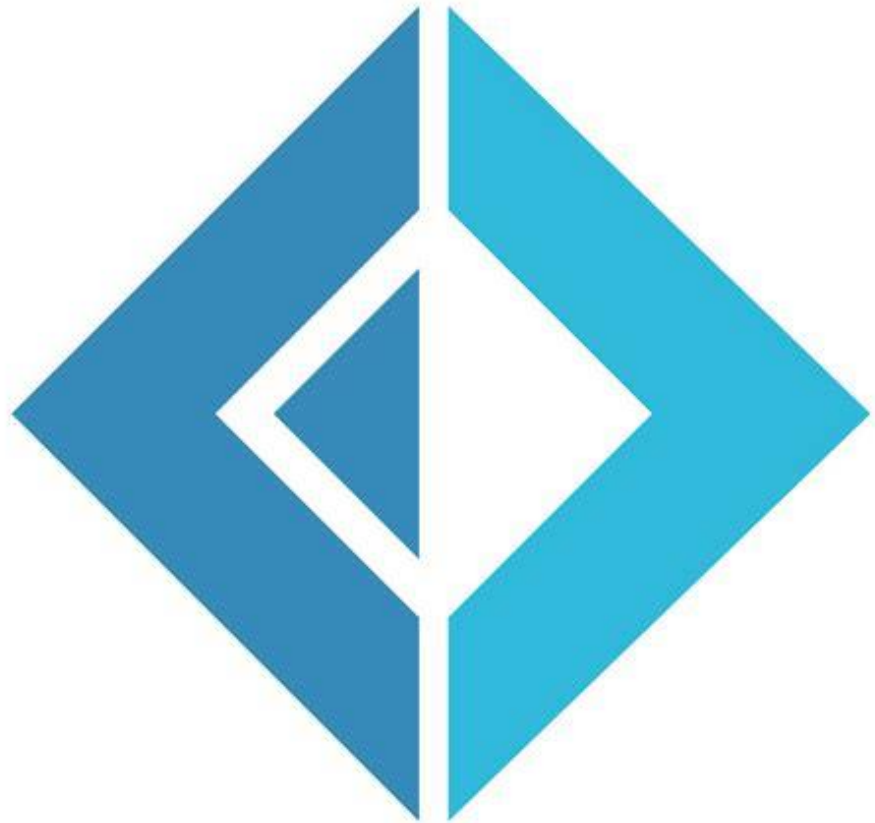
Fabulous Functional Frontends

Mark Allibone

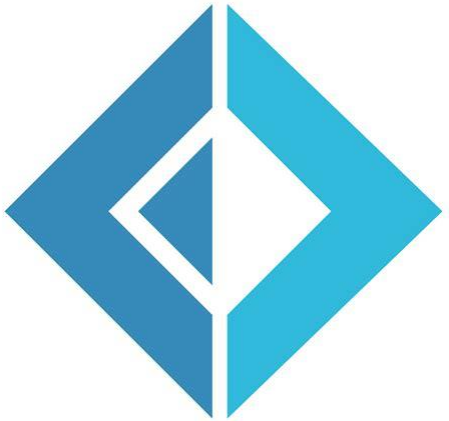
Head of Mobile

Noser Engineering AG

@mallibone



- No Nulls
- No mutable variables
- Data handling
- Declarative Programming
- Based on .Net



F# is great for BLOBs, FLOBs – overall a great general purpose language

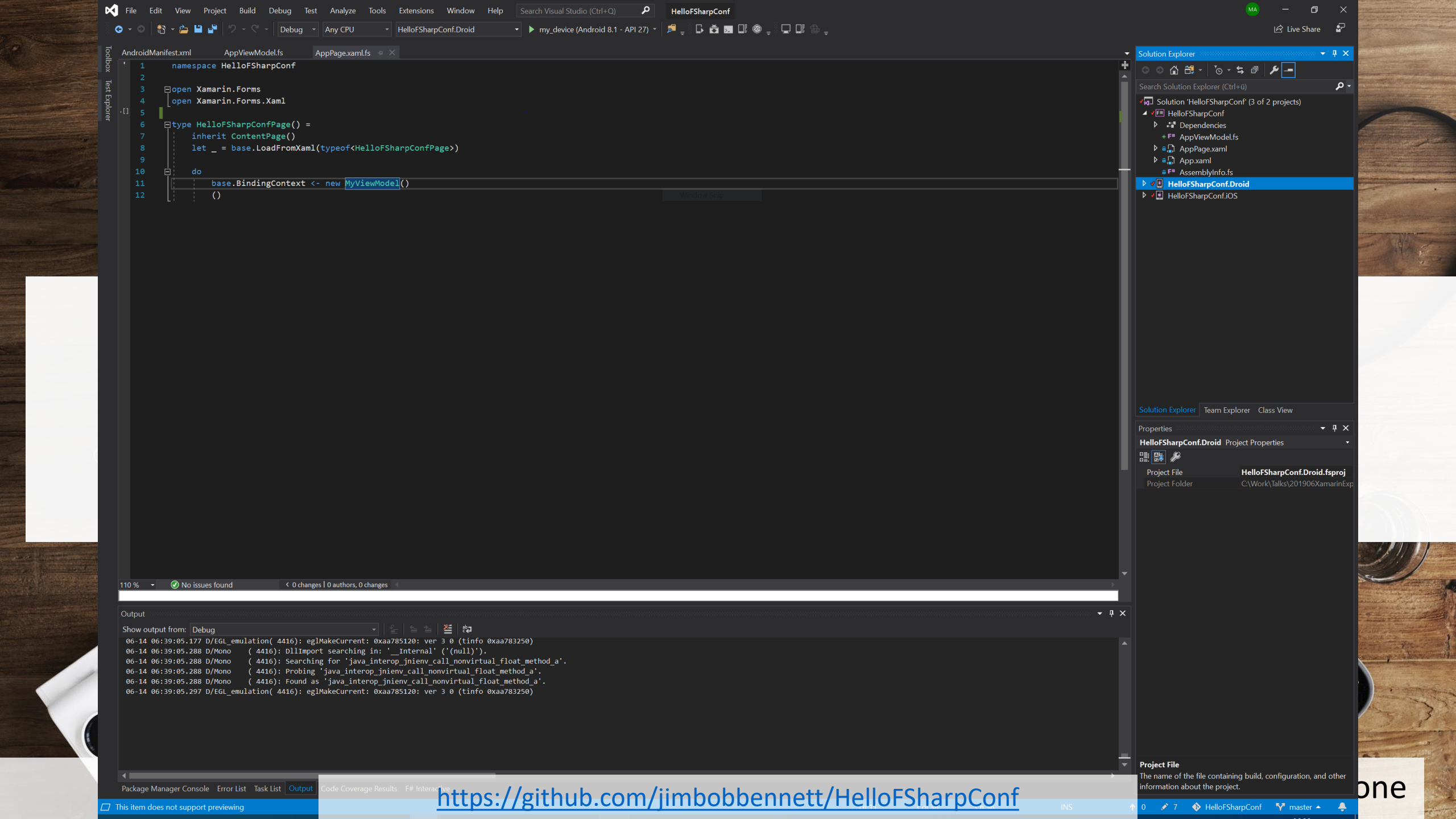
@mallibone

Fabulous – a functional Wrapper for F# around Xamarin.Forms

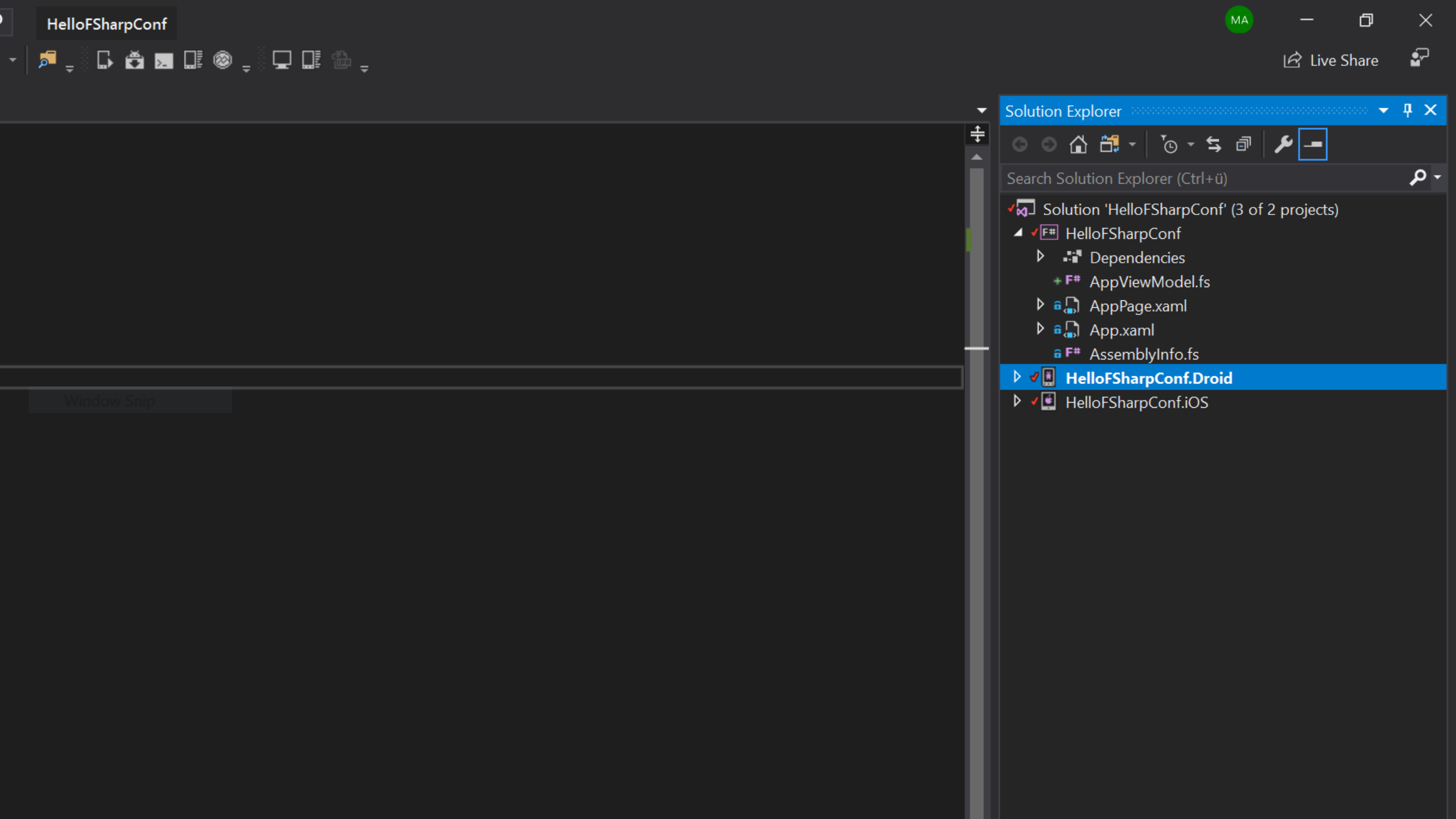
@mallibone

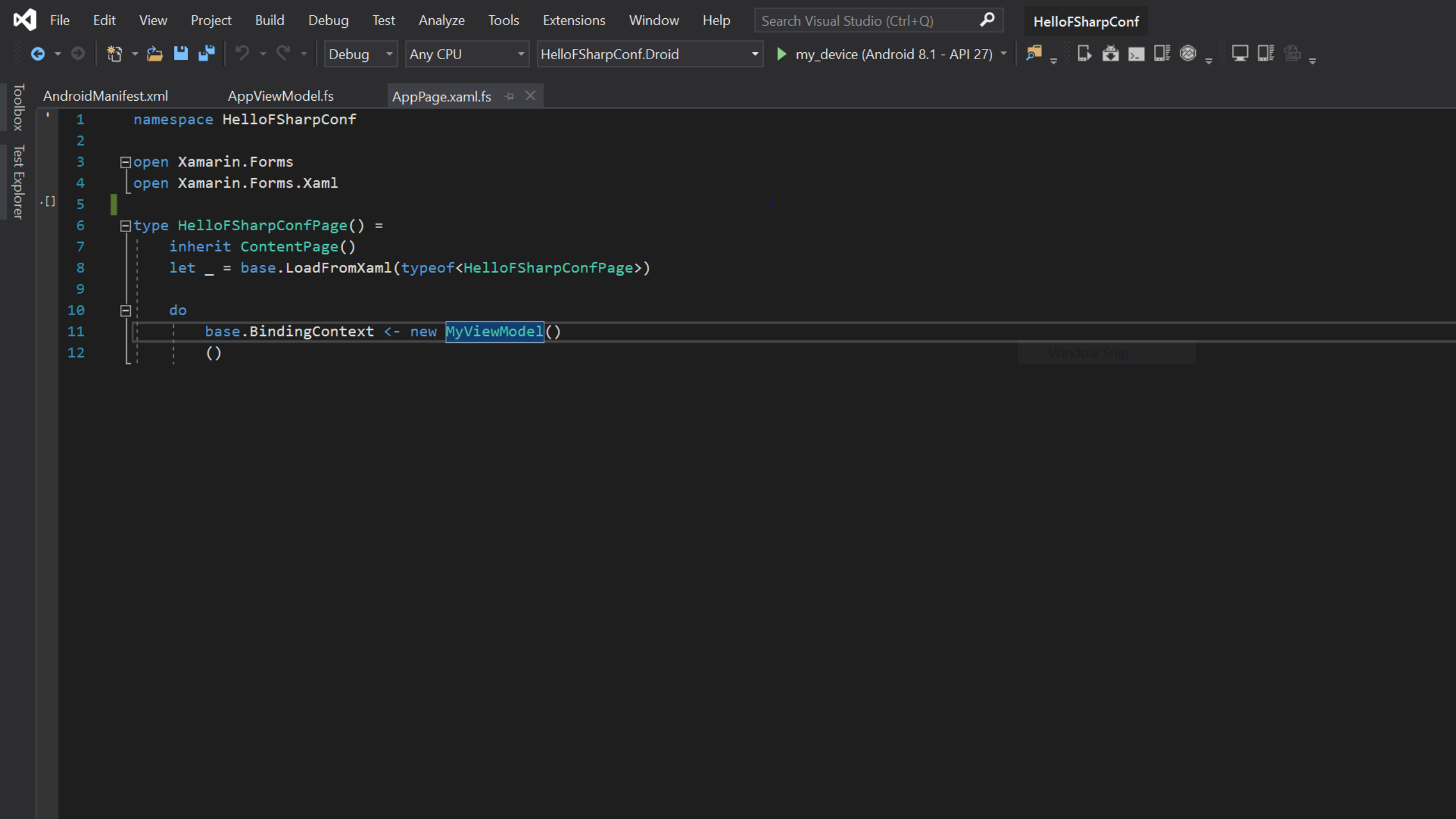
Why not just Xamarin Forms?!

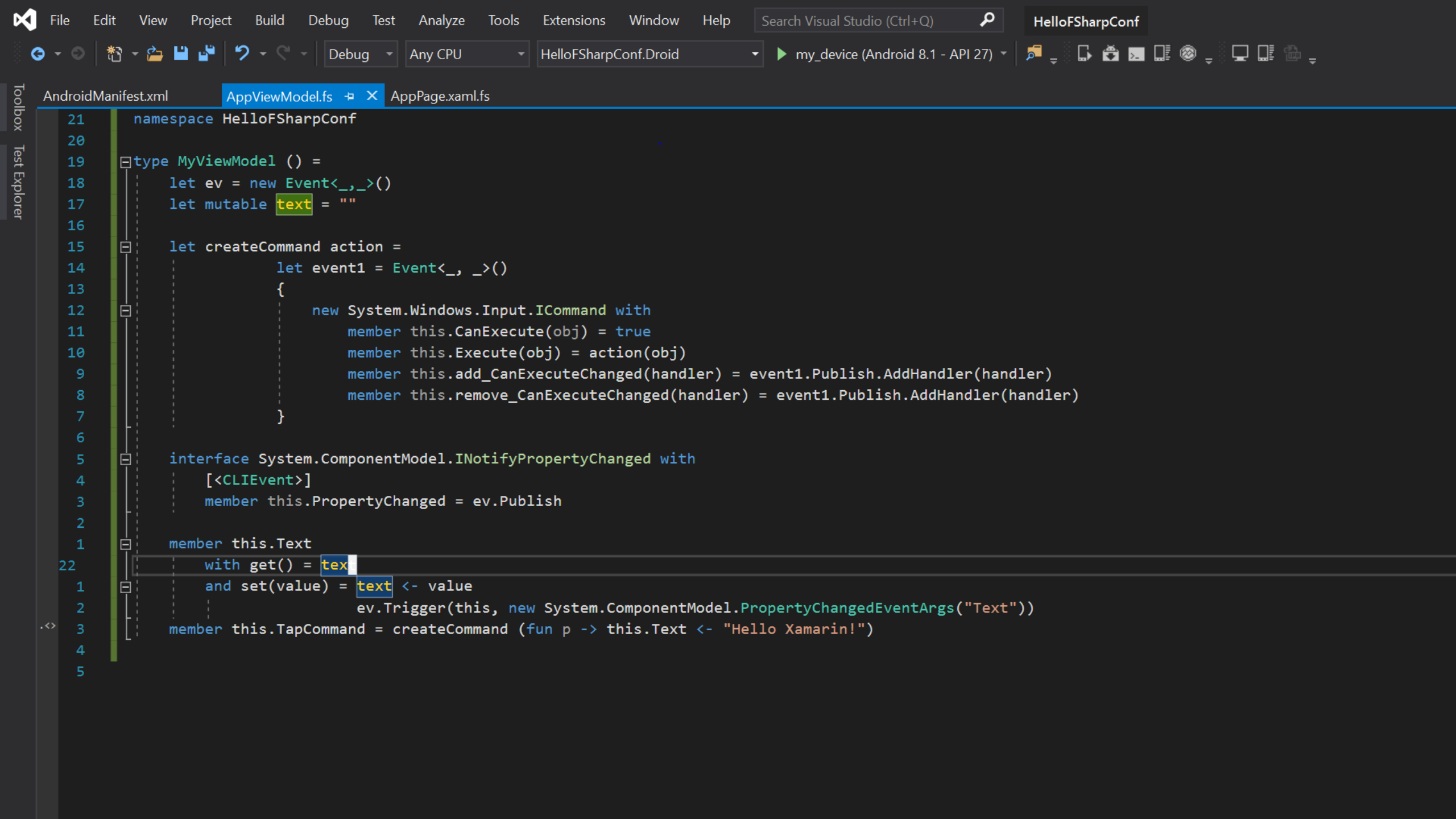
@mallibone

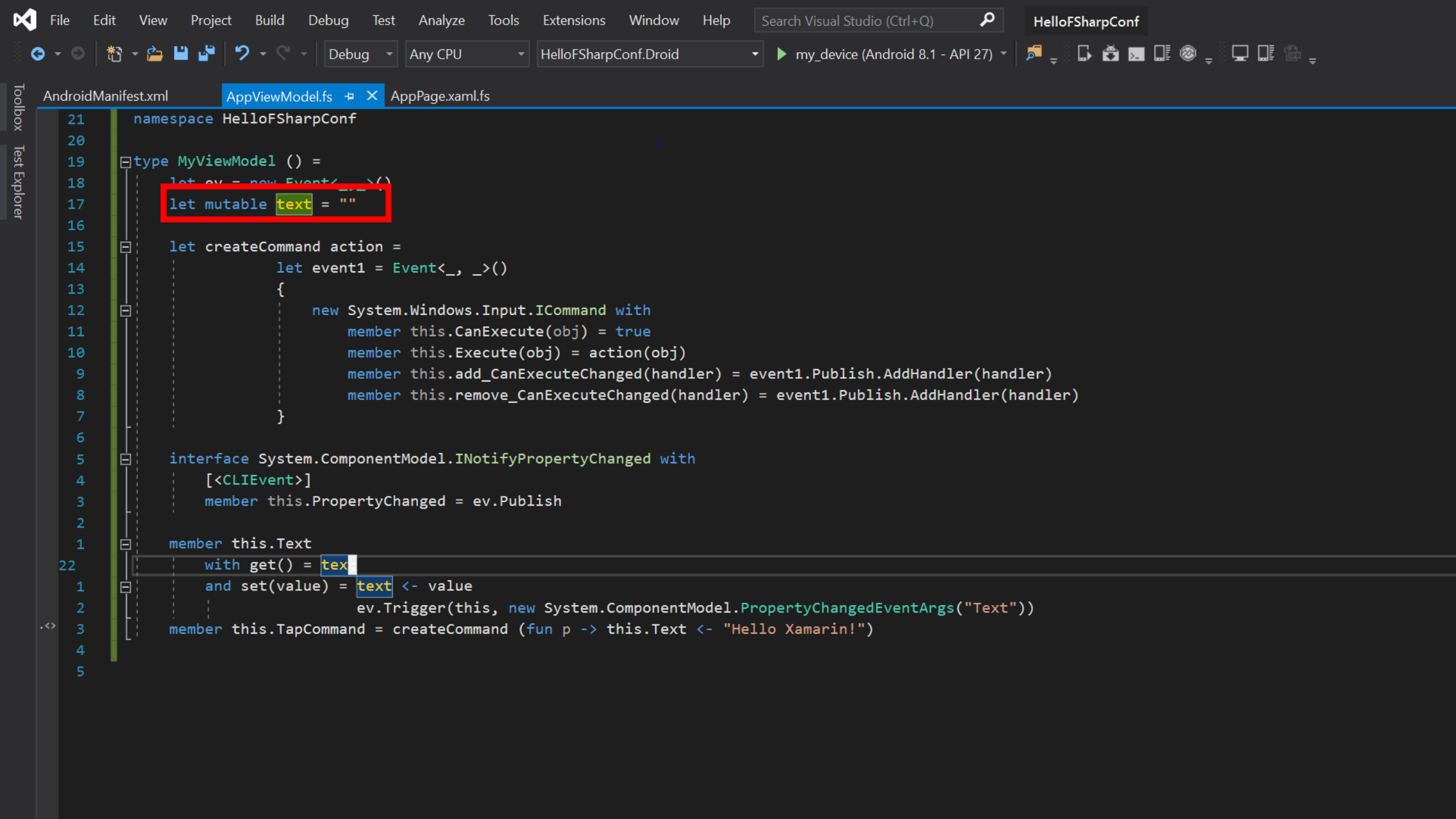


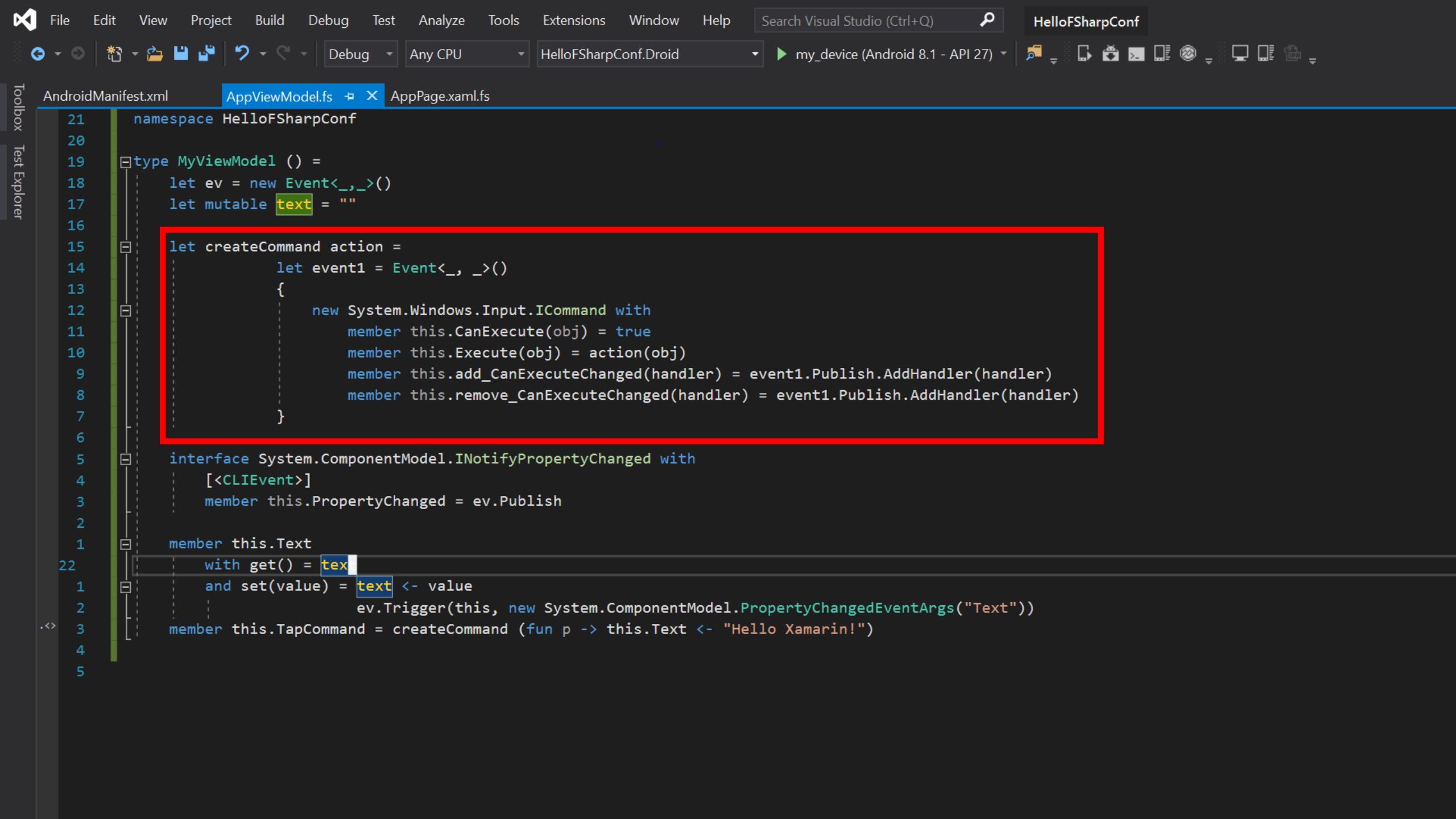
https://github.com/jimbobbennett/HelloFSharpConf

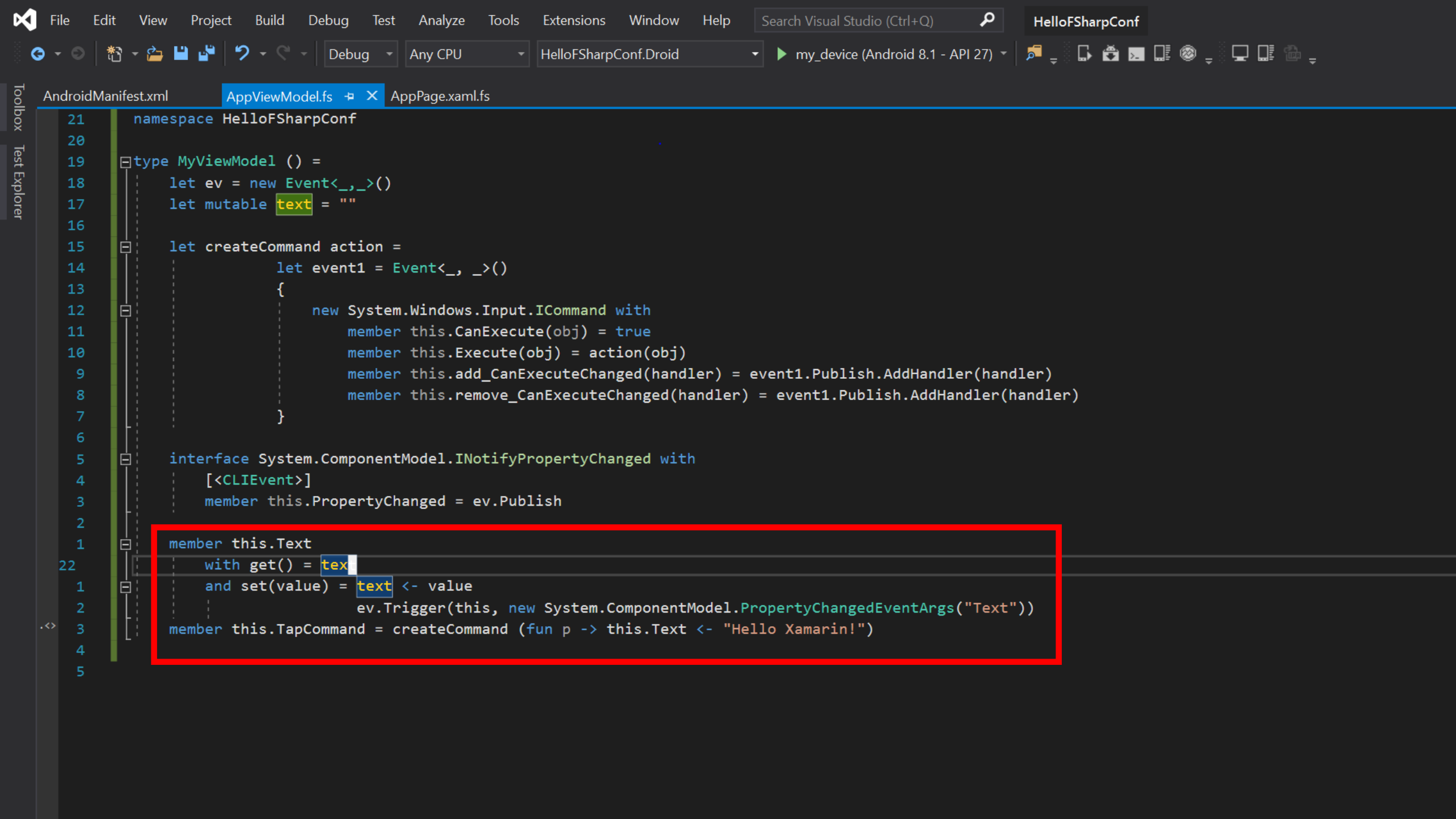

HelloFSharpConf
MA
Live Share
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'HelloFSharpConf' (3 of 2 projects)
HelloFSharpConf
Dependencies
AppViewModel.fs
AppPage.xaml
App.xaml
AssemblyInfo.fs
HelloFSharpConf.Droid
HelloFSharpConf.iOS
Window Snip

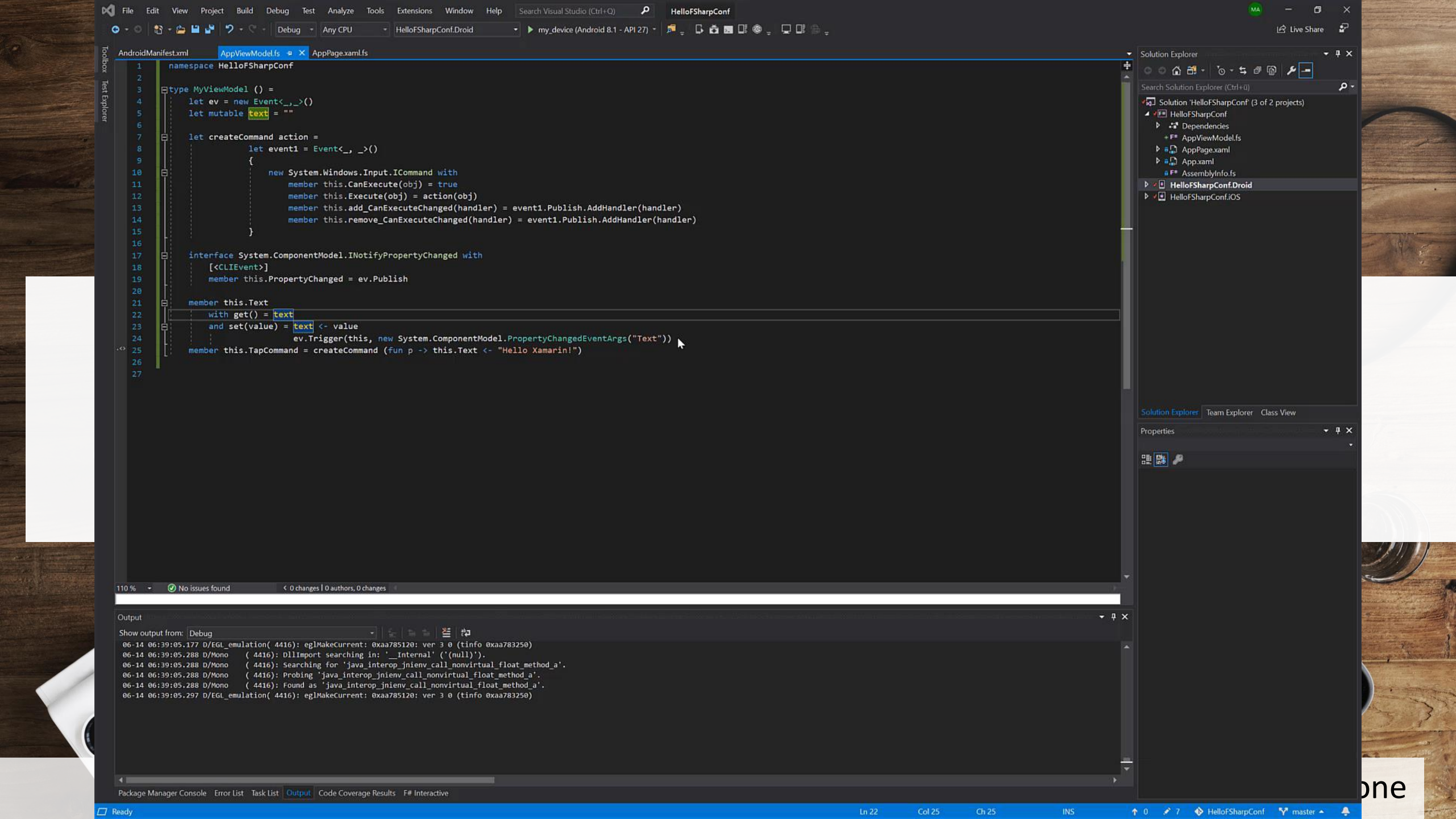

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
Toolbox
Test Explorer
1 namespace HelloFSharpConf
2
3 open 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 ()


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() = 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)
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

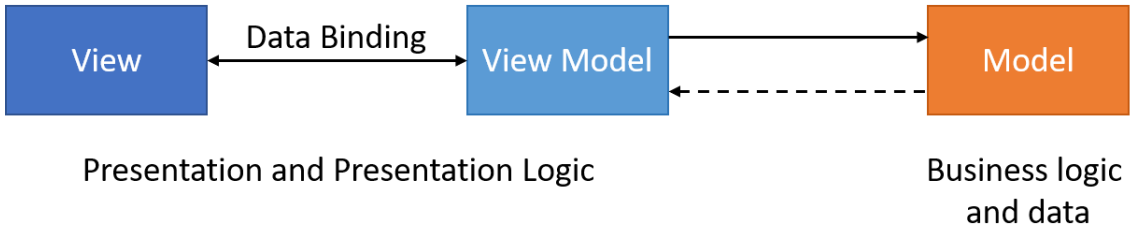
F# can do Object Oriented programming but it is functional first

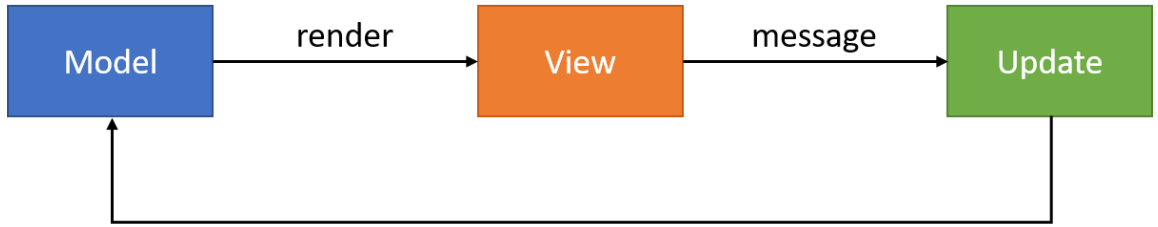
@mallibone

MVU the functional MV* View Pattern

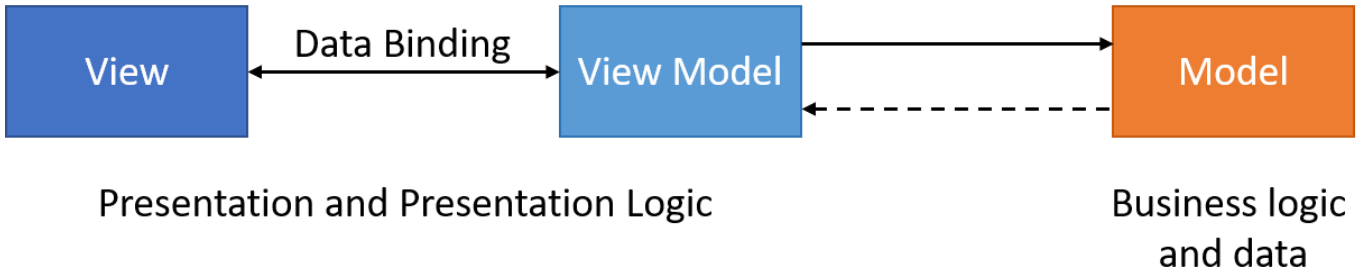
@mallibone

MVVM vs MVU

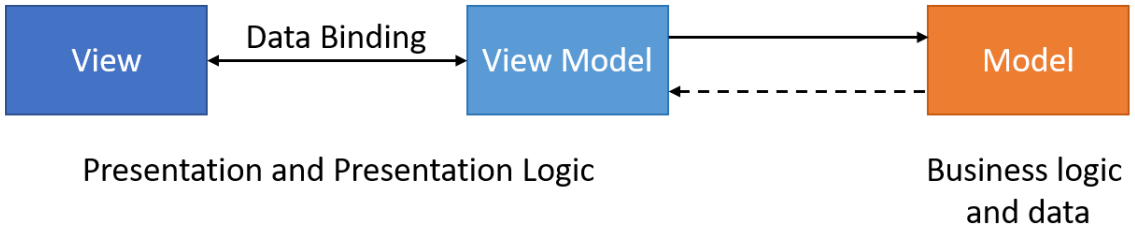

View
Data Binding
View Model
Model
Presentation and Presentation Logic
Business logic
and data

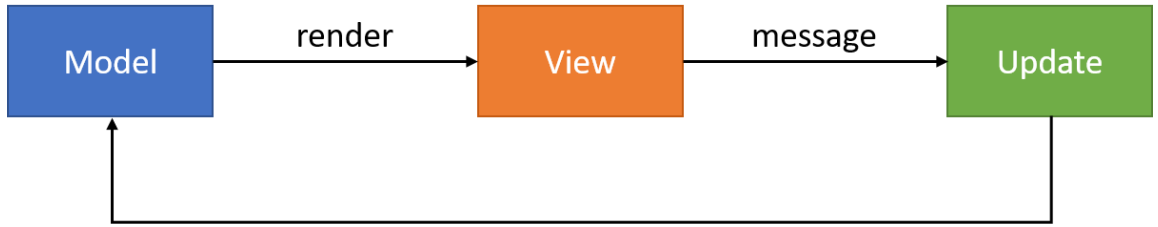

Model
render
View
message
Update

Model View View Model

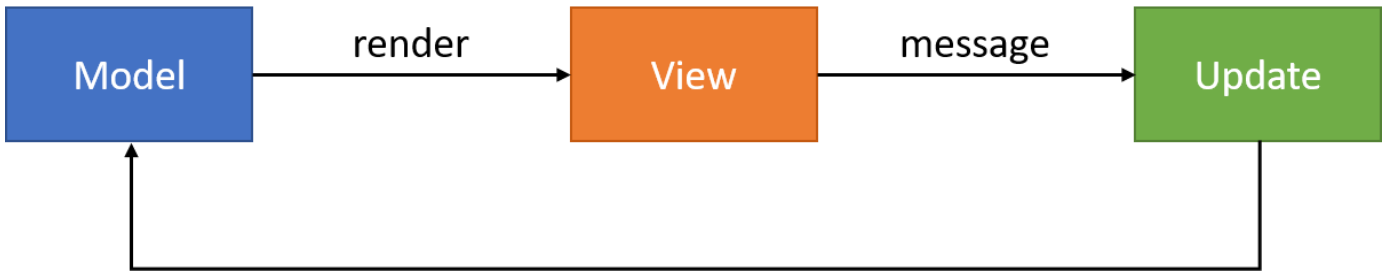


MVVM vs MVU


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


Model
render
View
message
Update

Model View Update



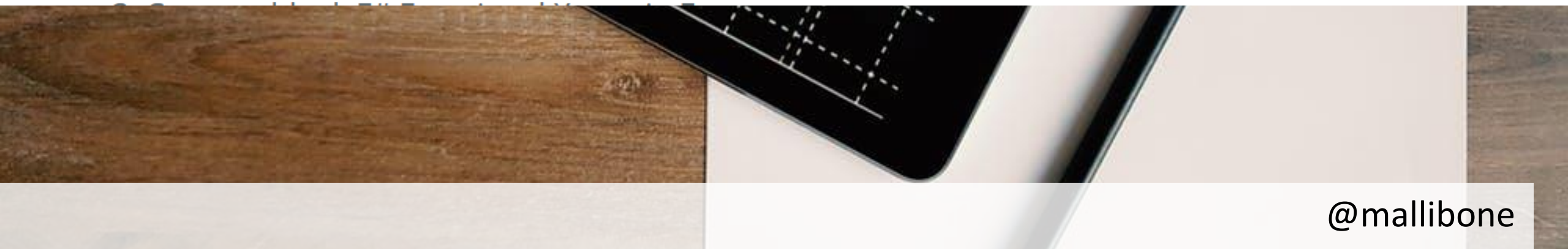
Functional Frontends

Getting started

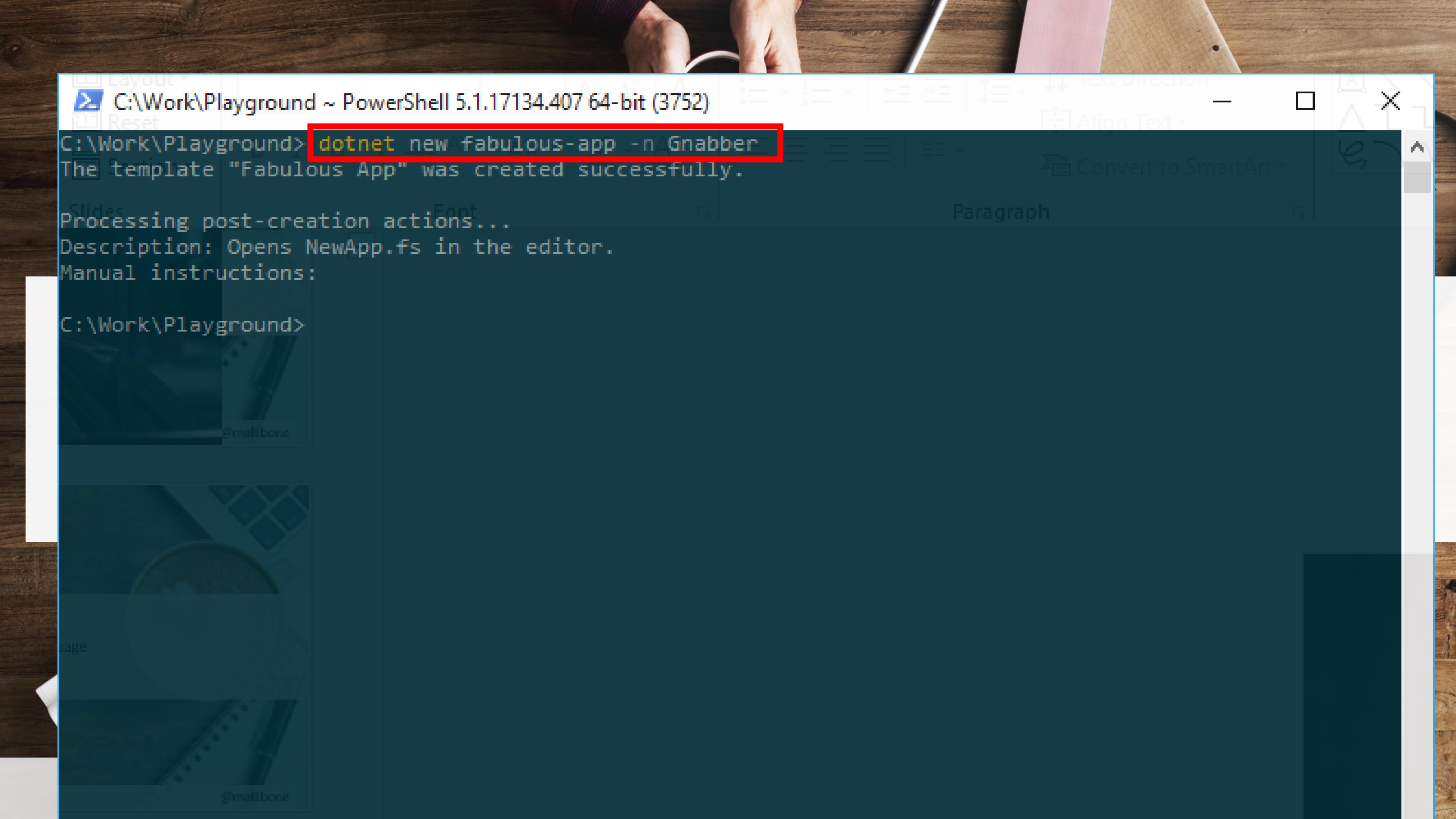
1. Install Visual Studio or Visual Studio for Mac and enable both Xamarin and .NET Core support.

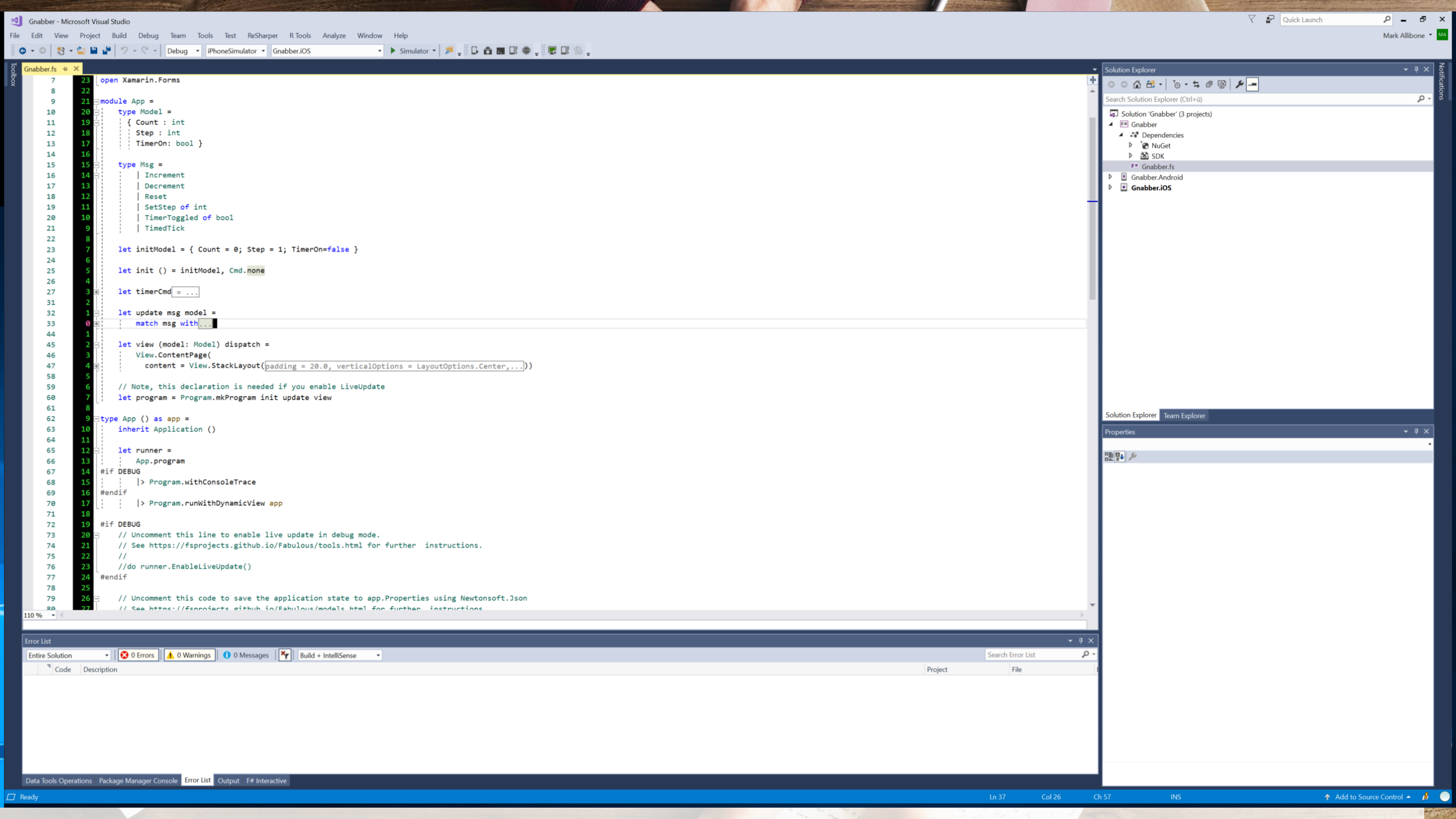
2. Install the template pack:

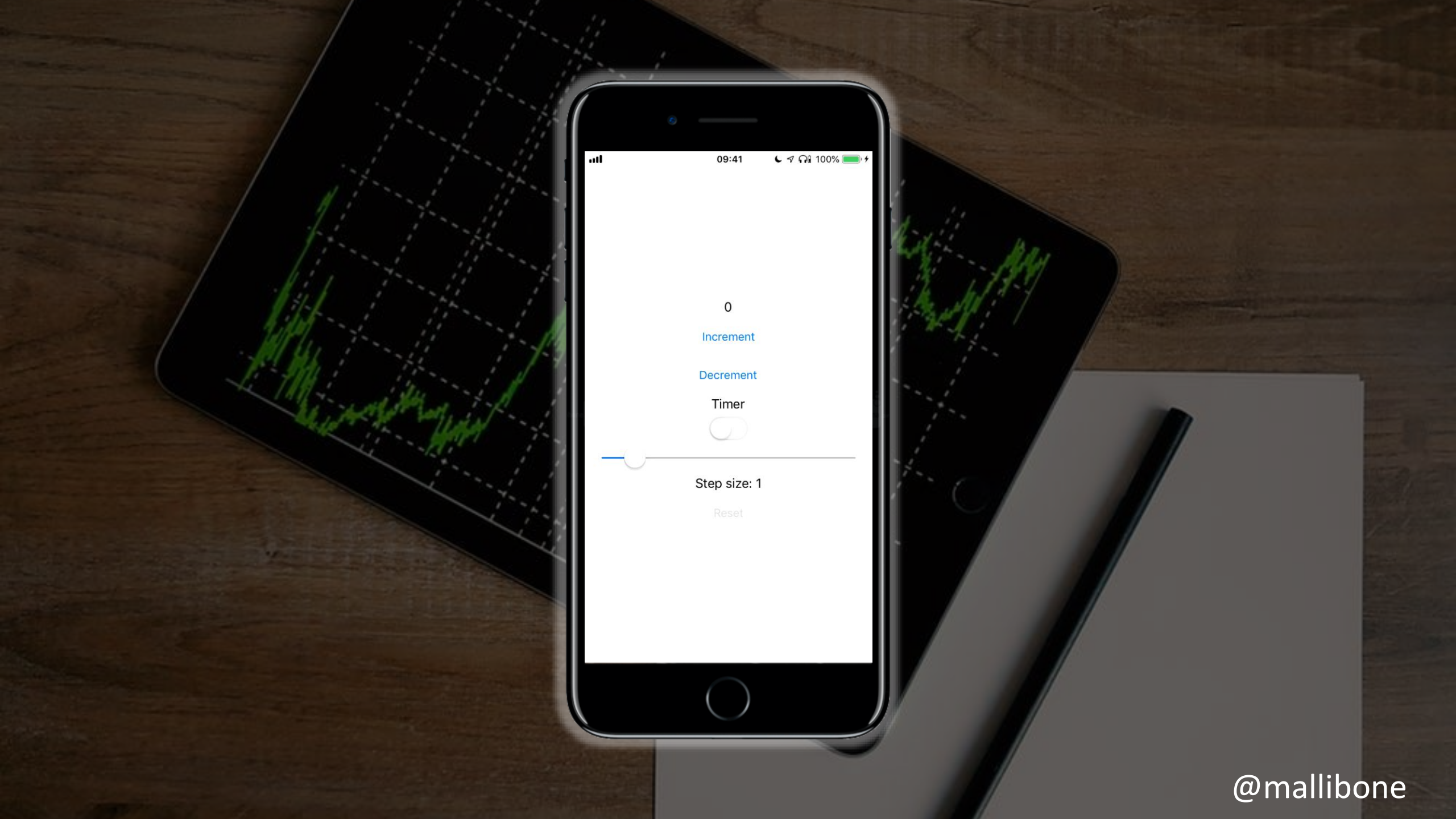
```
dotnet new -i Fabulous.Templates
```



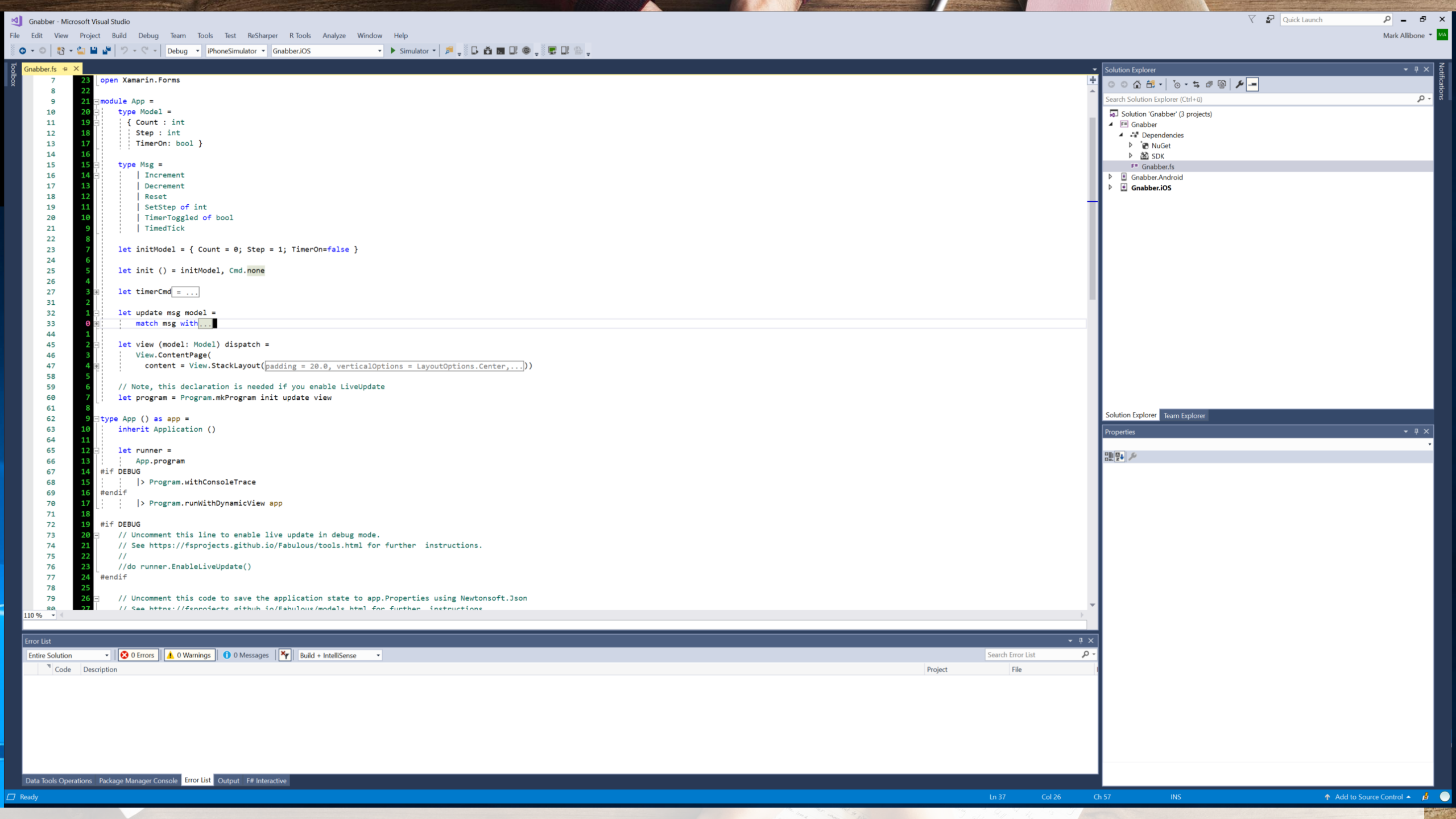
@mallibone

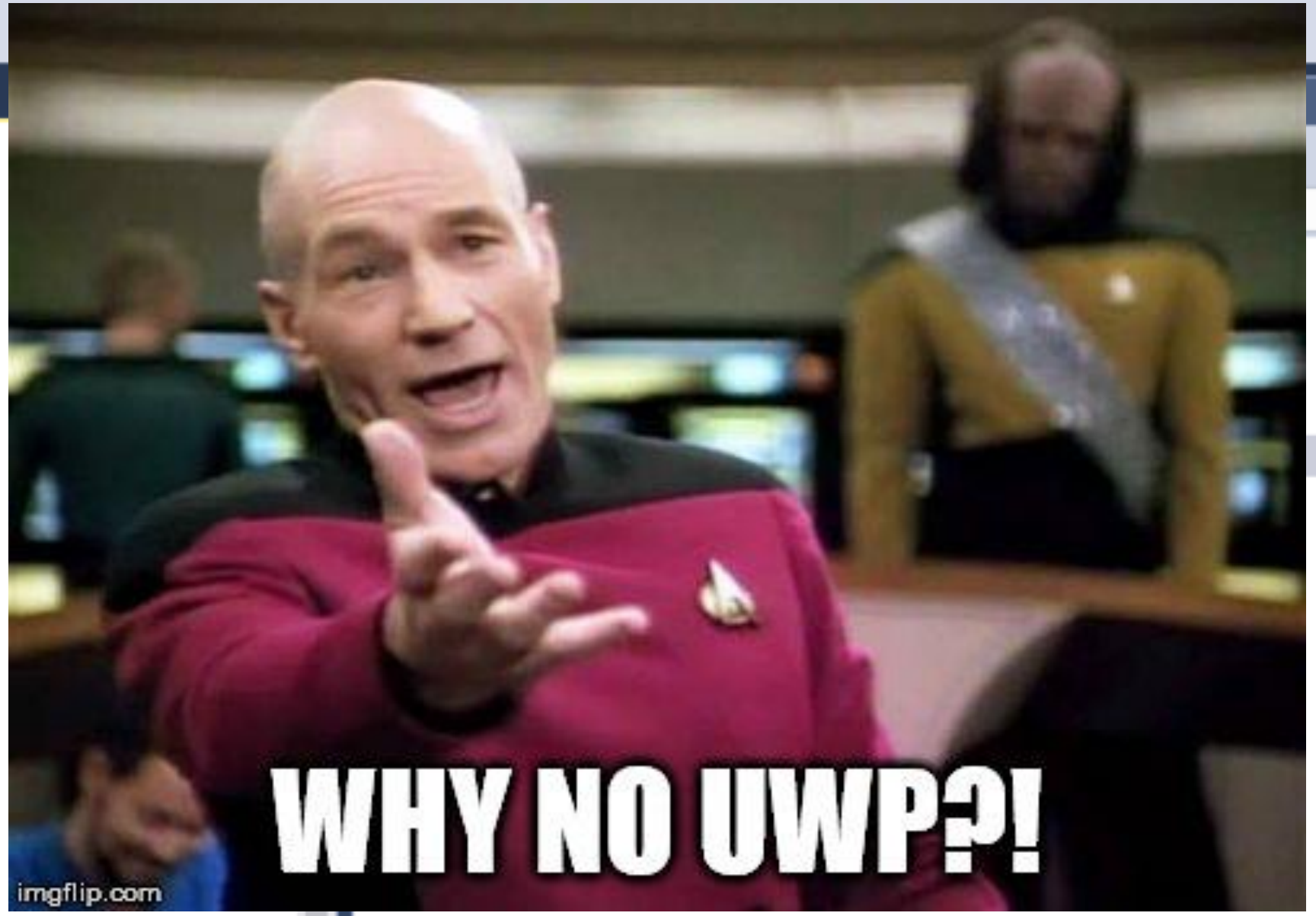

C:\Work\Playground ~ PowerShell 5.1.17134.407 64-bit (3752)
C:\Work\Playground> dotnet new fabulous-app -n Gnabber
The template "Fabulous App" was created successfully.
Processing post-creation actions...
Description: Opens NewApp.fs in the editor.
Manual instructions:
C:\Work\Playground>

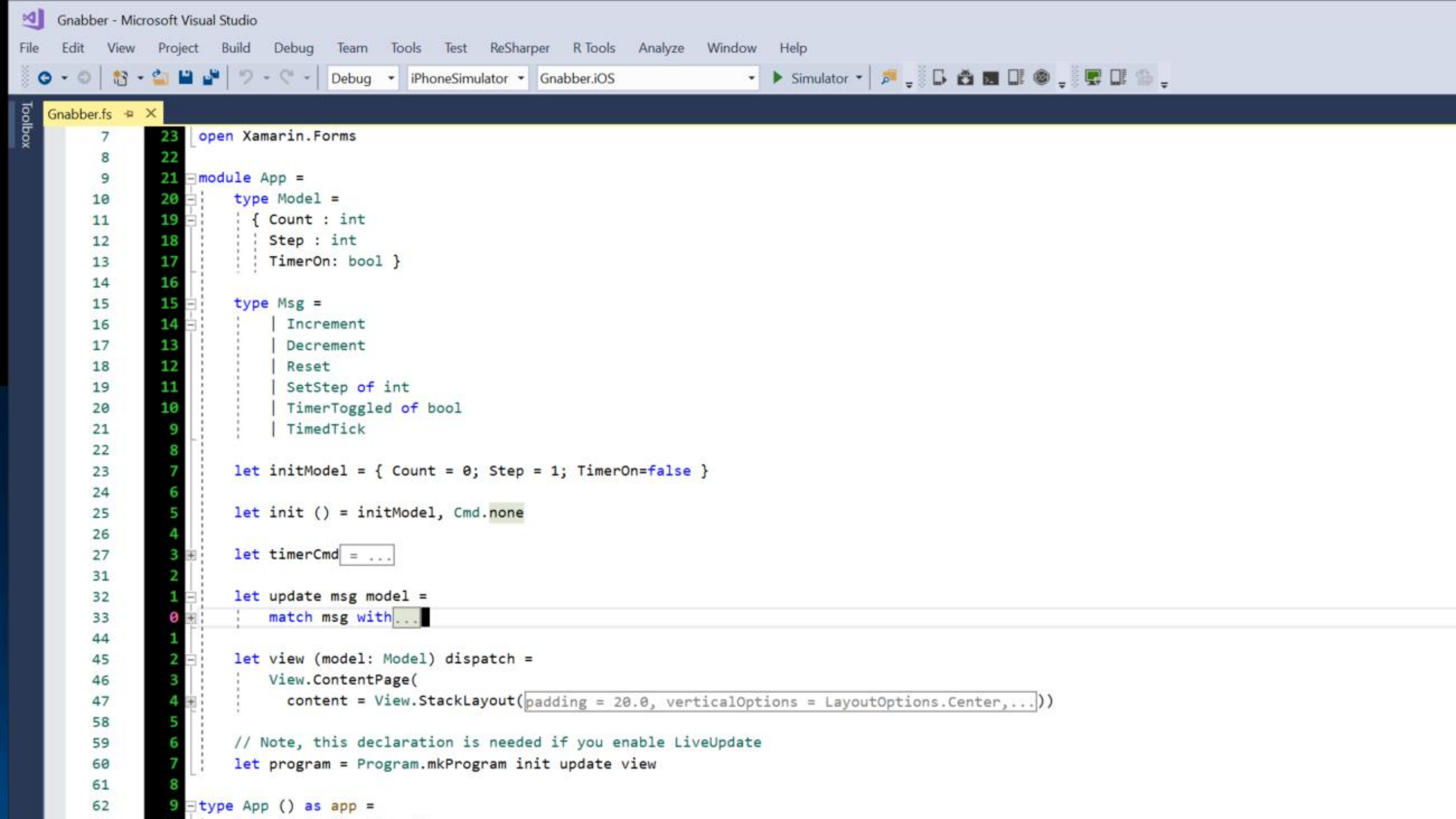

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Mark Allibone
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| 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
110 %
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

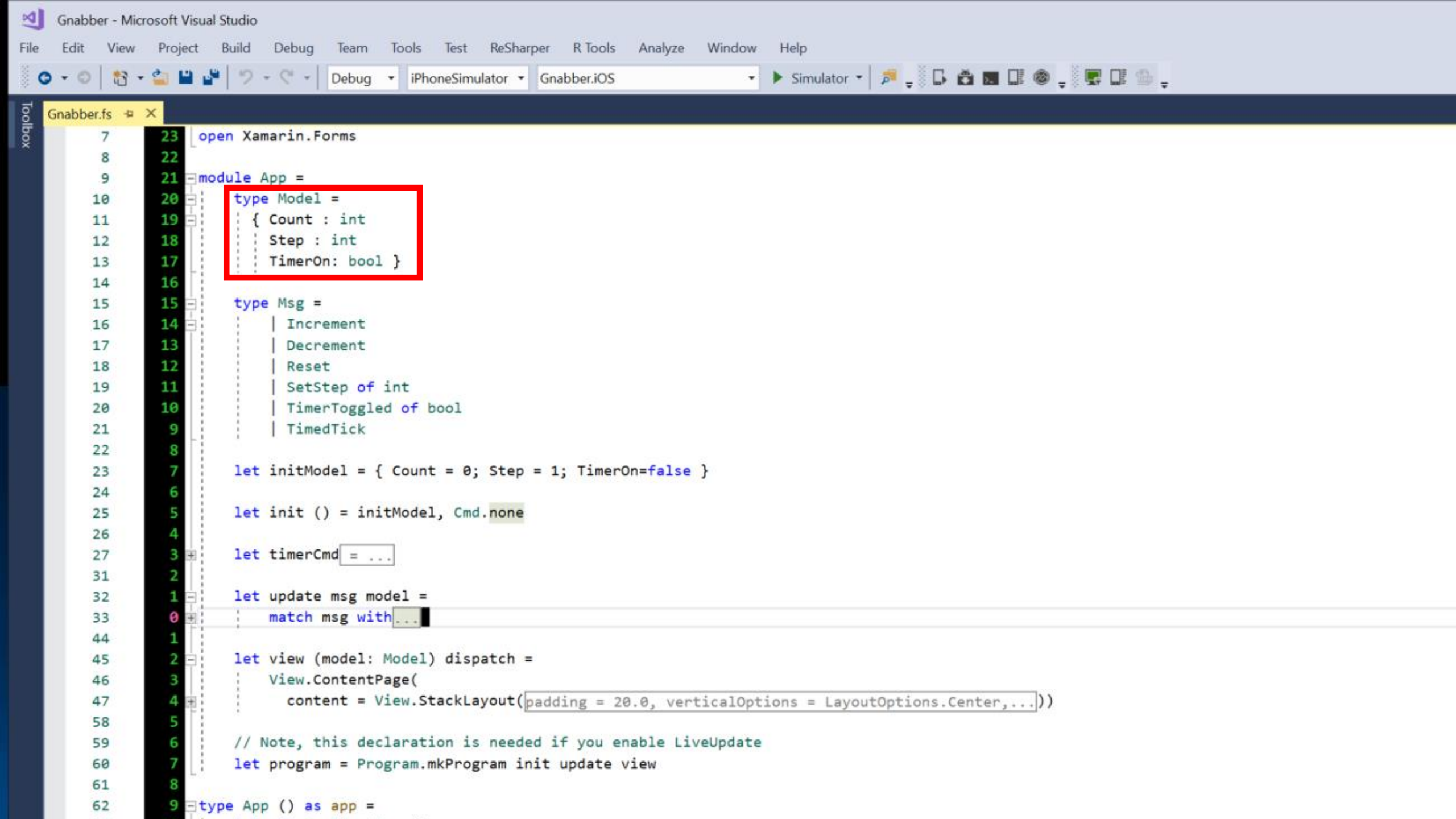


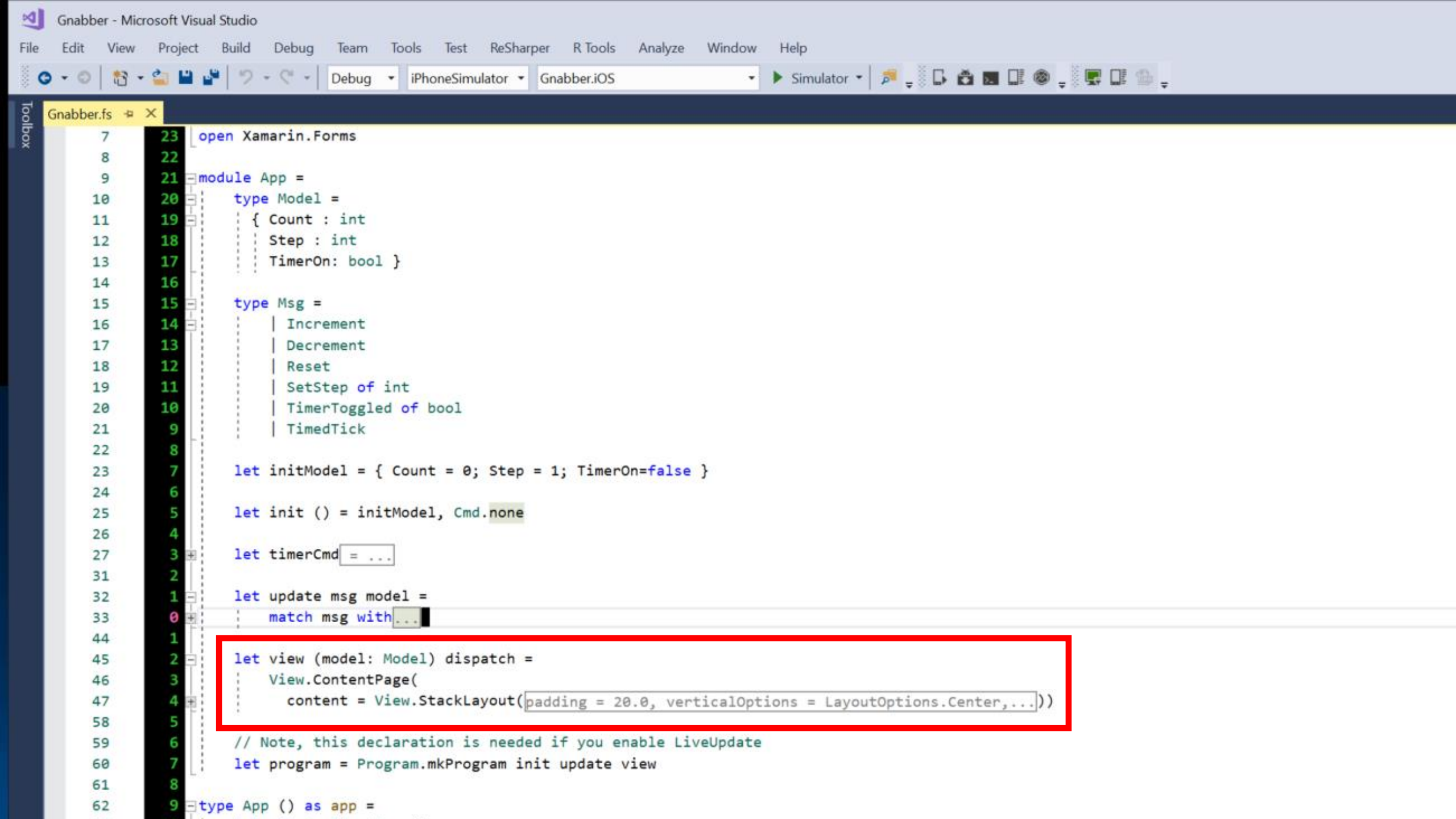
@mallibone

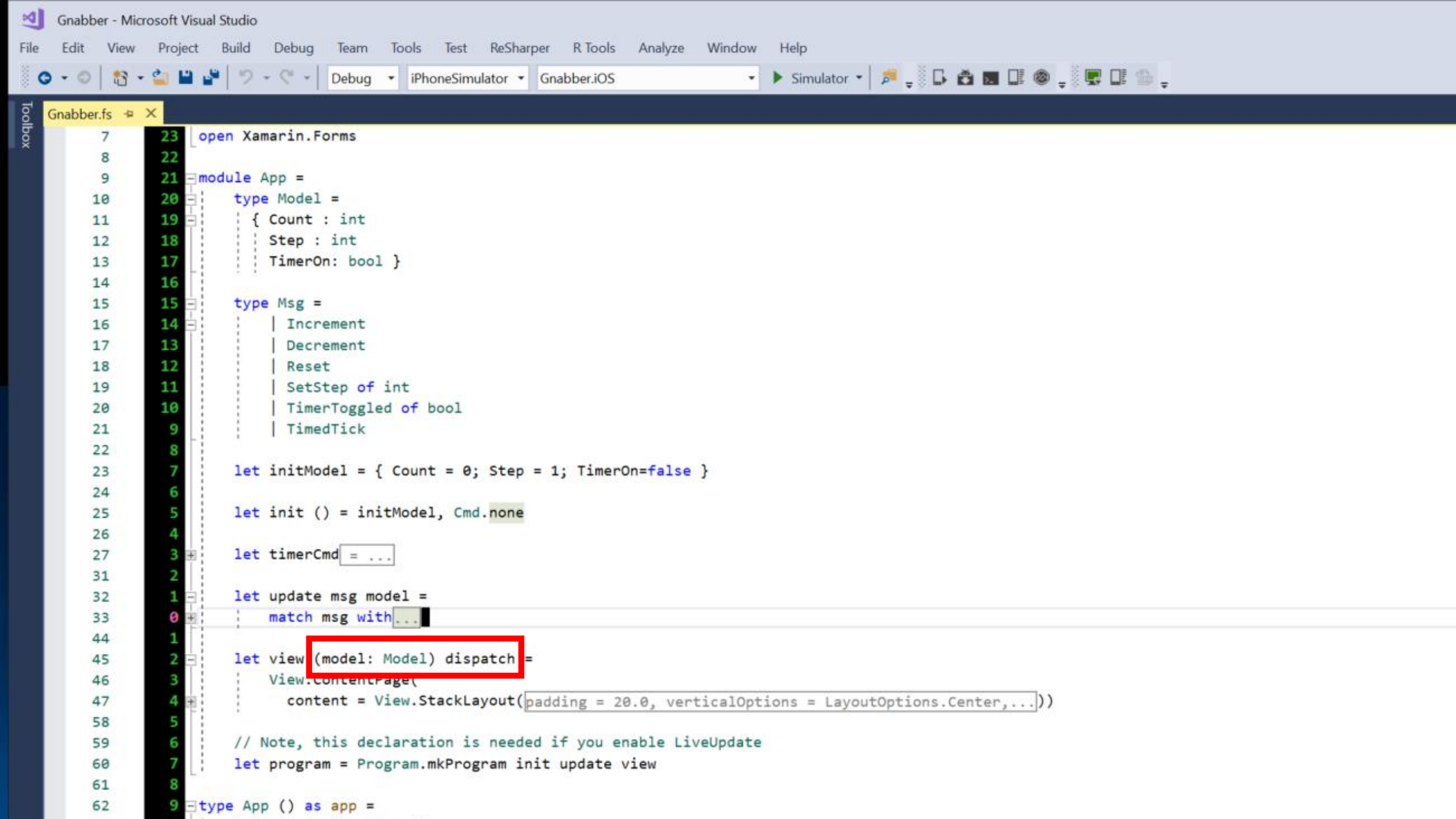

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Mark Allibone
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| 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
110 %
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

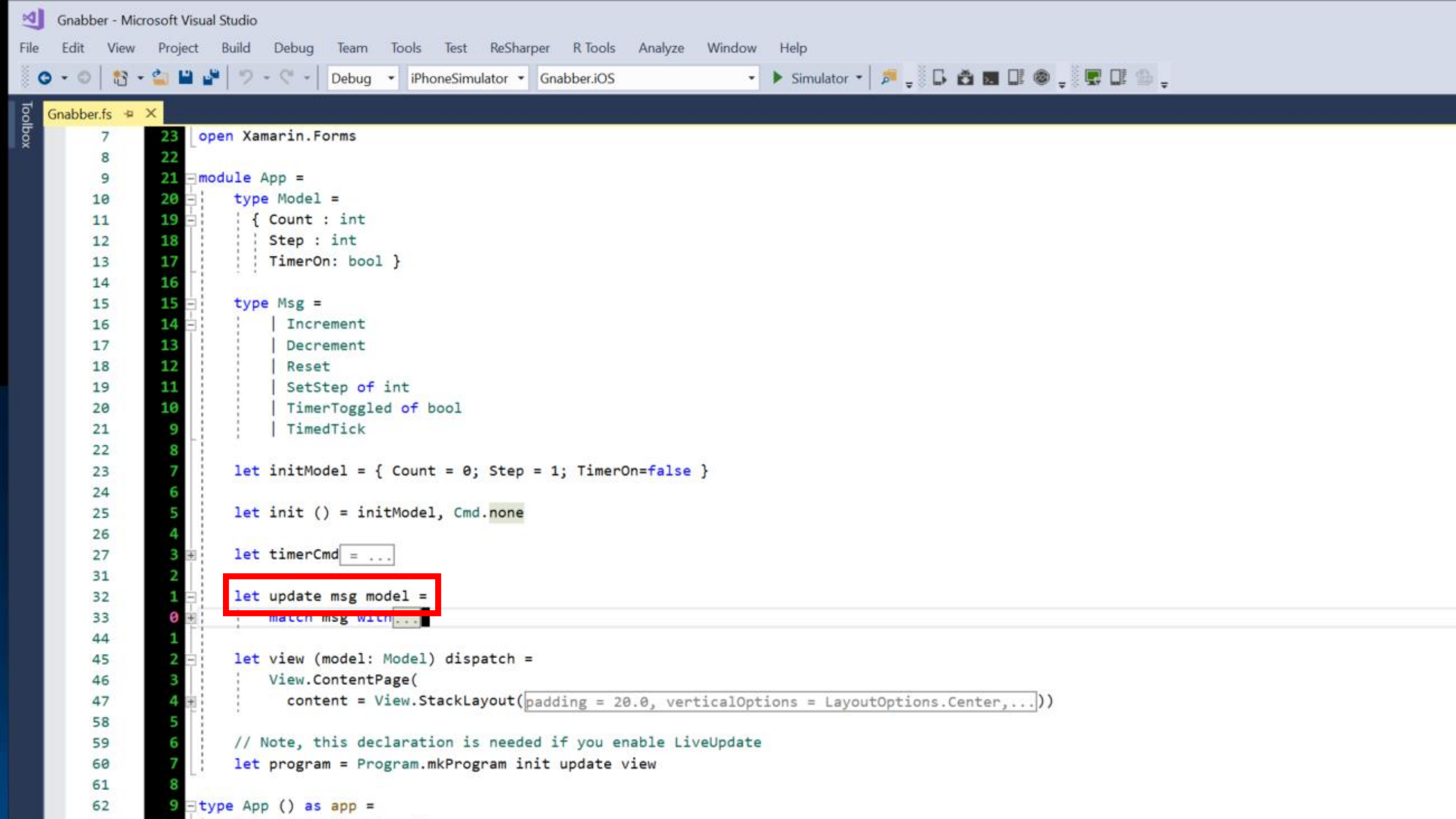

WHY NO UWP?!
imgflip.com

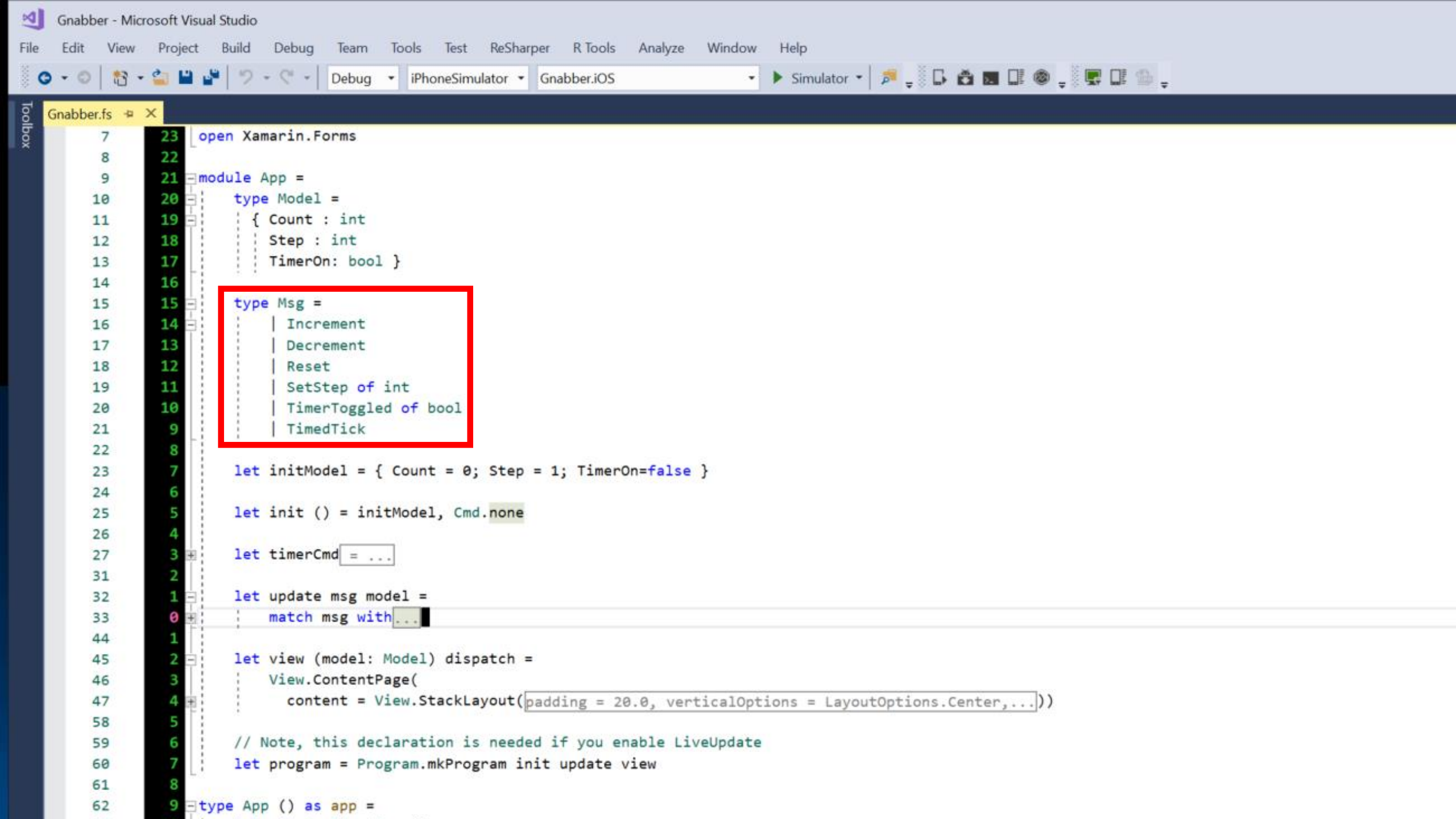

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement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

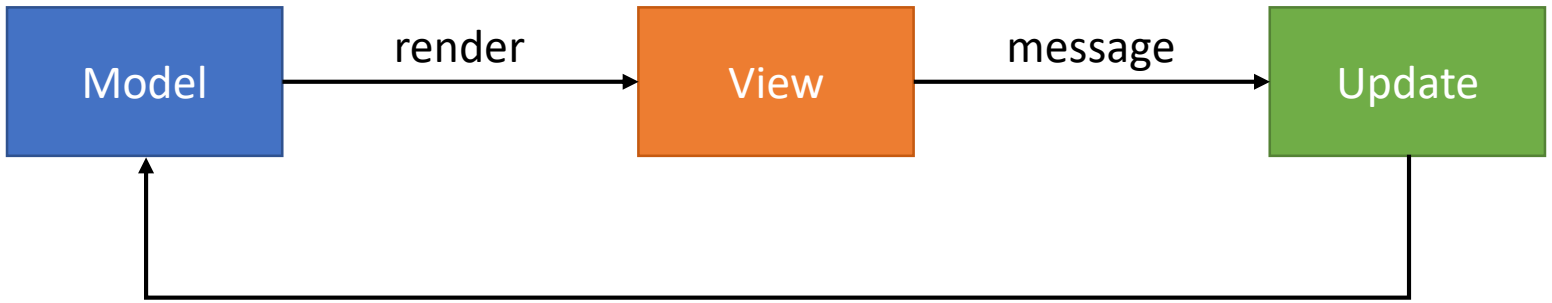

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement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

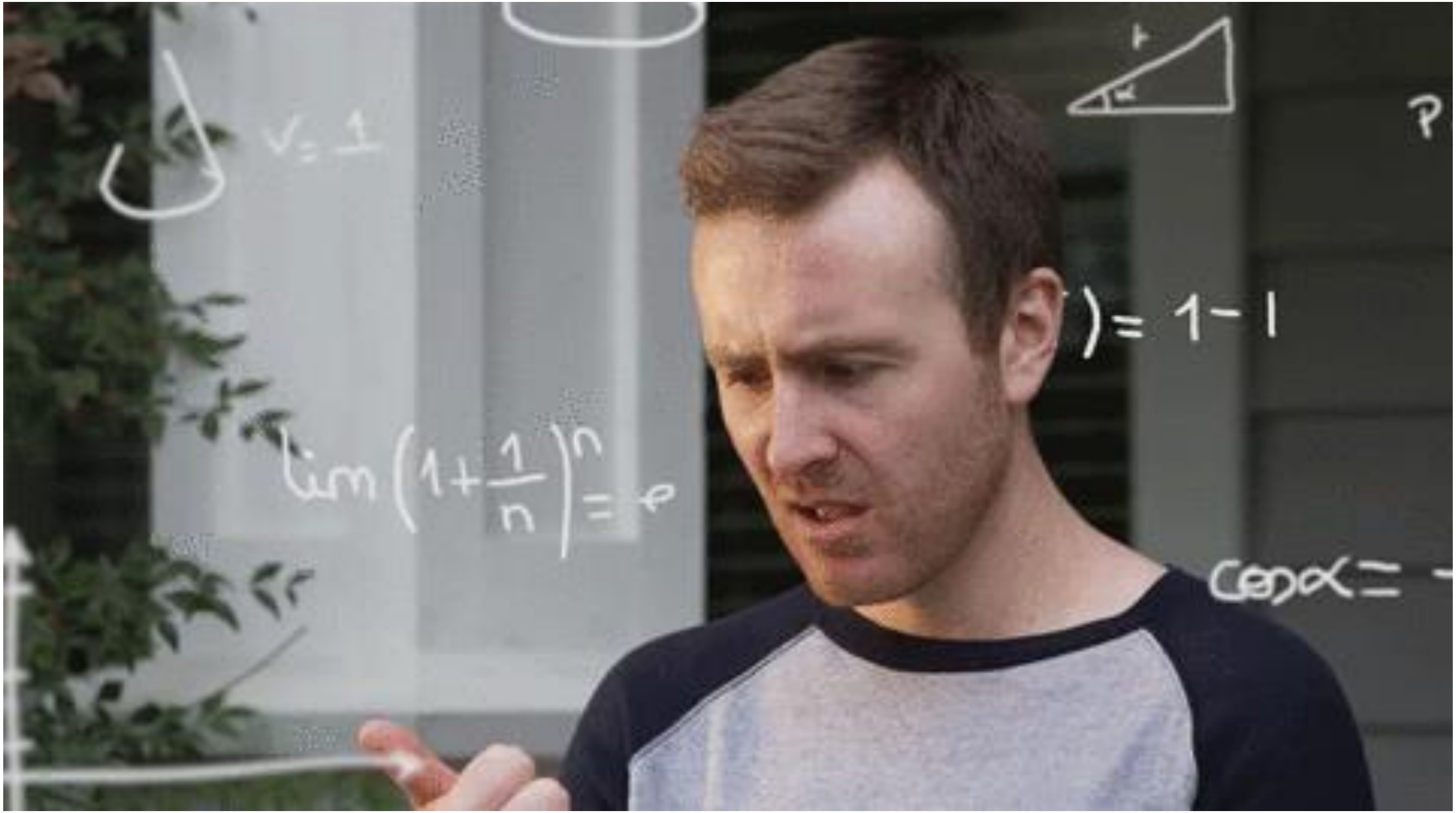

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement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

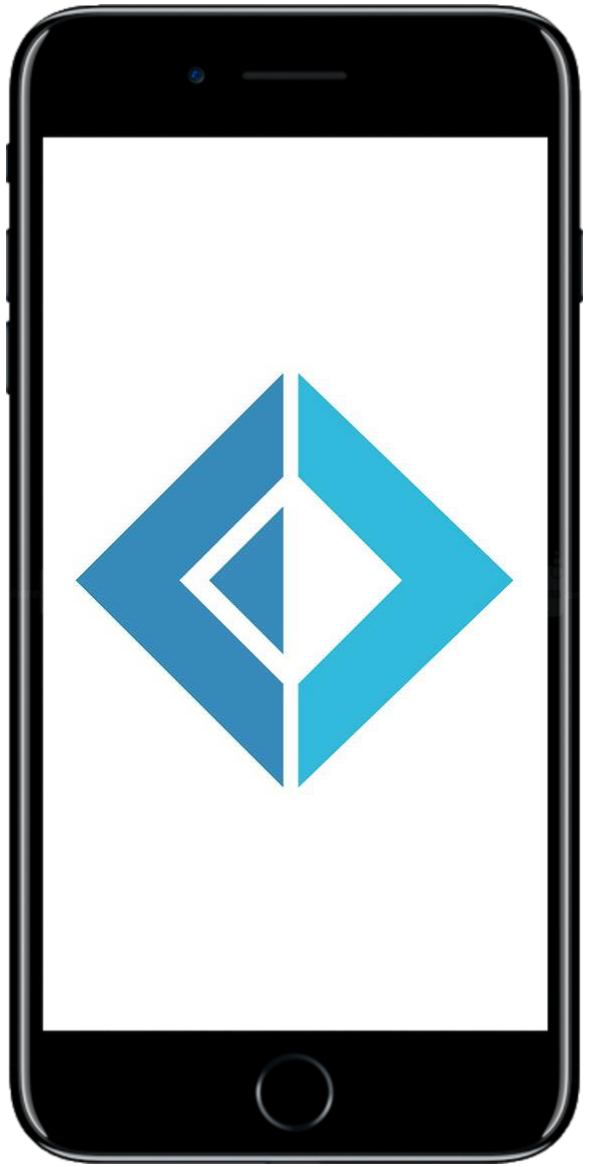

DEMO
BUSINESS
search
@mallibone

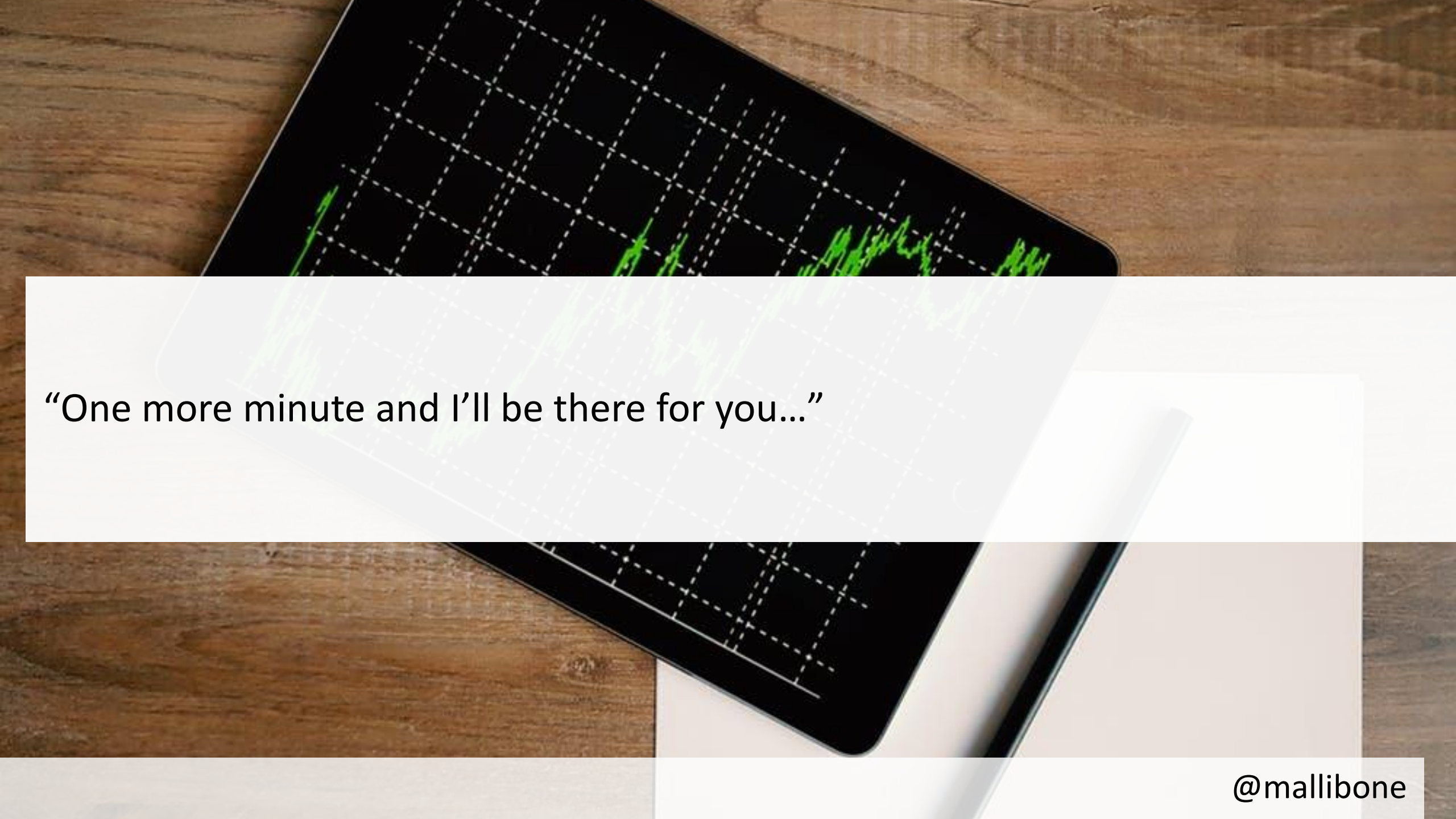
Model View Update



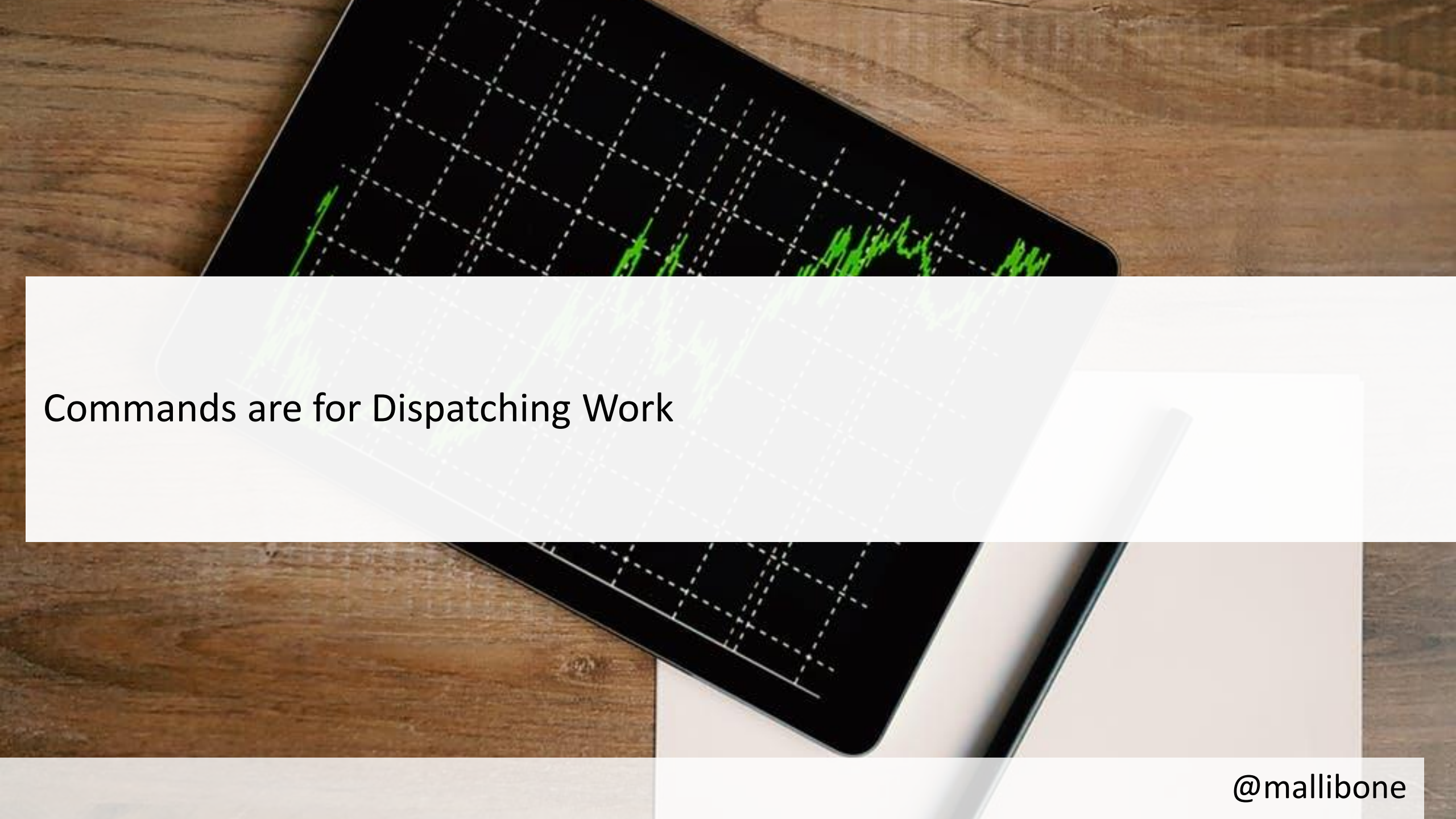

$\lim\left(1+\frac{1}{n}\right)^n = e$

- Driven from the model not the view
- States changes are defined with messages
- Single place for change in the update method
- Based on Xamarin Forms





“One more minute and I’ll be there for you…”



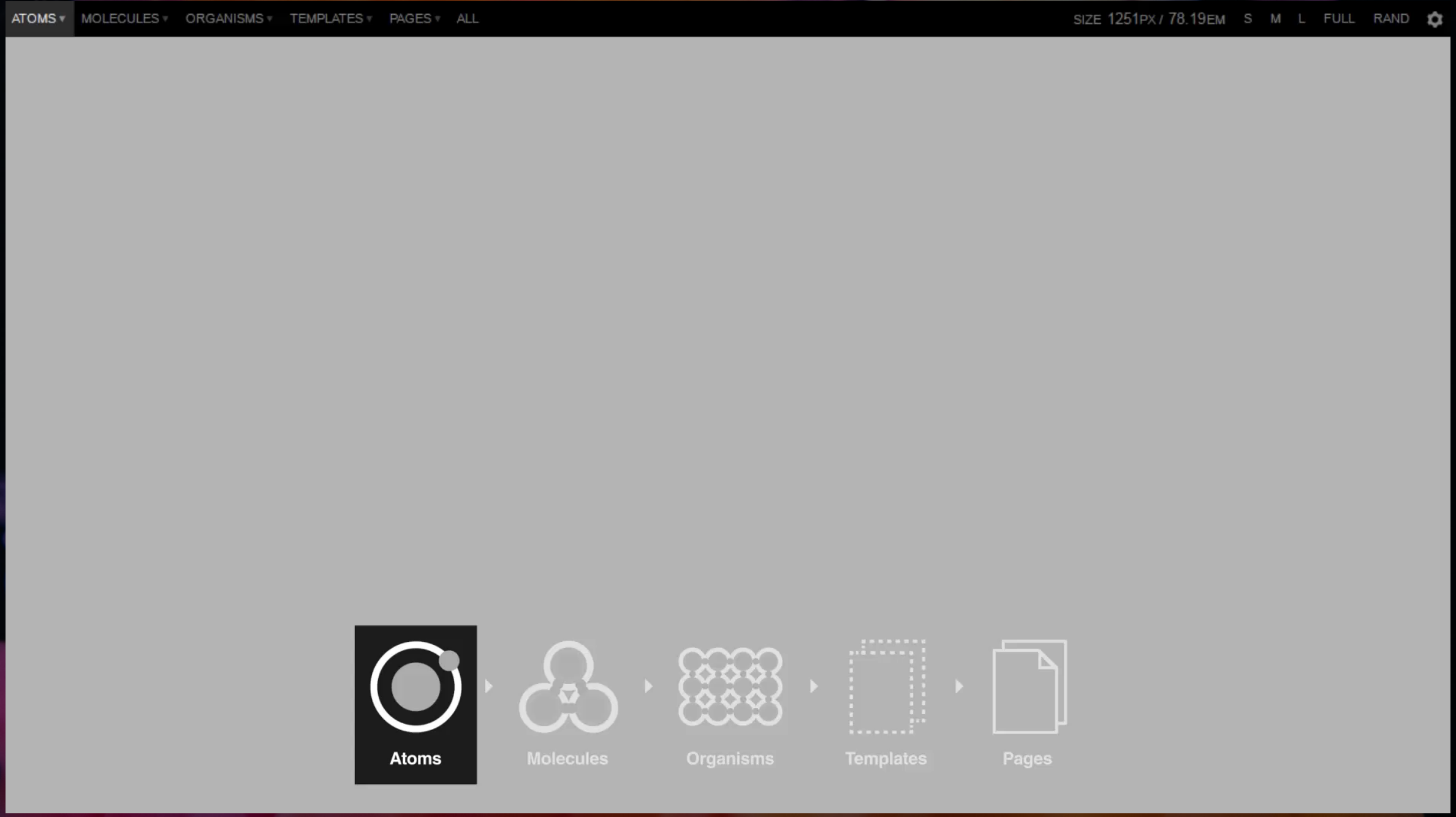
Commands are for Dispatching Work


DEMO
@mallibone

@mallibone

Doing it with style

@mallibone



https://patternlab.io/

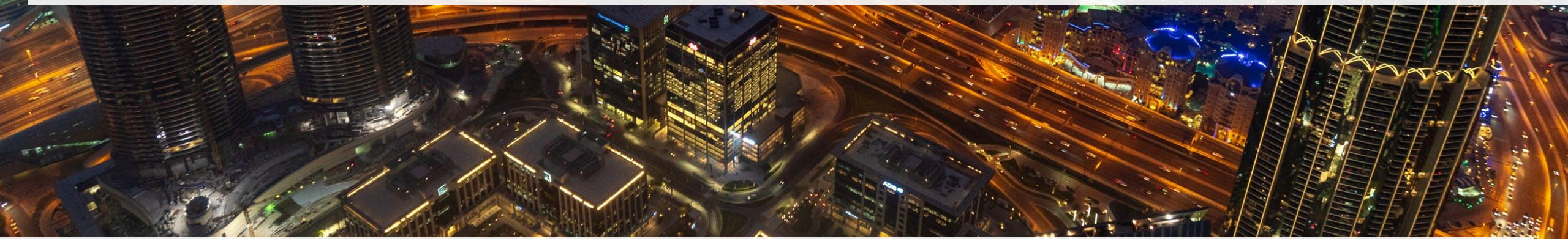
@mallibone


Demo
@mallibone

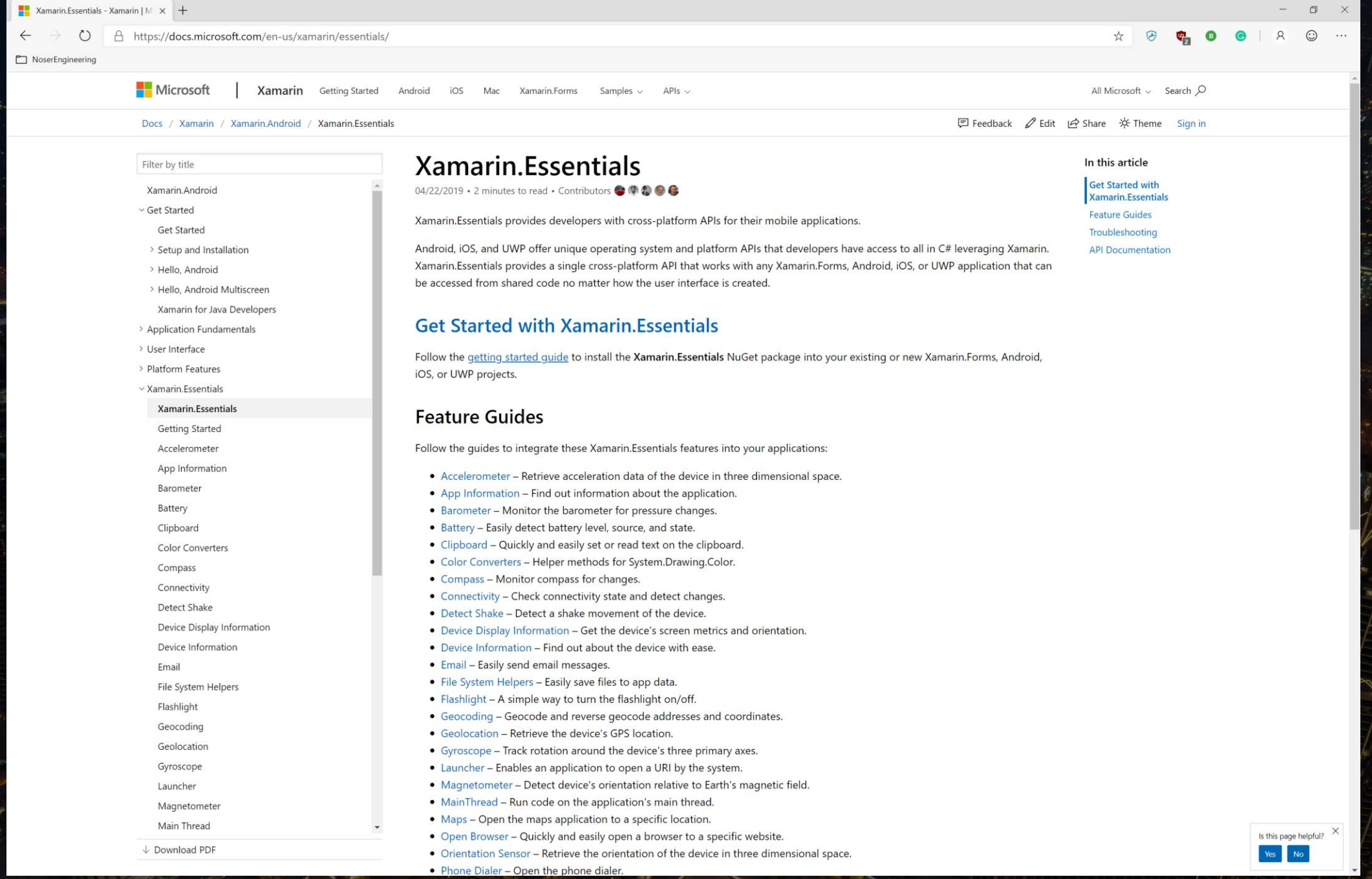
https://fsprojects.github.io/Fabulous/tools.html

@mallibone

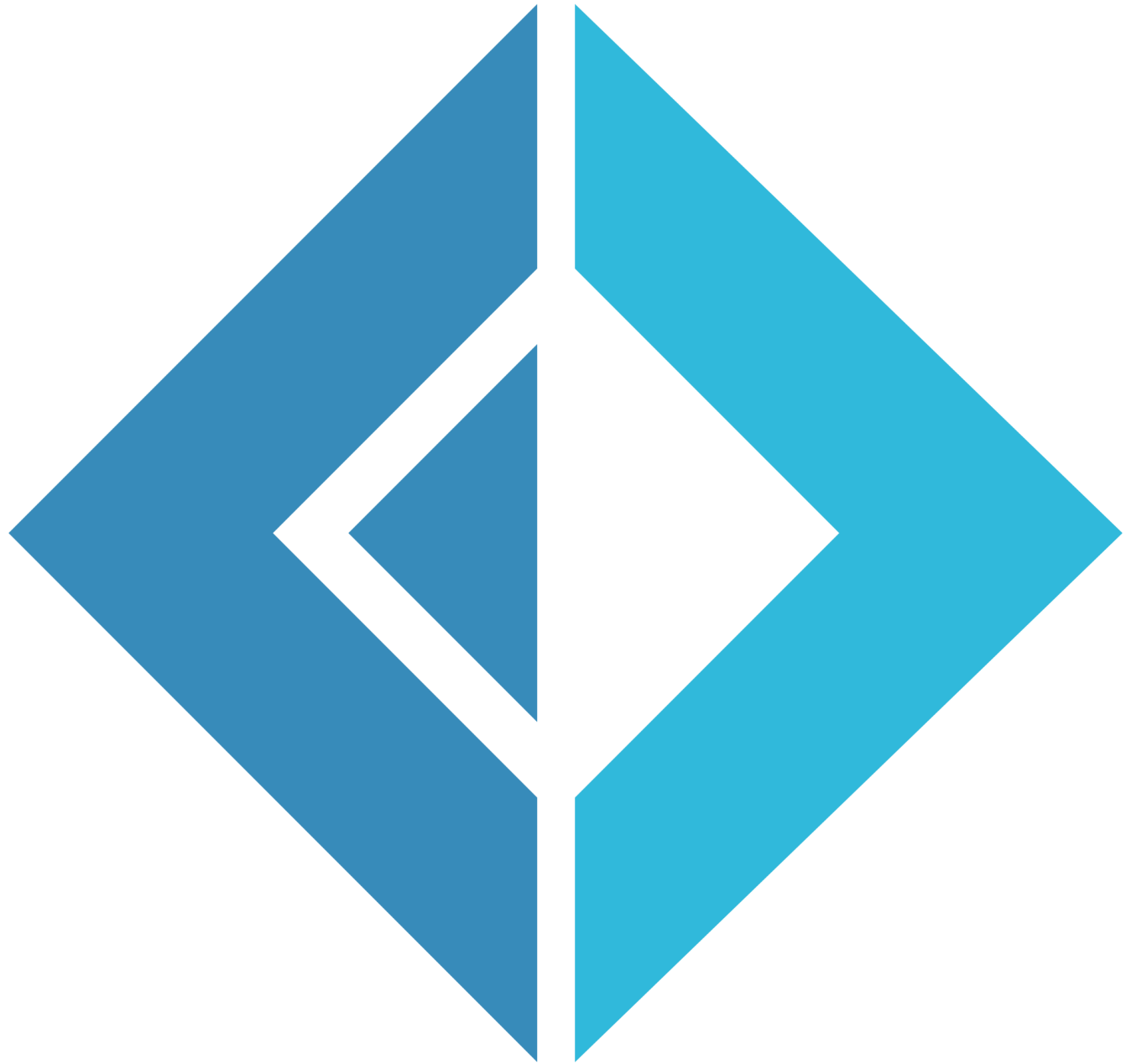
What about Shell? Visual Material? Or your other favorite Xamarin feature?



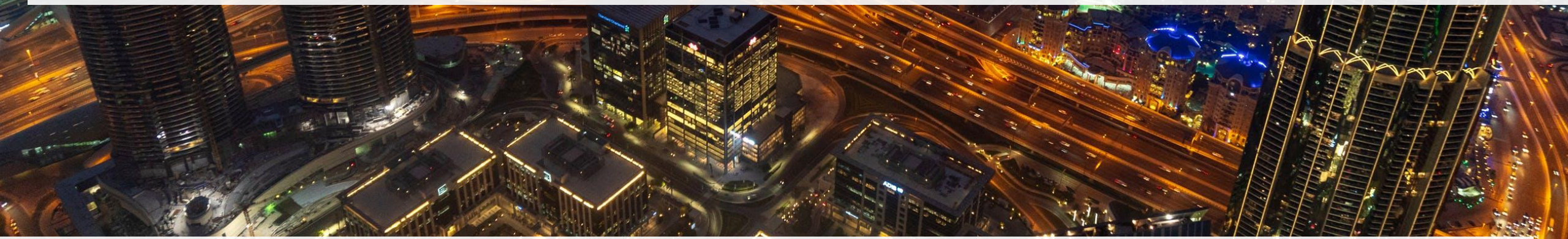
@mallibone

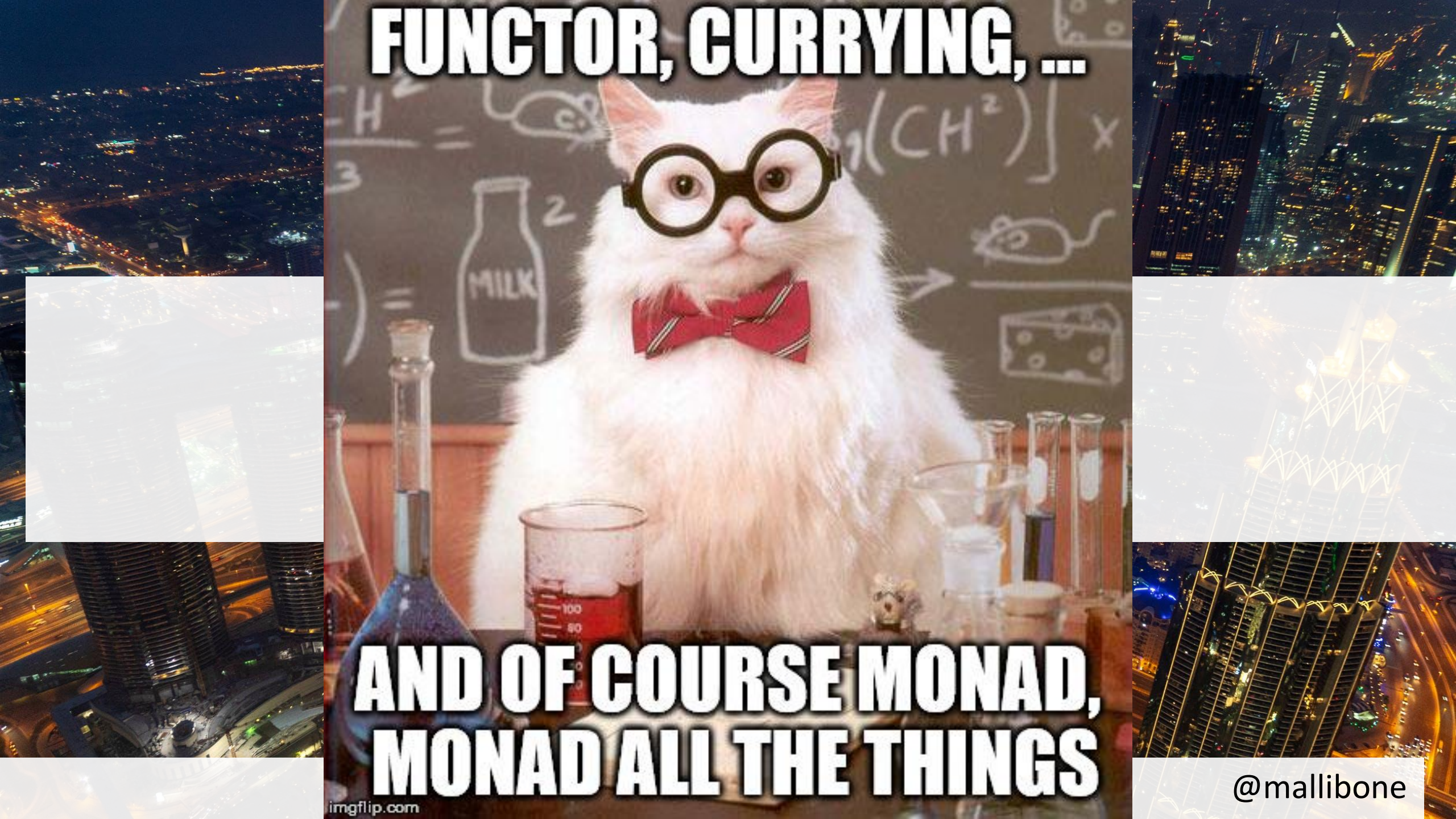

Xamarin.Essentials - Xamarin | M
https://docs.microsoft.com/en-us/xamarin/essentials/
NoserEngineering
Microsoft | Xamarin Getting Started Android iOS Mac Xamarin.Forms Samples APIs
All Microsoft Search
Docs / Xamarin / Xamarin.Android / Xamarin.Essentials
Feedback Edit Share Theme Sign in
Filter by 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
Accelerometer
App Information
Barometer
Battery
Clipboard
Color Converters
Compass
Connectivity
Detect Shake
Device Display Information
Device Information
Email
File System Helpers
Flashlight
Geocoding
Geolocation
Gyroscope
Launcher
Magnetometer
Main Thread
Download PDF
Xamarin.Essentials
04/22/2019 • 2 minutes to read • Contributors
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 Information – Find out information about 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.
Detect Shake – Detect a shake movement of the device.
Device Display Information – Get the device's screen metrics and orientation.
Device Information – Find out about the device with ease.
Email – Easily send email messages.
File System Helpers – Easily save files to app data.
Flashlight – A simple way to turn the flashlight on/off.
Geocoding – Geocode and reverse geocode addresses and coordinates.
Geolocation – Retrieve the device's GPS location.
Gyroscope – Track rotation around the device's three primary axes.
Launcher – Enables an application to open a URI by the system.
Magnetometer – Detect device's orientation relative to Earth's magnetic field.
MainThread – Run code on the application's main thread.
Maps – Open the maps application to a specific location.
Open Browser – Quickly and easily open a browser to a specific website.
Orientation Sensor – Retrieve the orientation of the device in three dimensional space.
Phone Dialer – Open the phone dialer.
In this article
Get Started with Xamarin.Essentials
Feature Guides
Troubleshooting
API Documentation
Is this page helpful?
Yes
No

@mallibone



How to get started with F#?

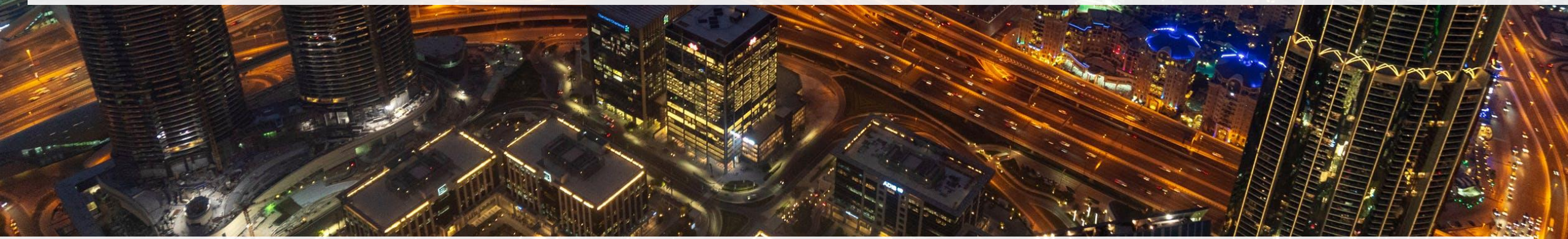



FUNCTOR, CURRYING, ...
MILK
AND OF COURSE MONAD,
MONAD ALL THE THINGS
imgflip.com

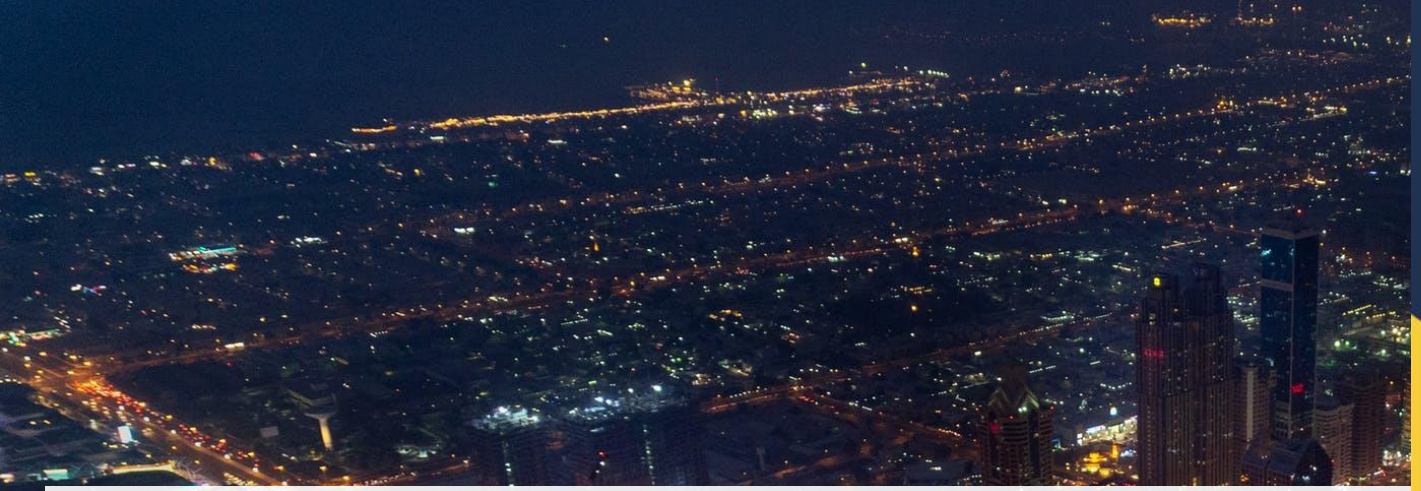
@mallibone


"HOLY FORKING SHIRT!"
#THEGOODPLACE
NBC

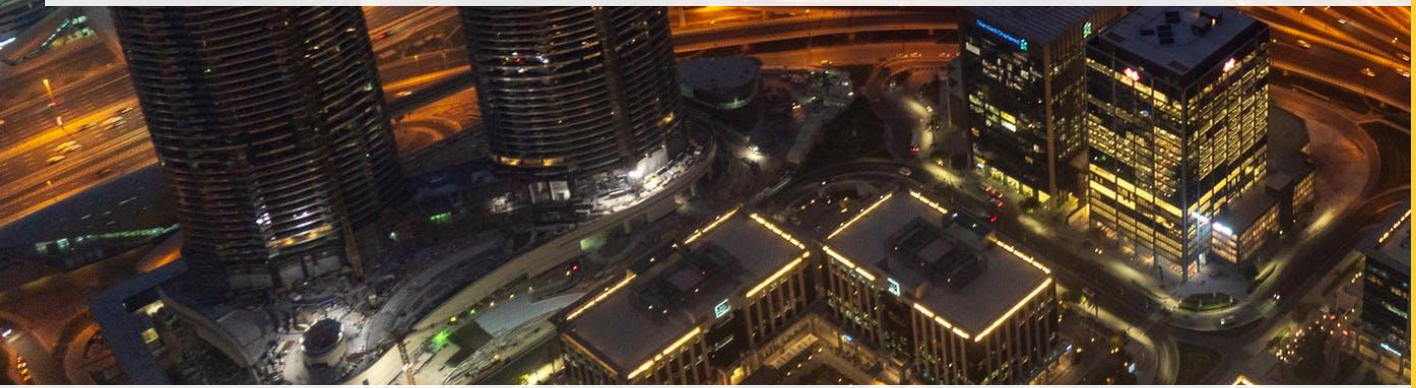
Learn Design patterns after learning the language

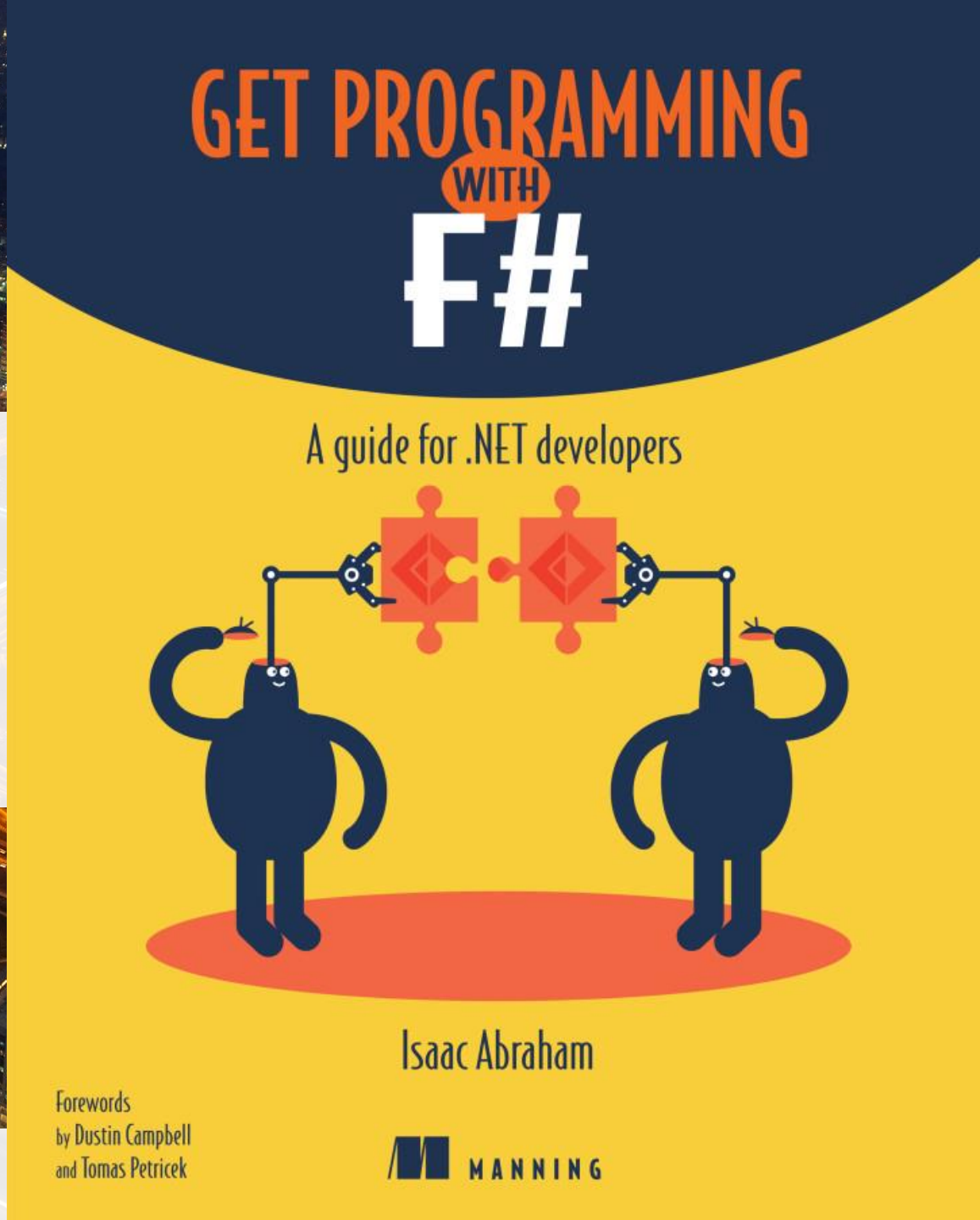


@mallibone

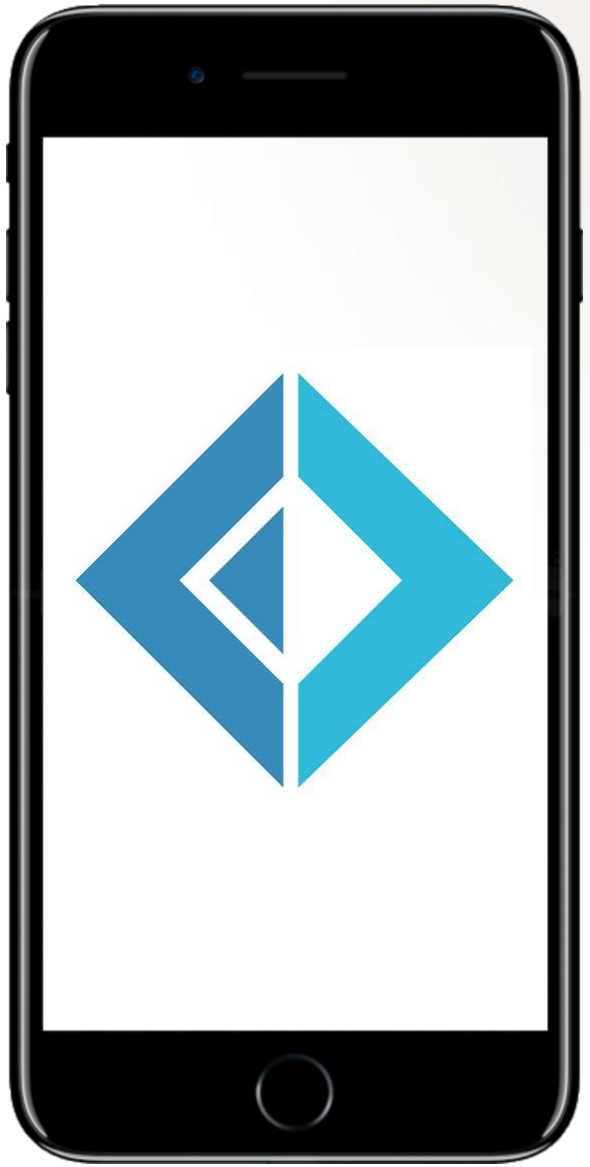


How to get started with F#?



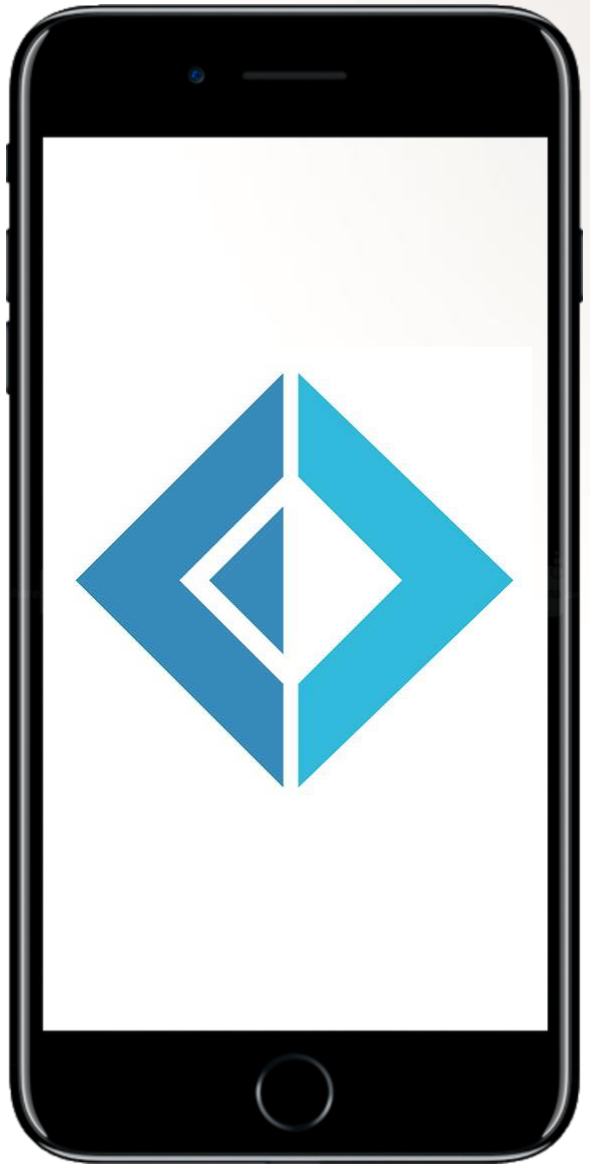


Takeaways



Takeaways

- F# is not scary it’s just yet another .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

Head of Mobile, Noser Engineering AG

https://fsprojects.github.io/Fabulous

https://fsharpforfunandprofit.com/

https://mallibone.com

PLANET X FEATURED COMMUNITY BLOG

MVP Microsoft® Most Valuable Professional

Microsoft CERTIFIED Solution Developer

Xamarin Certified Mobile Developer