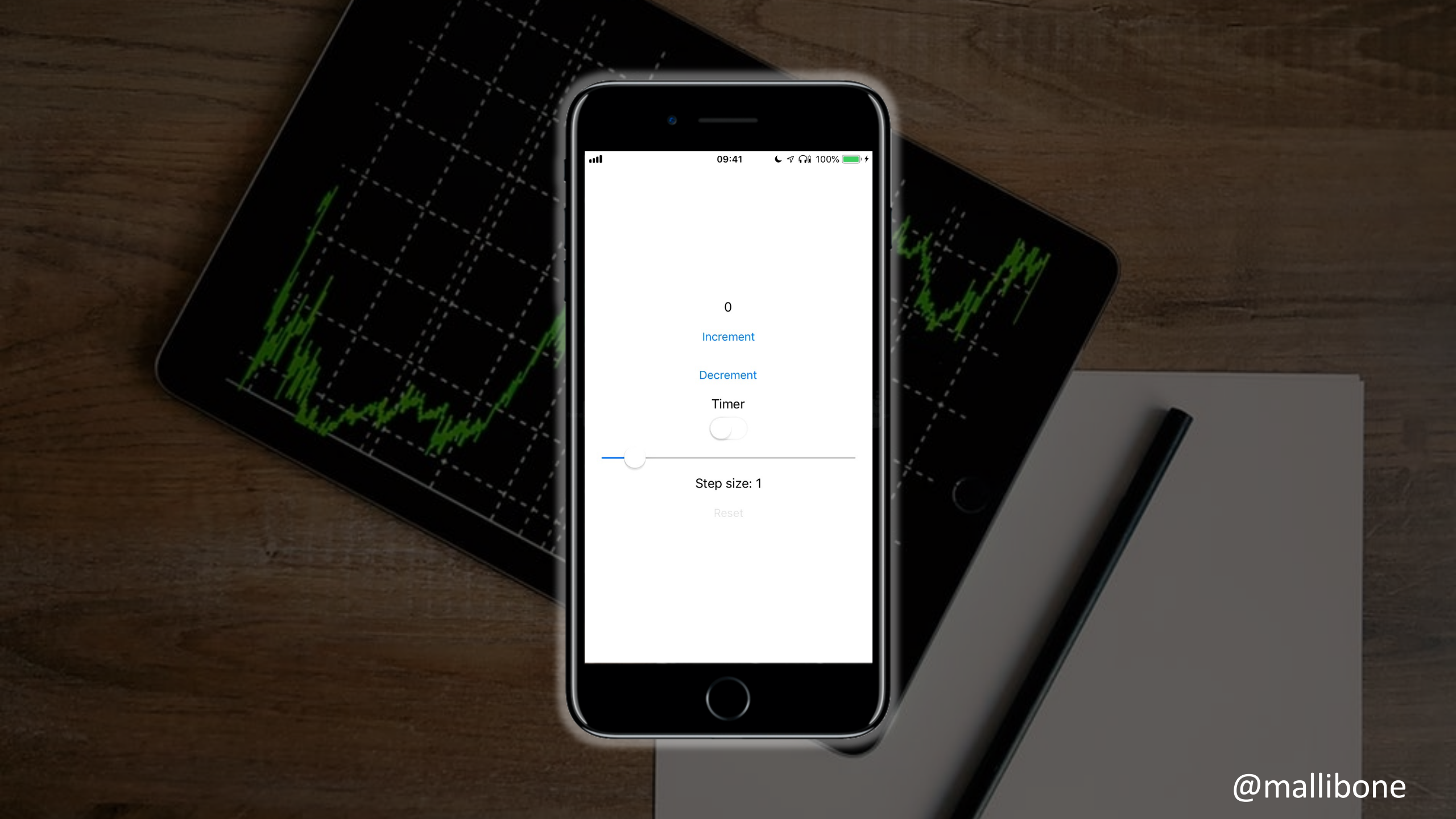


@mallibone

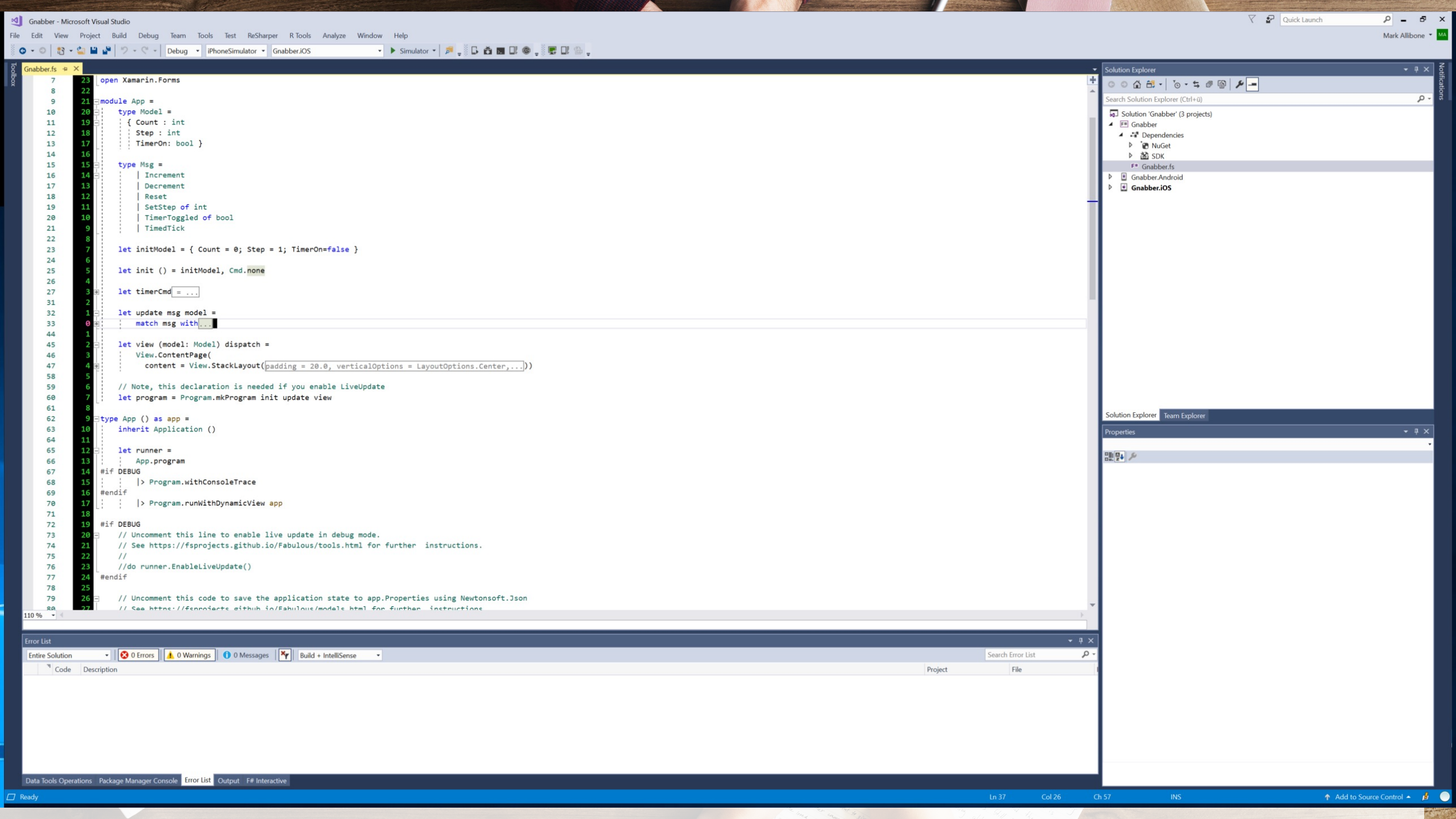

C:\Work\Playground
maal@CHREYNB0085 C:\Work\Playground
[14:19]
dotnet new fabulous-xf-app -n Gnabber

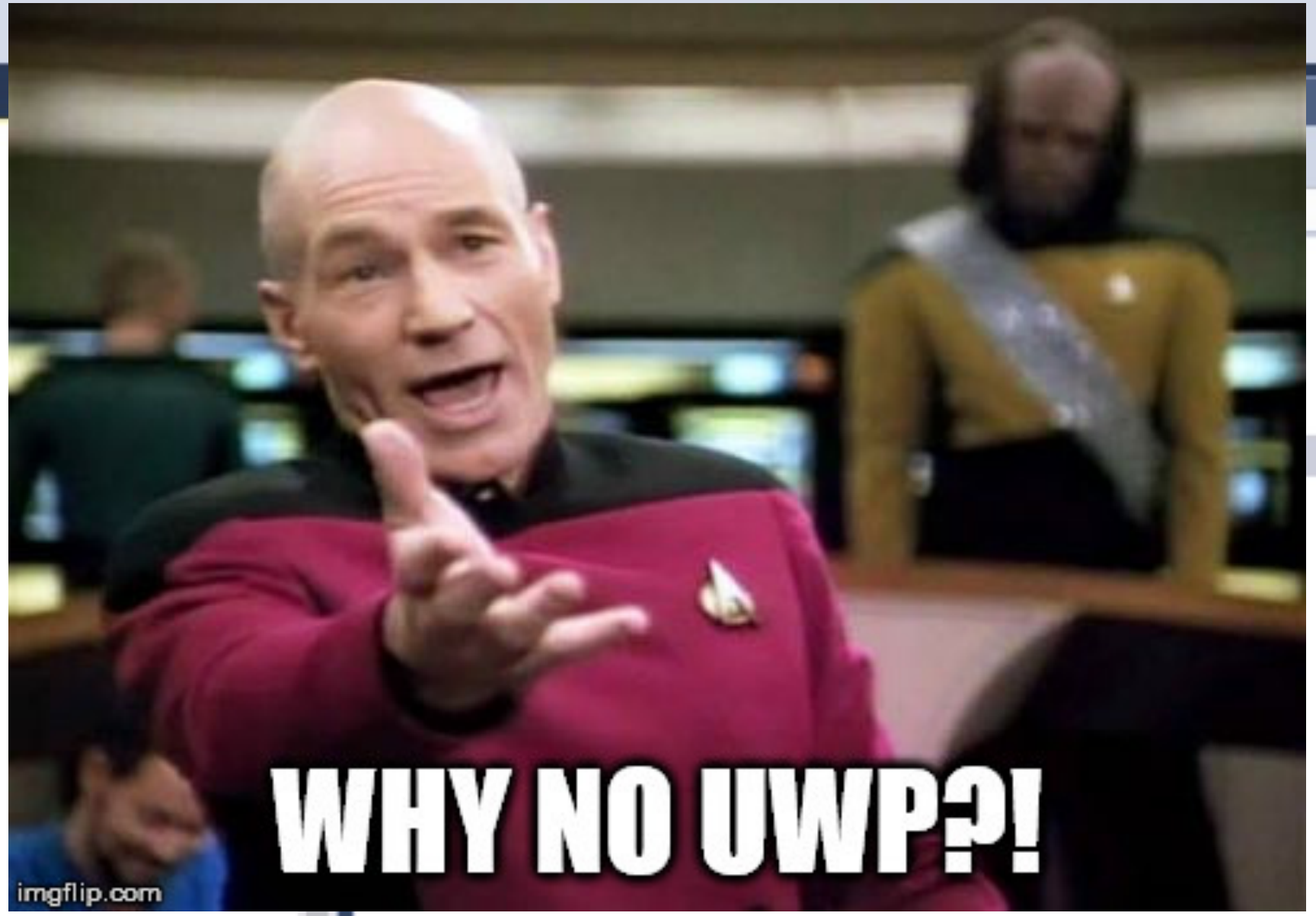
@mallibone

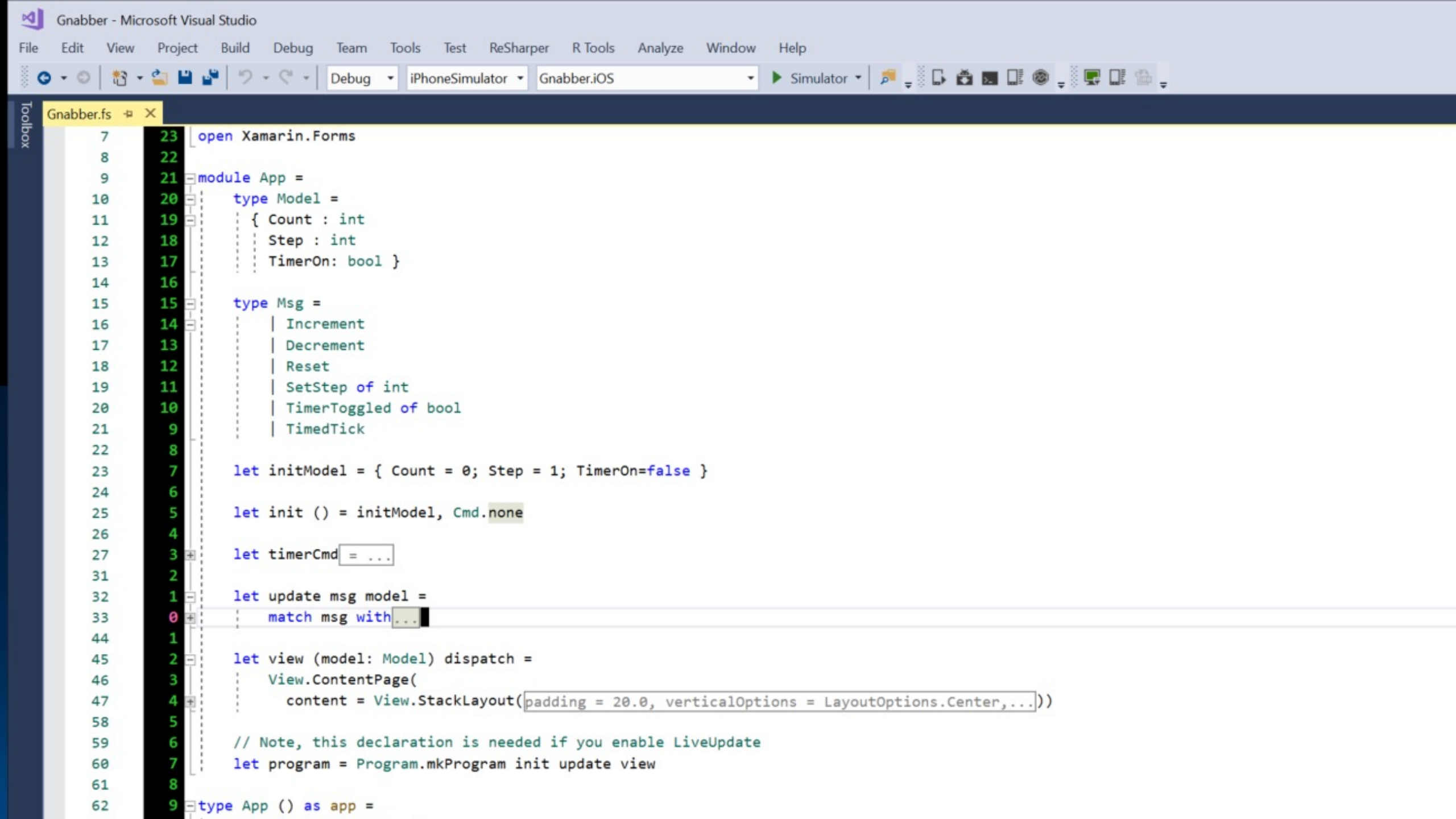

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| 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
// Uncomment this line to enable live update in debug mode.
// See https://fsprojects.github.io/Fabulous/tools.html for further instructions.
//
//do runner.EnableLiveUpdate()
#endif
// Uncomment this code to save the application state to app.Properties using Newtonsoft.Json
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'Gnabber' (3 projects)
Gnabber
Dependencies
NuGet
SDK
Gnabber.fs
Gnabber.Android
Gnabber.iOS
Solution Explorer
Team Explorer
Properties
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Code
Description
Project
File
Data Tools Operations
Package Manager Console
Error List
Output
F# Interactive
Ready
Ln 37
Col 26
Ch 57
INS
Add to Source Control

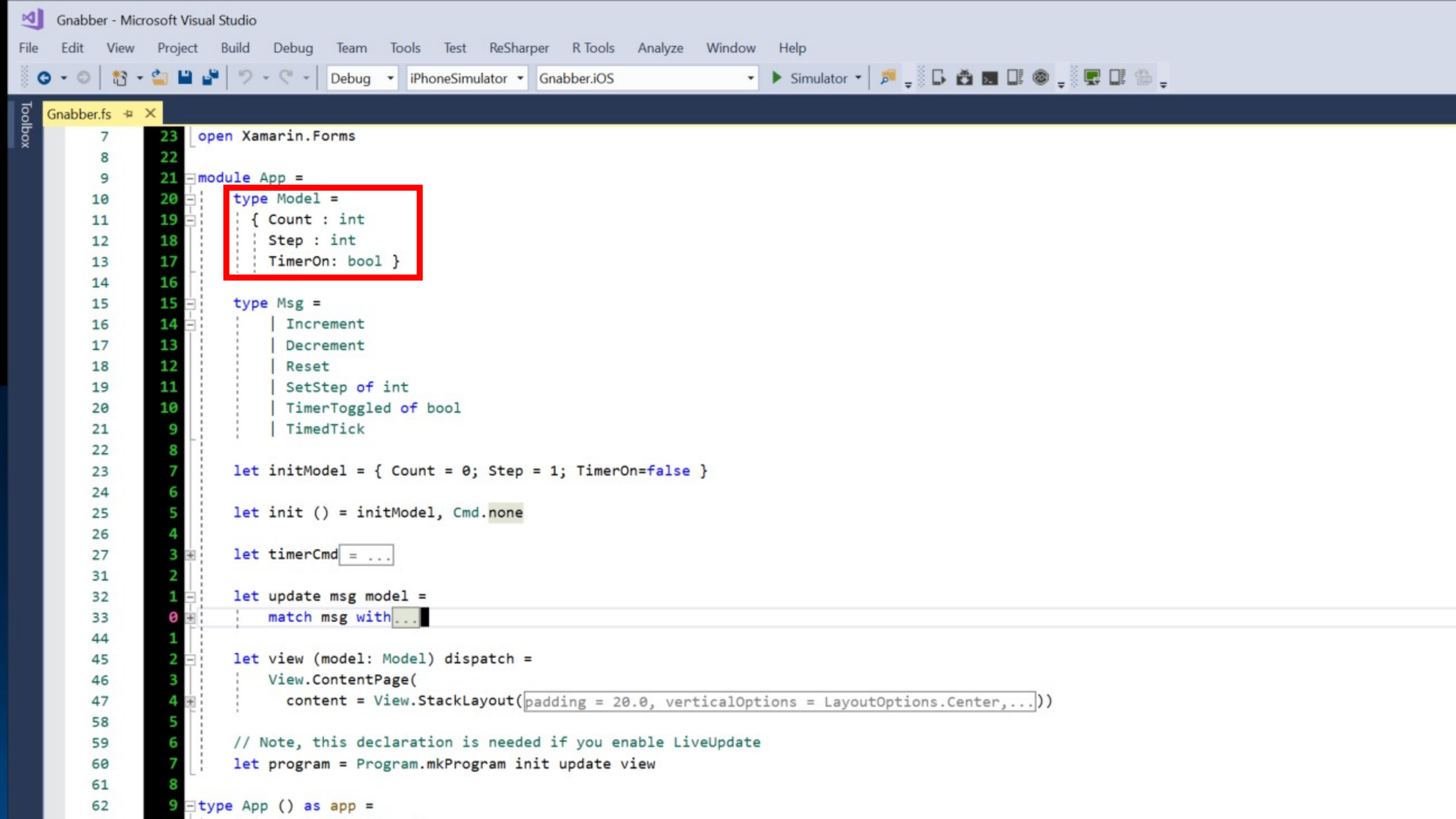


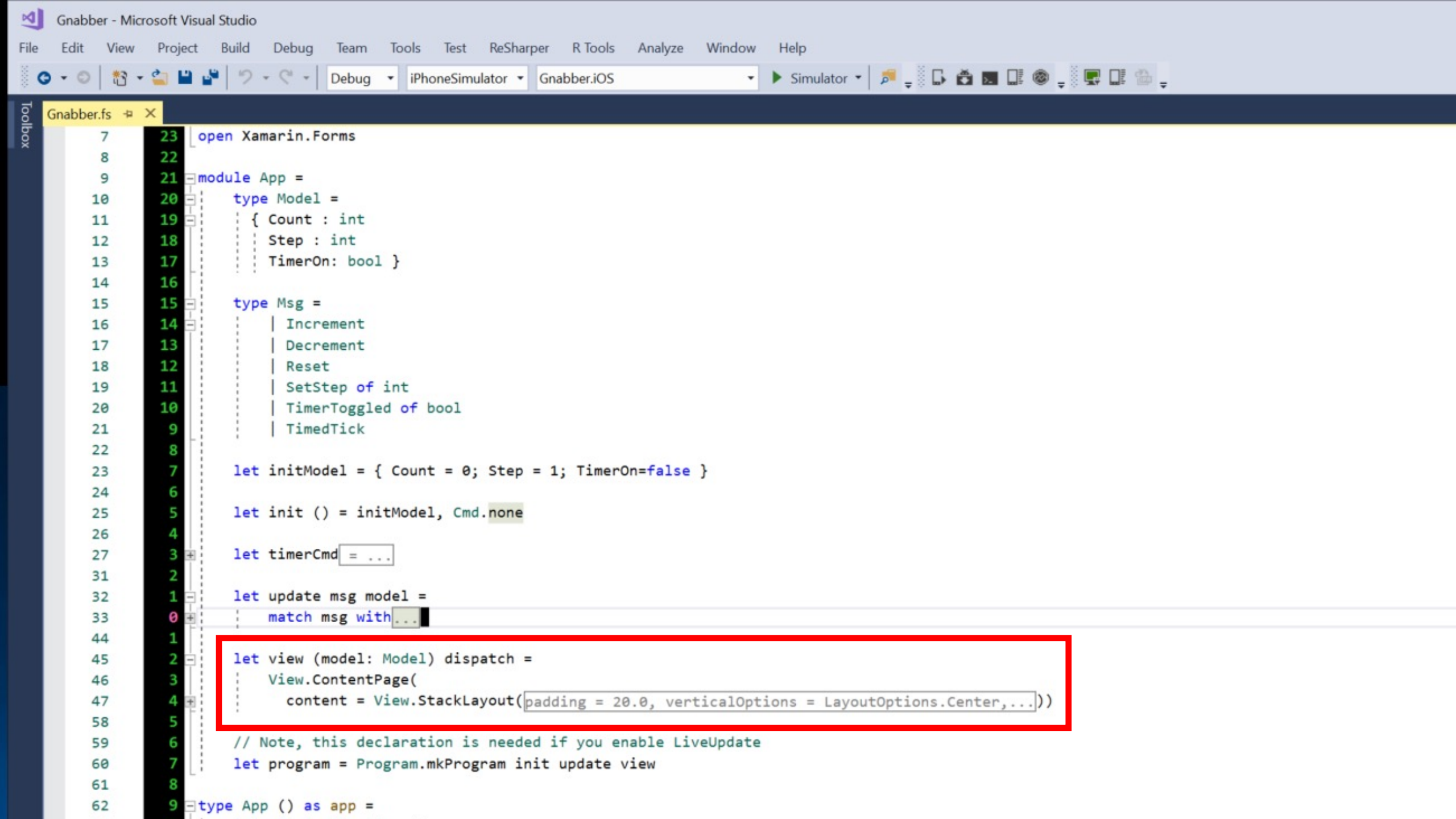
@mallibone

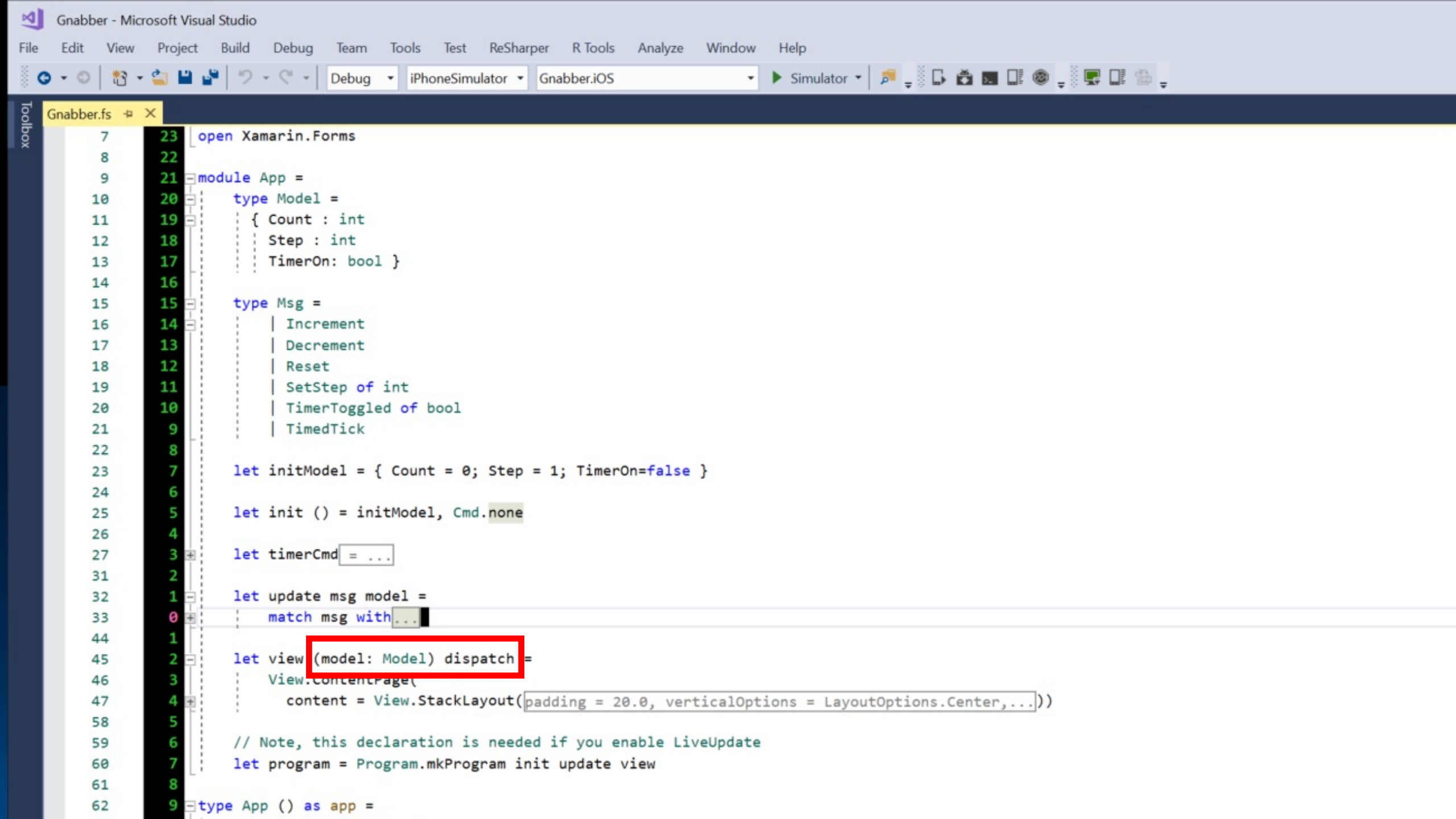

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| 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
// Uncomment this line to enable live update in debug mode.
// See https://fsprojects.github.io/Fabulous/tools.html for further instructions.
//
//do runner.EnableLiveUpdate()
#endif
// Uncomment this code to save the application state to app.Properties using Newtonsoft.Json
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'Gnabber' (3 projects)
Gnabber
Dependencies
NuGet
SDK
Gnabber.fs
Gnabber.Android
Gnabber.iOS
Solution Explorer
Team Explorer
Properties
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Code
Description
Project
File
Data Tools Operations
Package Manager Console
Error List
Output
F# Interactive
Ready
Ln 37
Col 26
Ch 57
INS
Add to Source Control

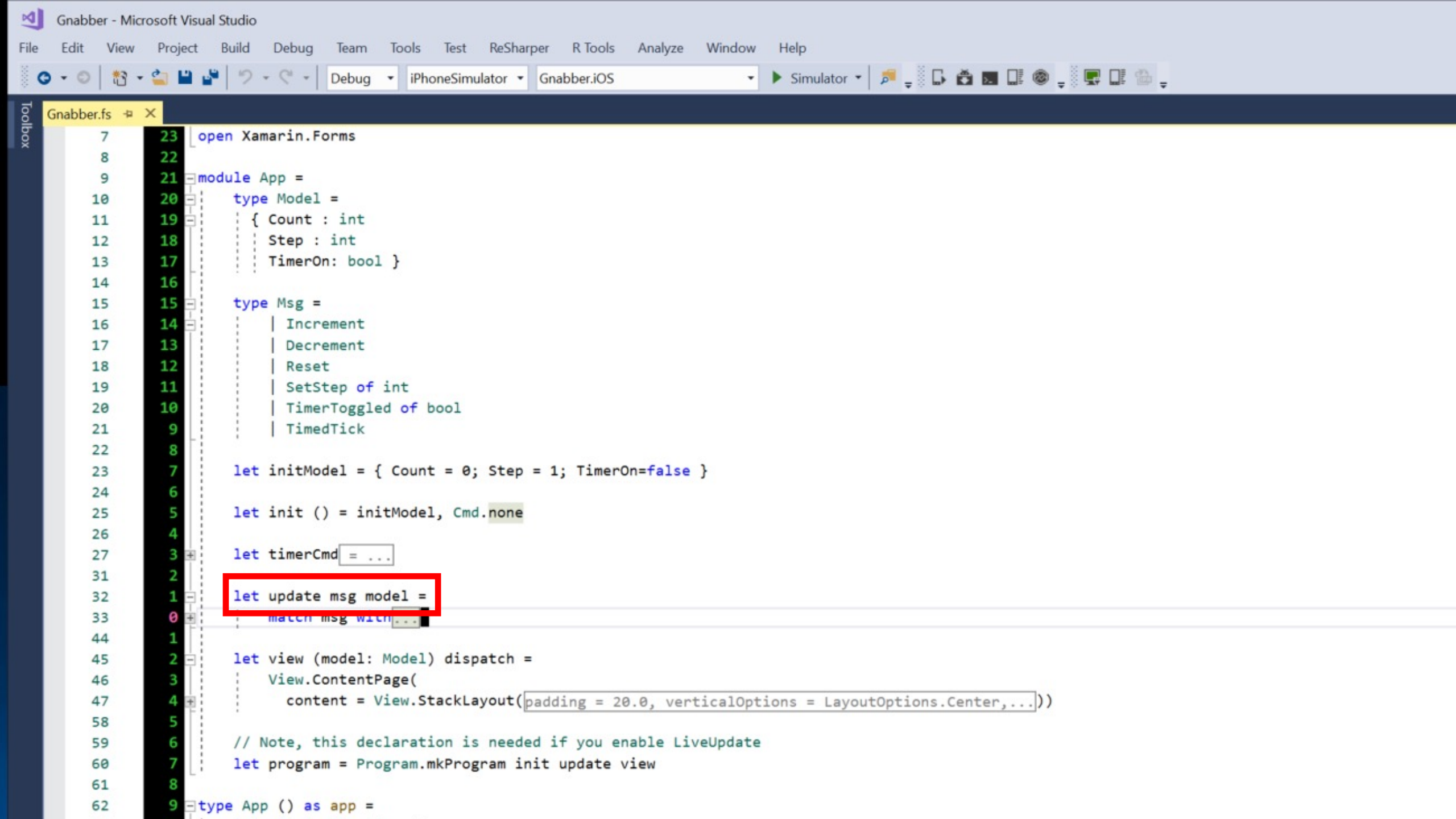

Quick Launch
Mark Allibone
MA
Notifications
WHY NO UWP?!
imgflip.com

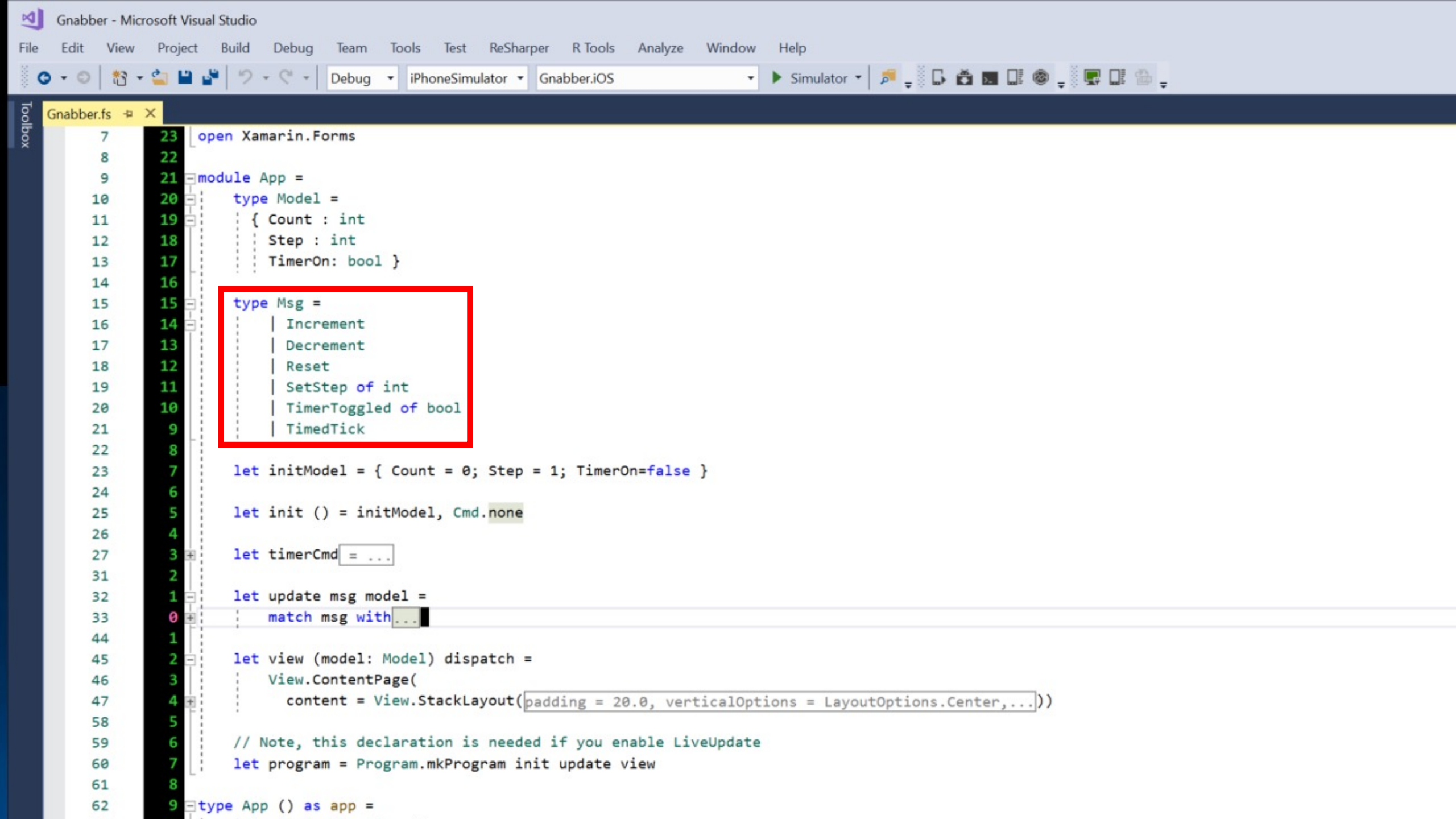

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| 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 =


Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
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 =

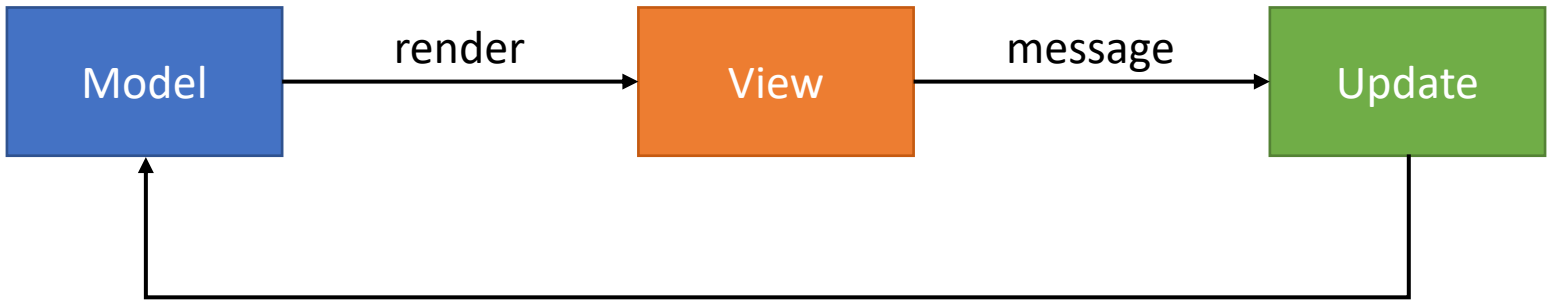

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| 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 =

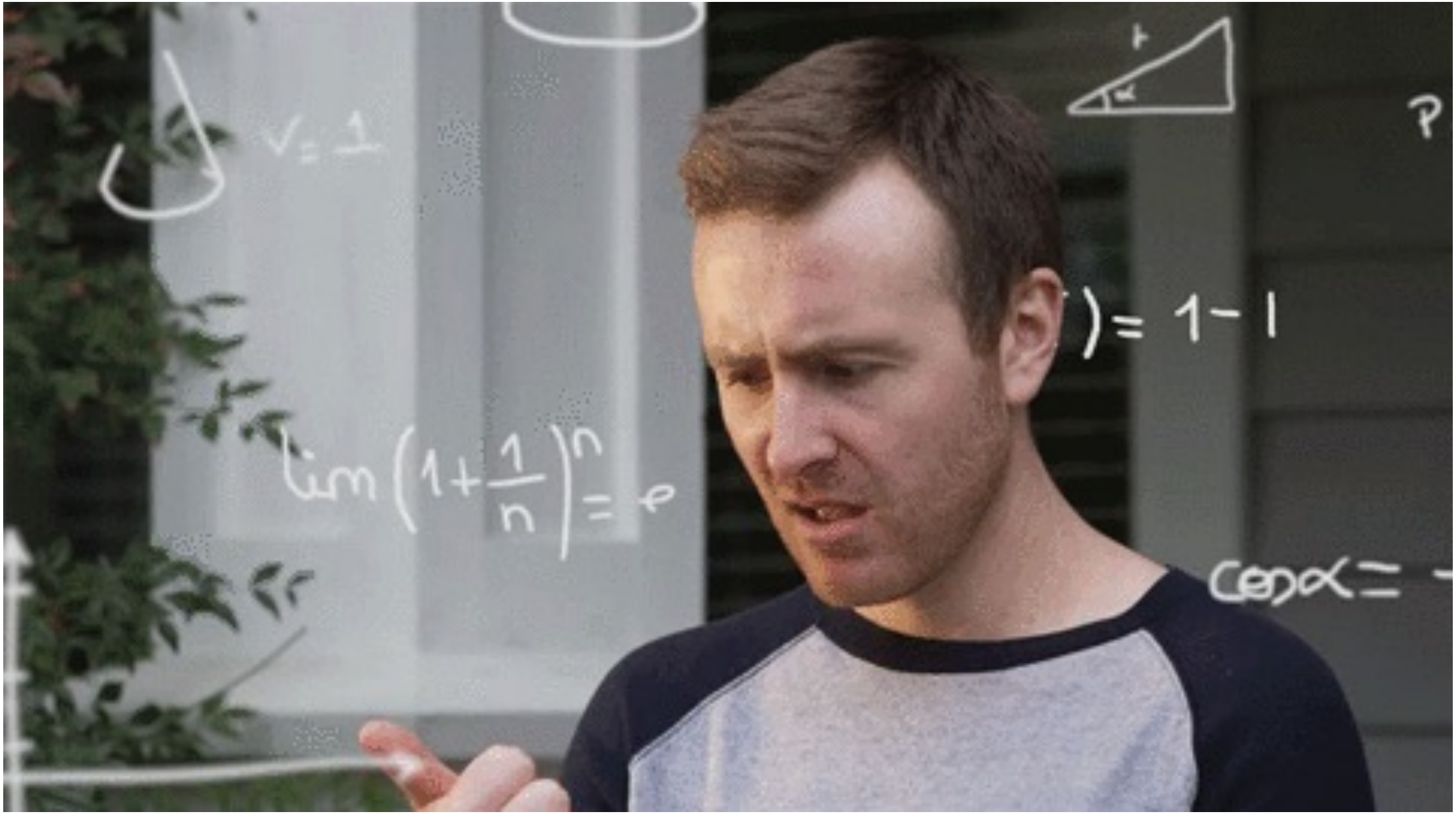

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| 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 =


Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
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 =


Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
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 =

Model View Update




$\lim\left(1+\frac{1}{n}\right)^n = e$

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

@mallibone

Doing it with style

@mallibone


XAML vs Coded UI


Demo
@mallibone

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

ATOMS ▾ MOLECULES ▾ ORGANISMS ▾ TEMPLATES ▾ PAGES ▾ ALL SIZE 1251PX / 78.19EM S M L FULL RAND

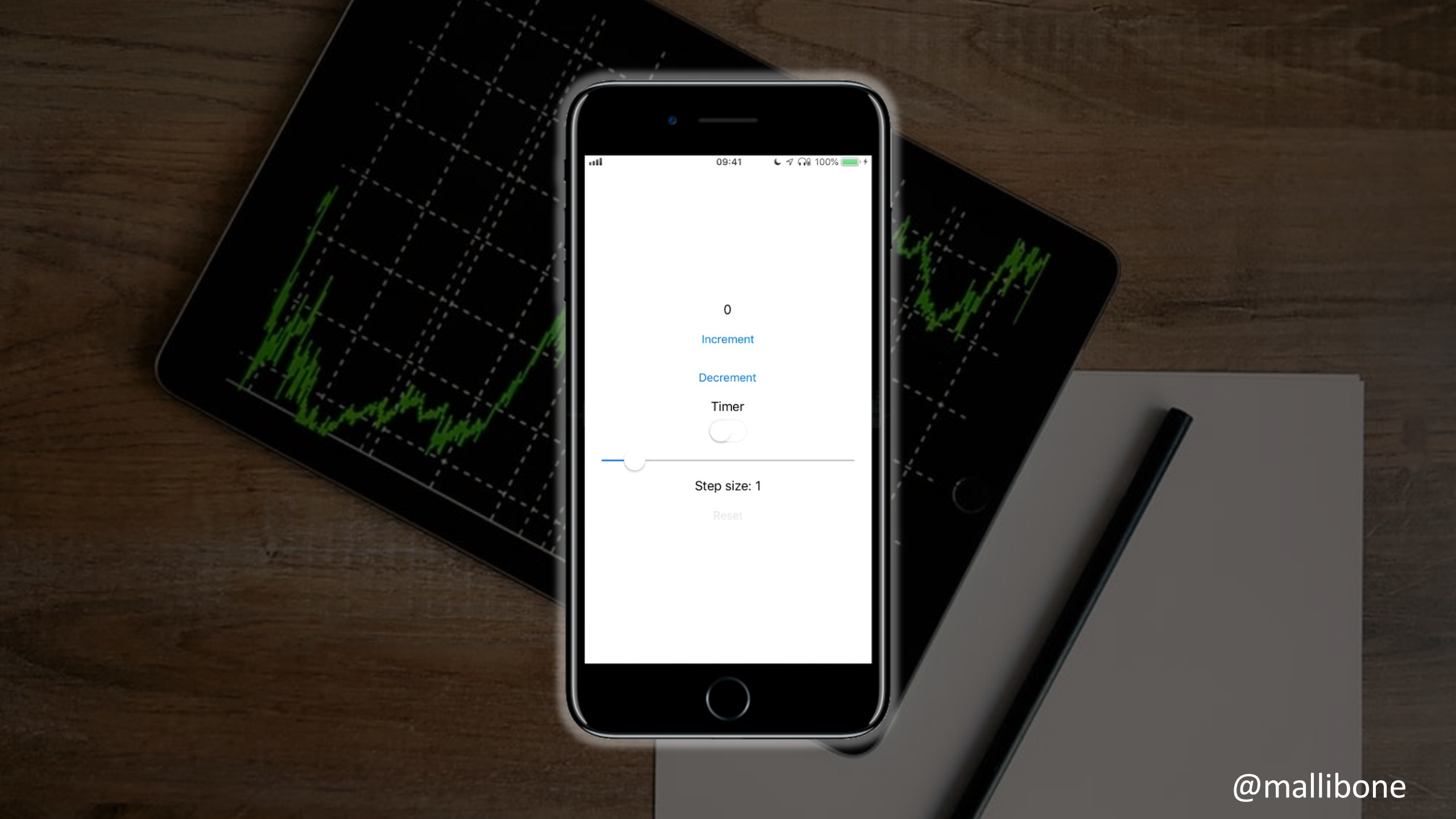
https://patternlab.io/

@mallibone

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

@mallibone

Photo by **Josh Hild** from **Pexels**



@mallibone

```fsharp
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
```

@mallibone

```fsharp
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
```

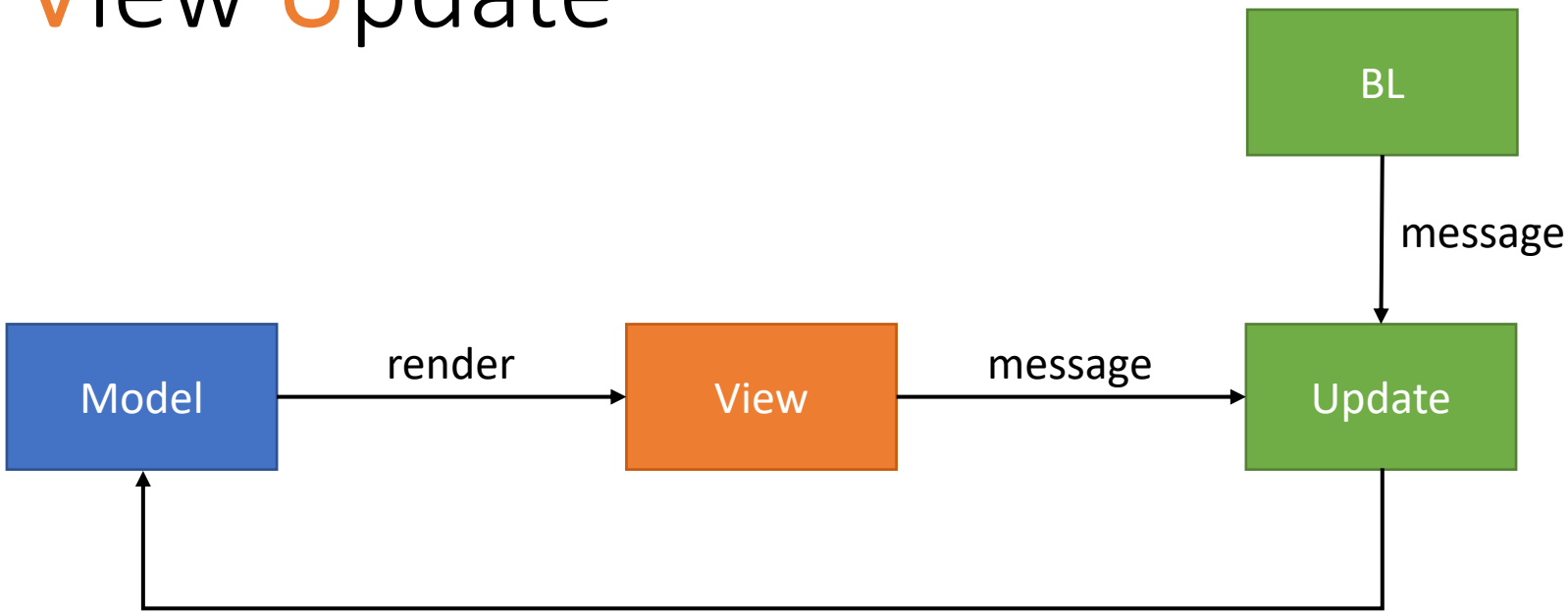
@mallibone

```fsharp
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
```

@mallibone

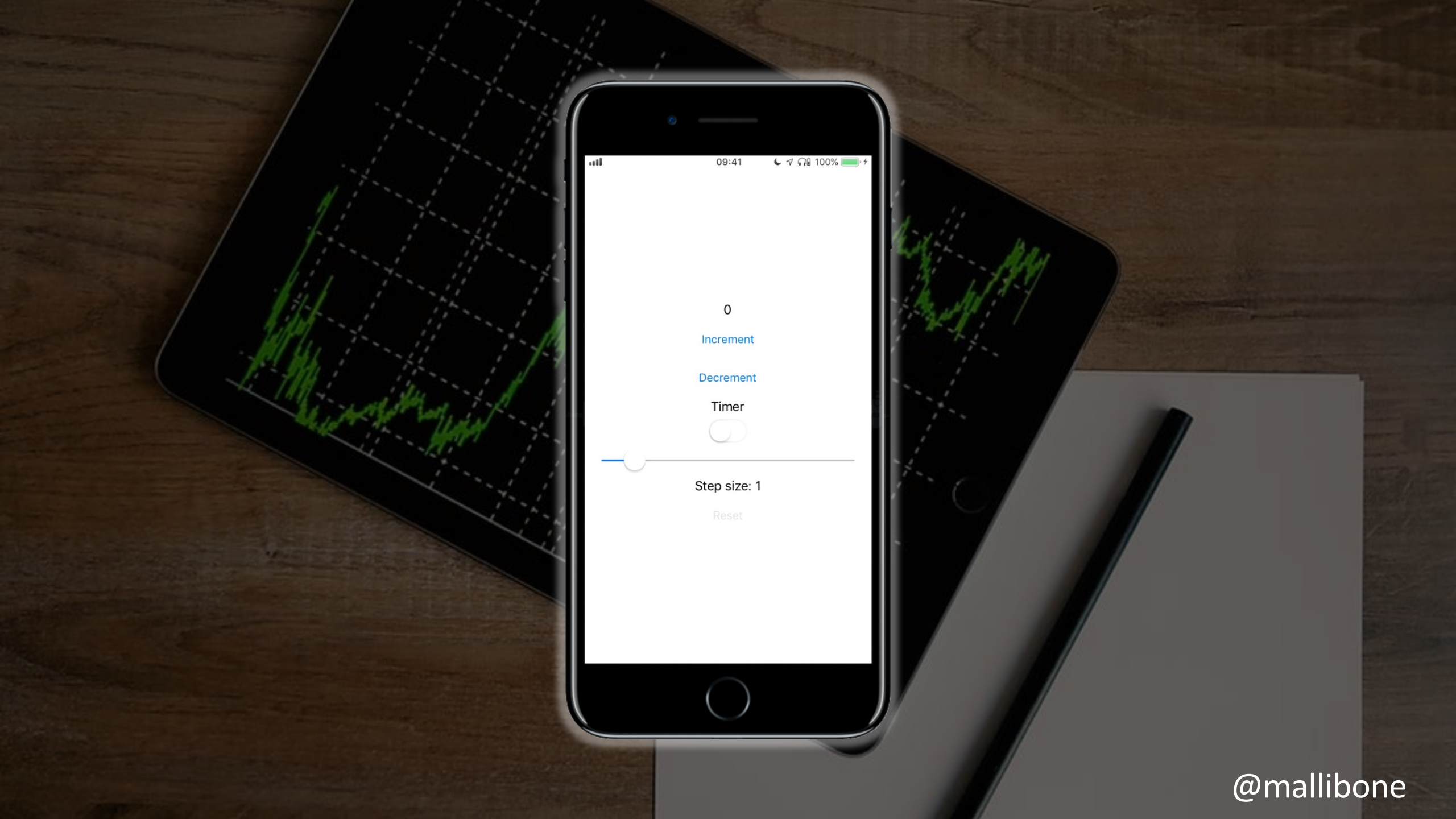
Model View Update



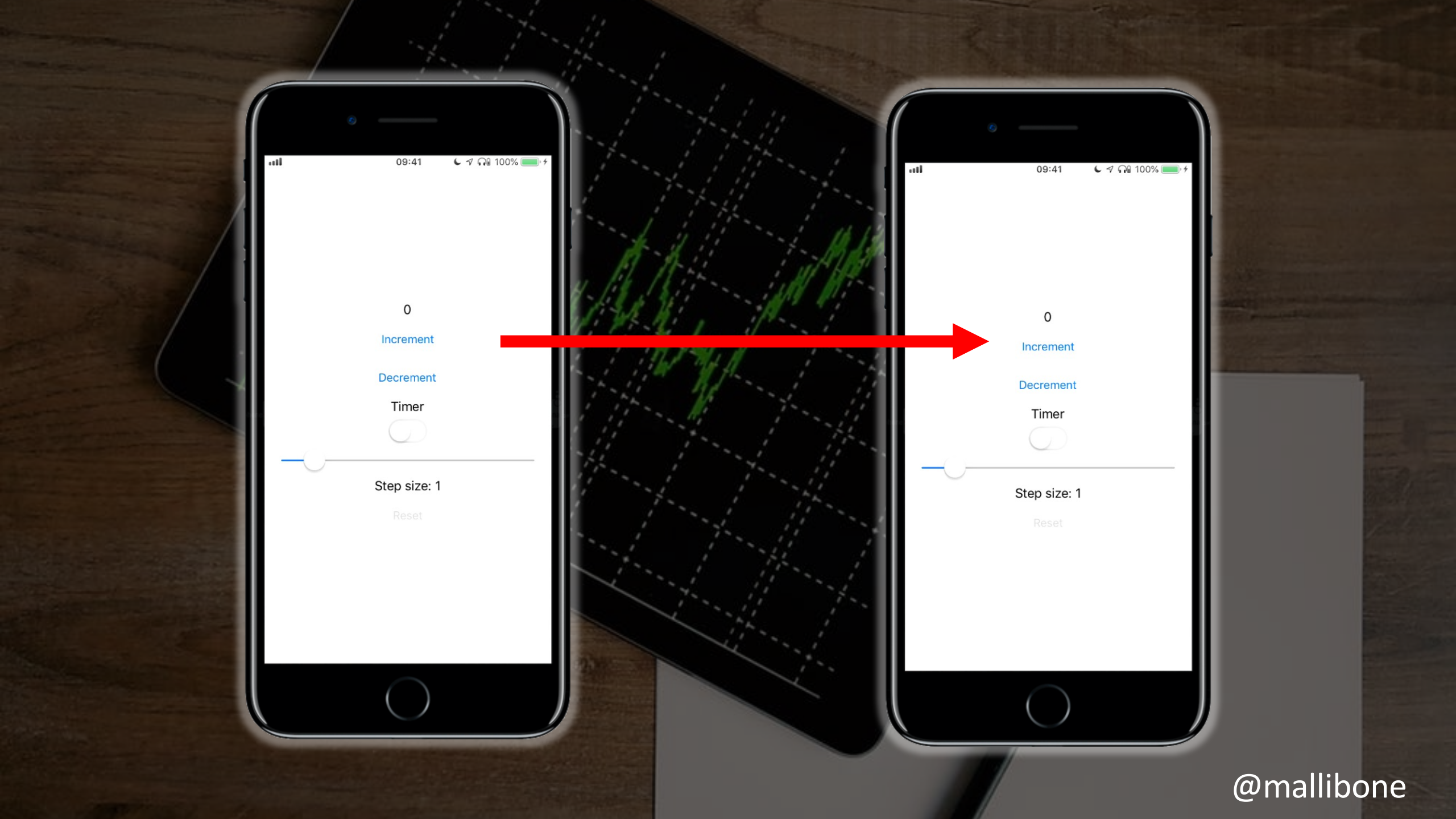

Demo
@mallibone

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

Photo by **Torsten Dettlaff** from **Pexels**

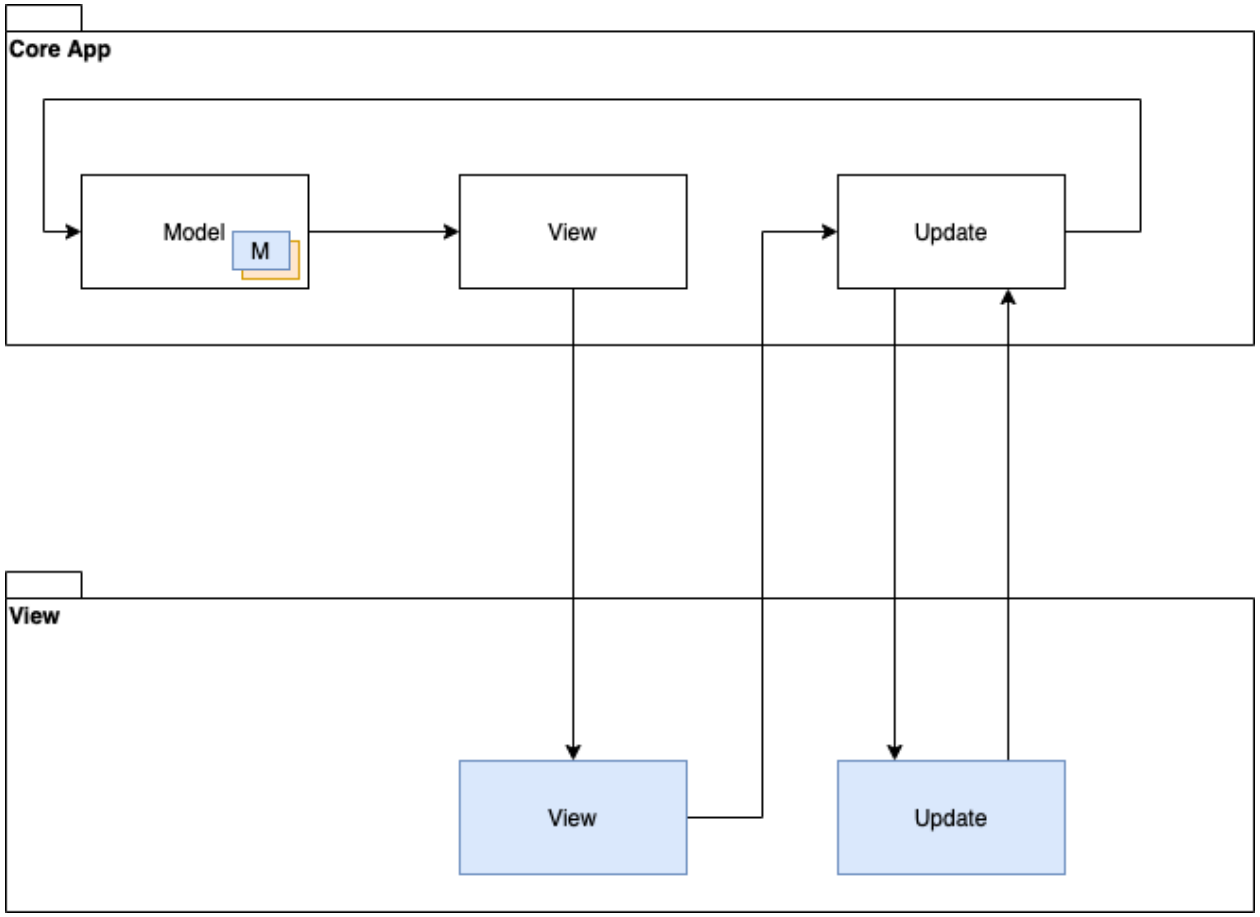


@mallibone


09:41
100%
0
Increment
Decrement
Timer
Step size: 1
Reset
09:41
100%
0
Increment
Decrement
Timer
Step size: 1
Reset

@mallibone

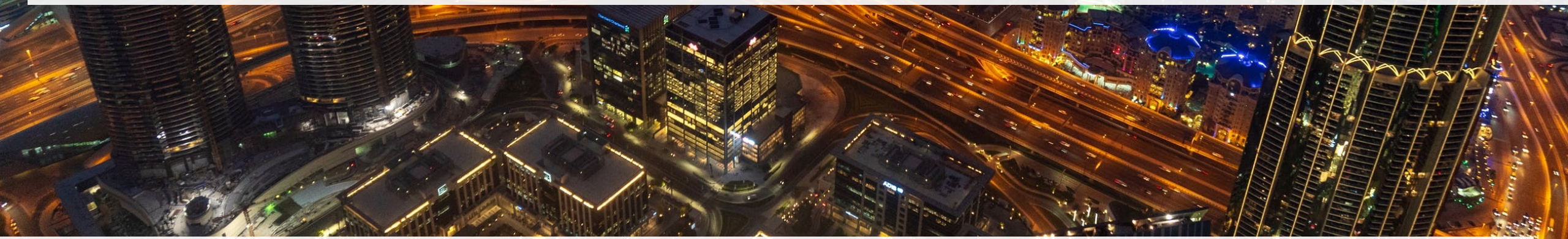
MVU with multiple pages



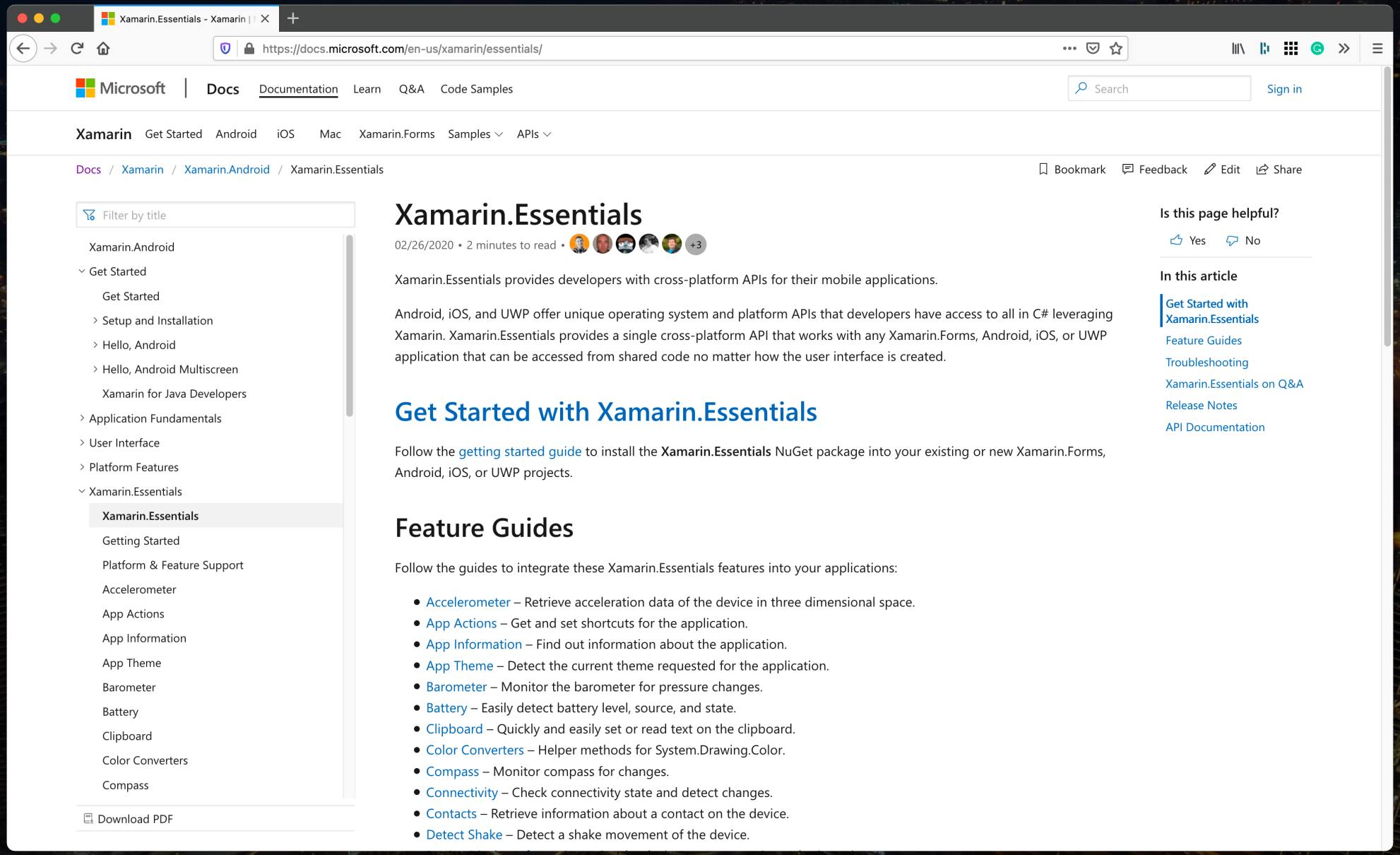

Demo
@mallibone

- 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

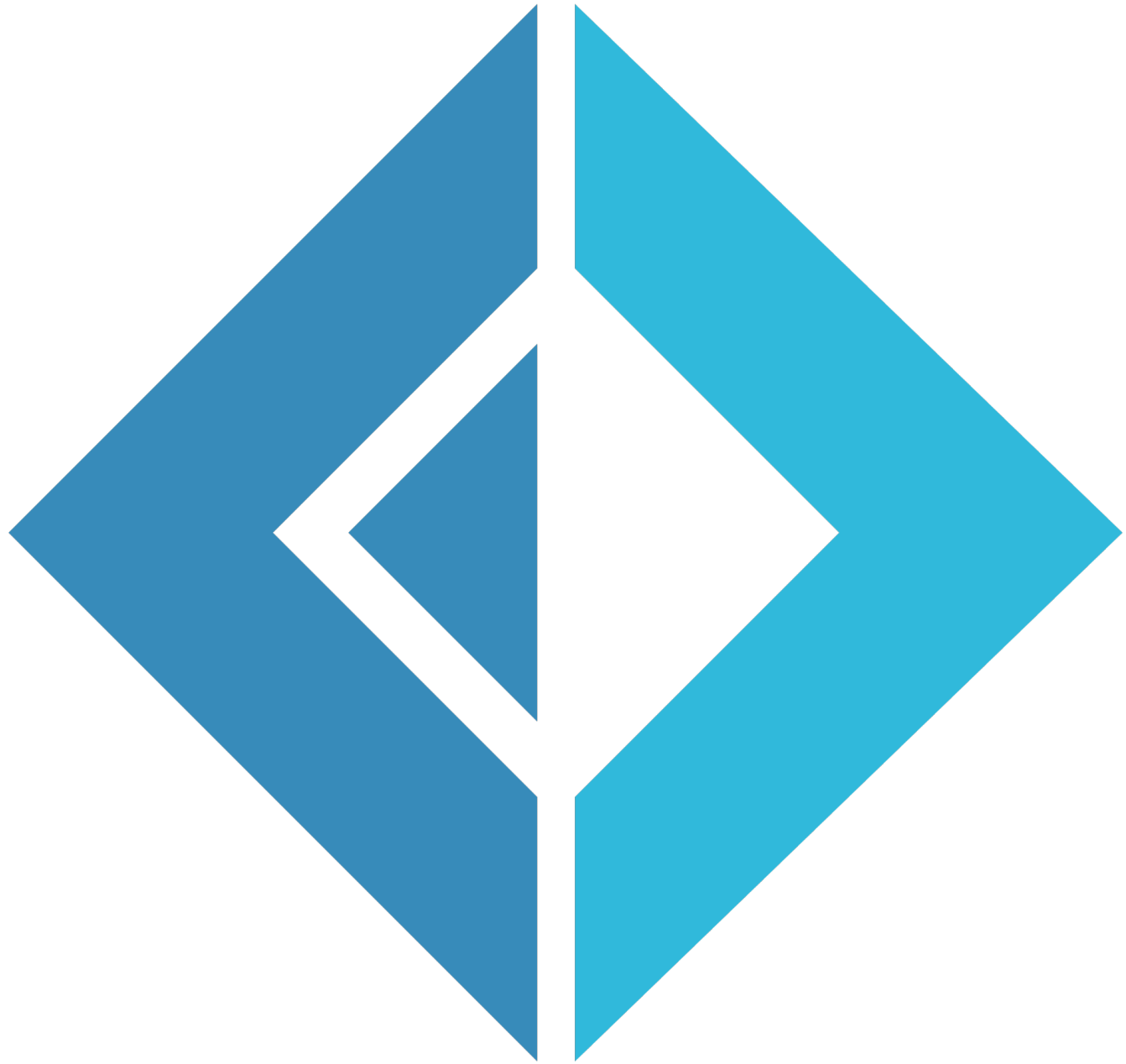
What about Shell? Visual Material? Or your other favorite Xamarin feature?



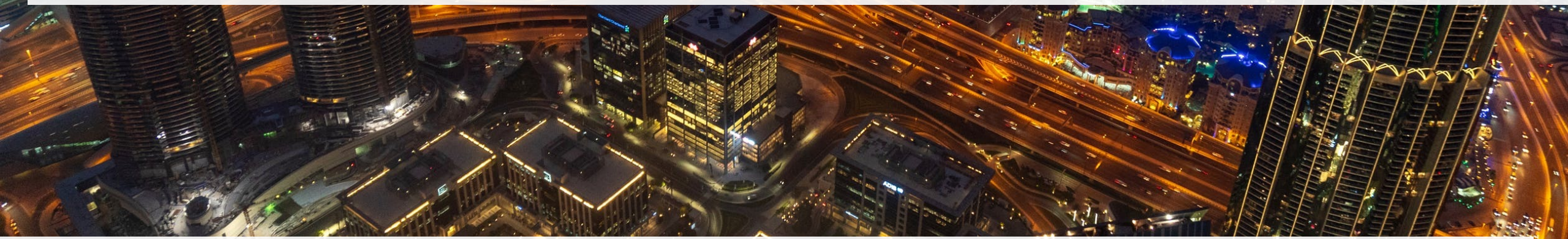

NuGet
@mallibone


Xamarin.Essentials - Xamarin
https://docs.microsoft.com/en-us/xamarin/essentials/
Microsoft | Docs Documentation Learn Q&A Code Samples
Search
Sign in
Xamarin Get Started Android iOS Mac Xamarin.Forms Samples APIs
Docs / Xamarin / Xamarin.Android / Xamarin.Essentials
Bookmark Feedback Edit Share
Filter by title
Xamarin.Android
Get Started
Get Started
Setup and Installation
Hello, Android
Hello, Android Multiscreen
Xamarin for Java Developers
Application Fundamentals
User Interface
Platform Features
Xamarin.Essentials
Xamarin.Essentials
Getting Started
Platform & Feature Support
Accelerometer
App Actions
App Information
App Theme
Barometer
Battery
Clipboard
Color Converters
Compass
Download PDF
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

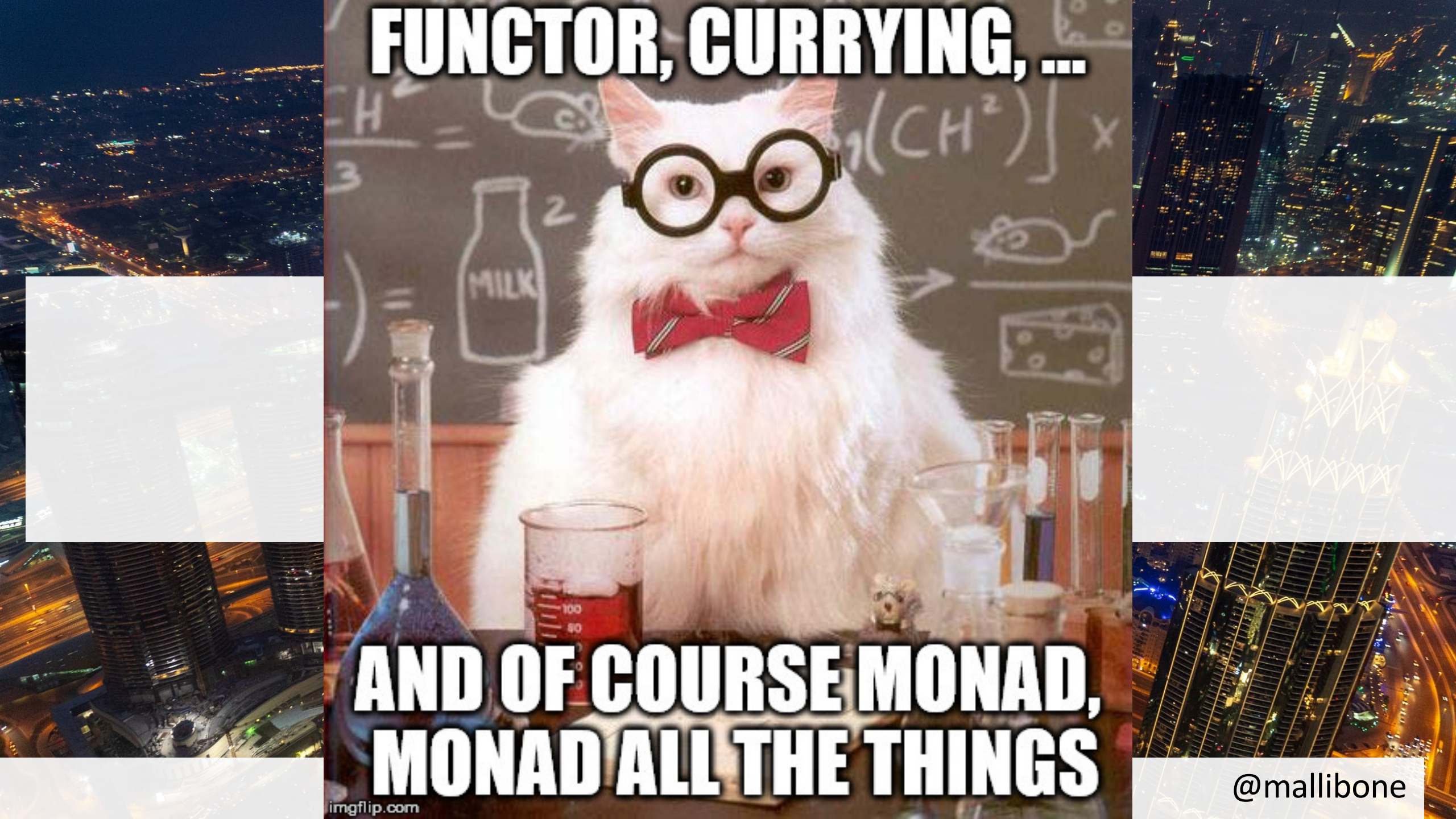
@mallibone



How to get started with F#?



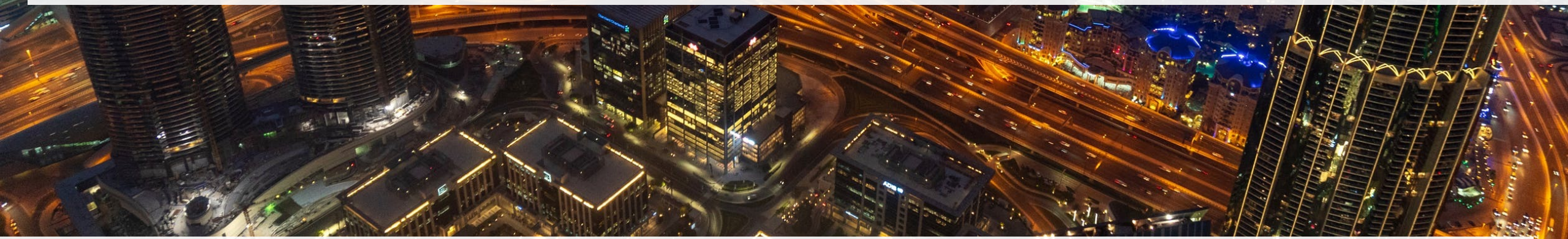
@mallibone

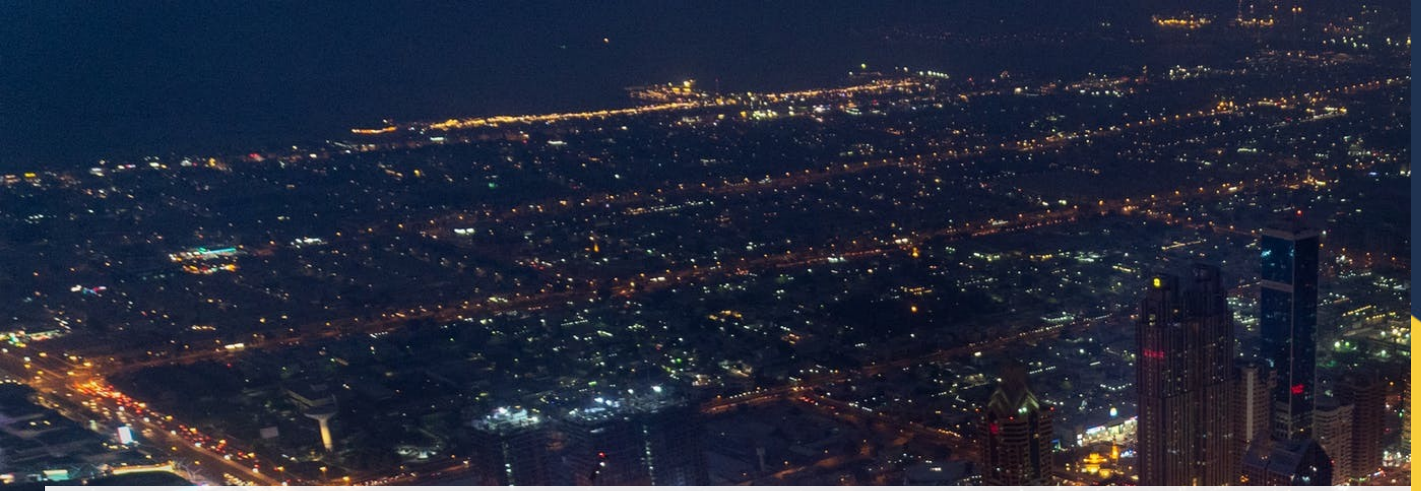

FUNCTOR, CURRYING, ...
AND OF COURSE MONAD,
MONAD ALL THE THINGS
imgflip.com

@mallibone

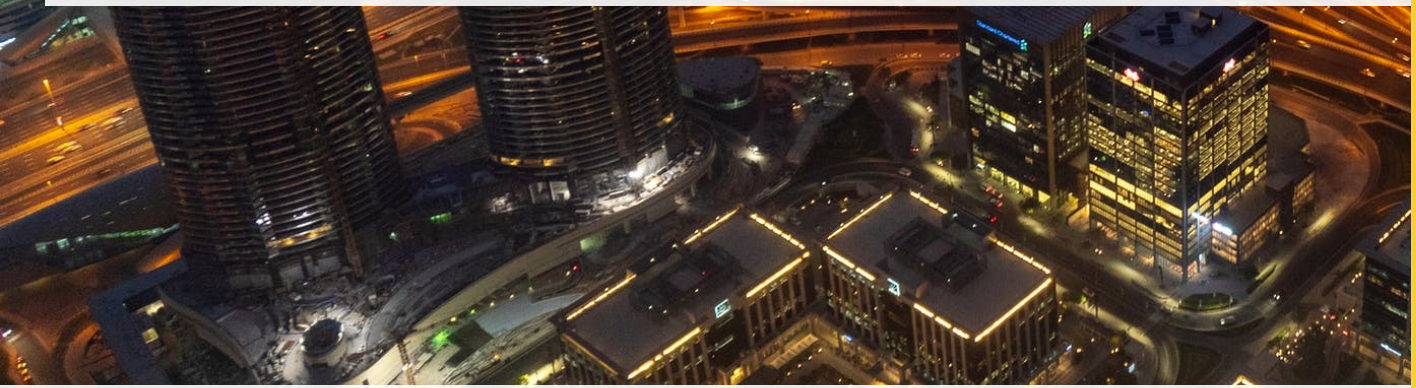

"HOLY FORKING SHIRT!"
#THEGOODPLACE
NBC

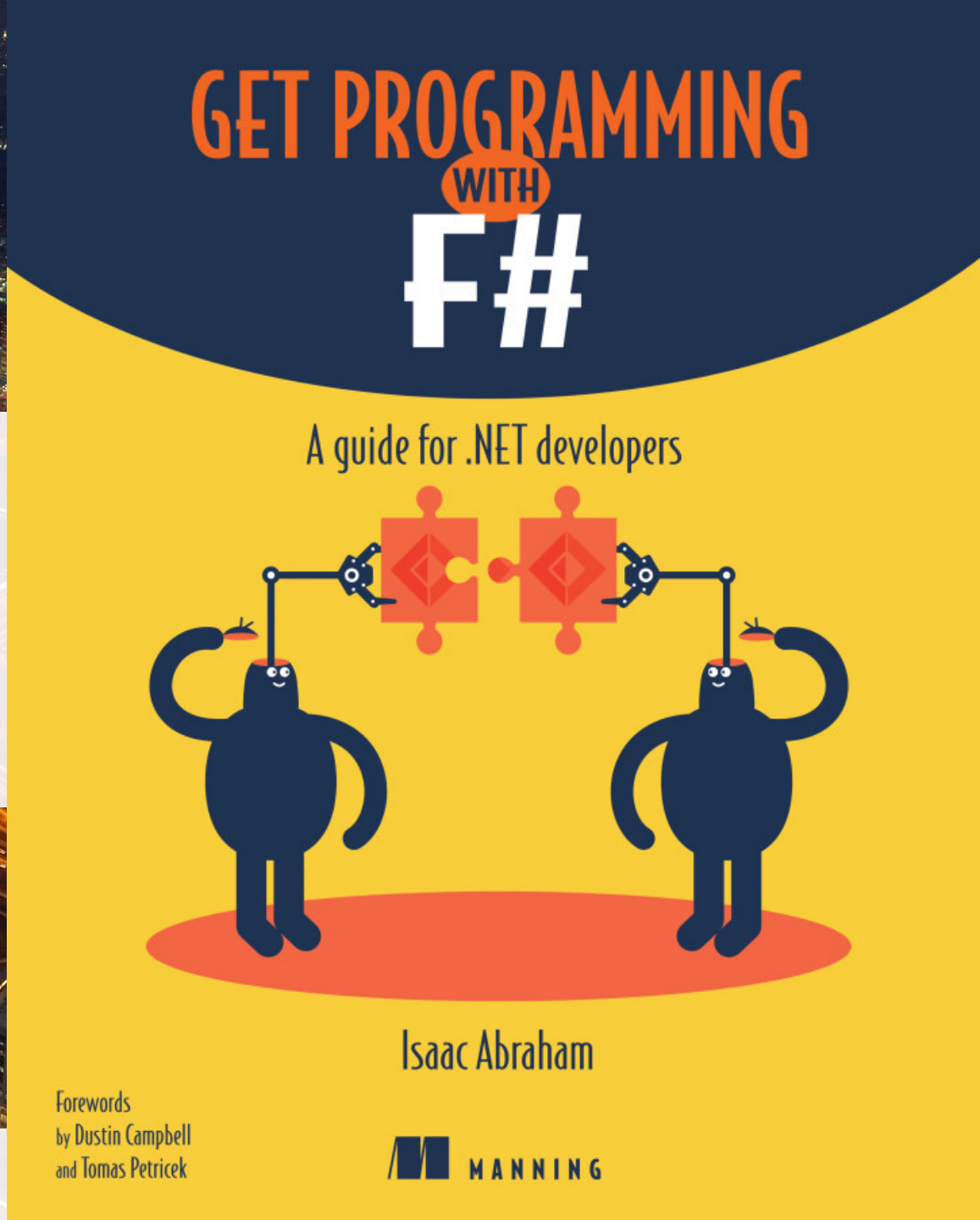
Learn Design patterns after learning the language





How to get started with F#?





REY TECHNOLOGY

Takeaways

REY TECHNOLOGY

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!

REY TECHNOLOGY

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

MVP Microsoft® Most Valuable Professional