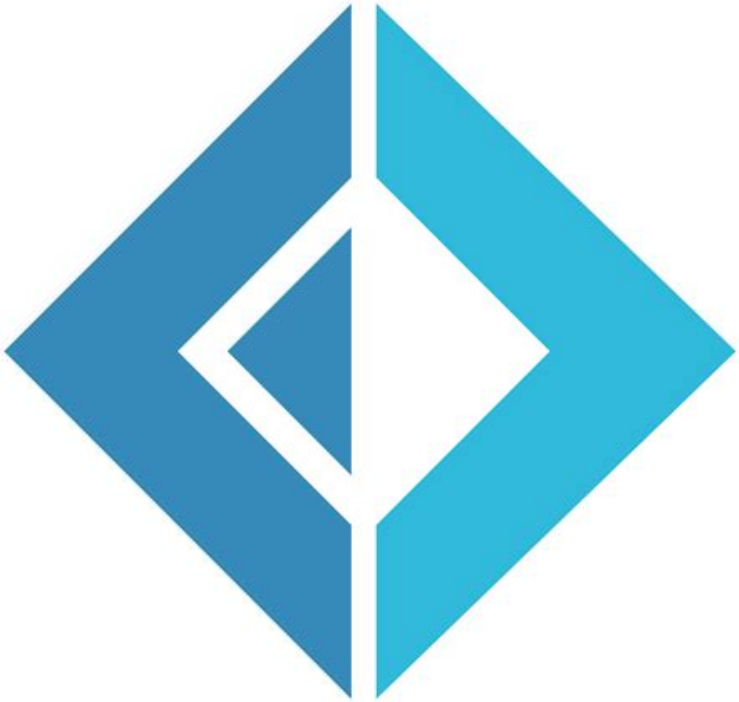
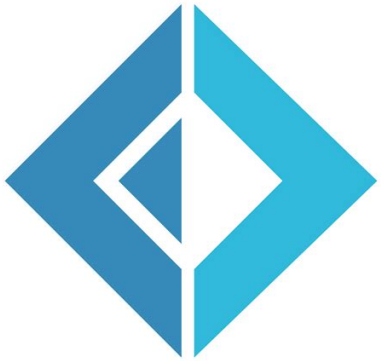

.NET
Frontend Day
Fabulous Functional Frontends
Mark Allibone
Mobile Lead
Rey Technology
#dotNETfrontend
@mallibone



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



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

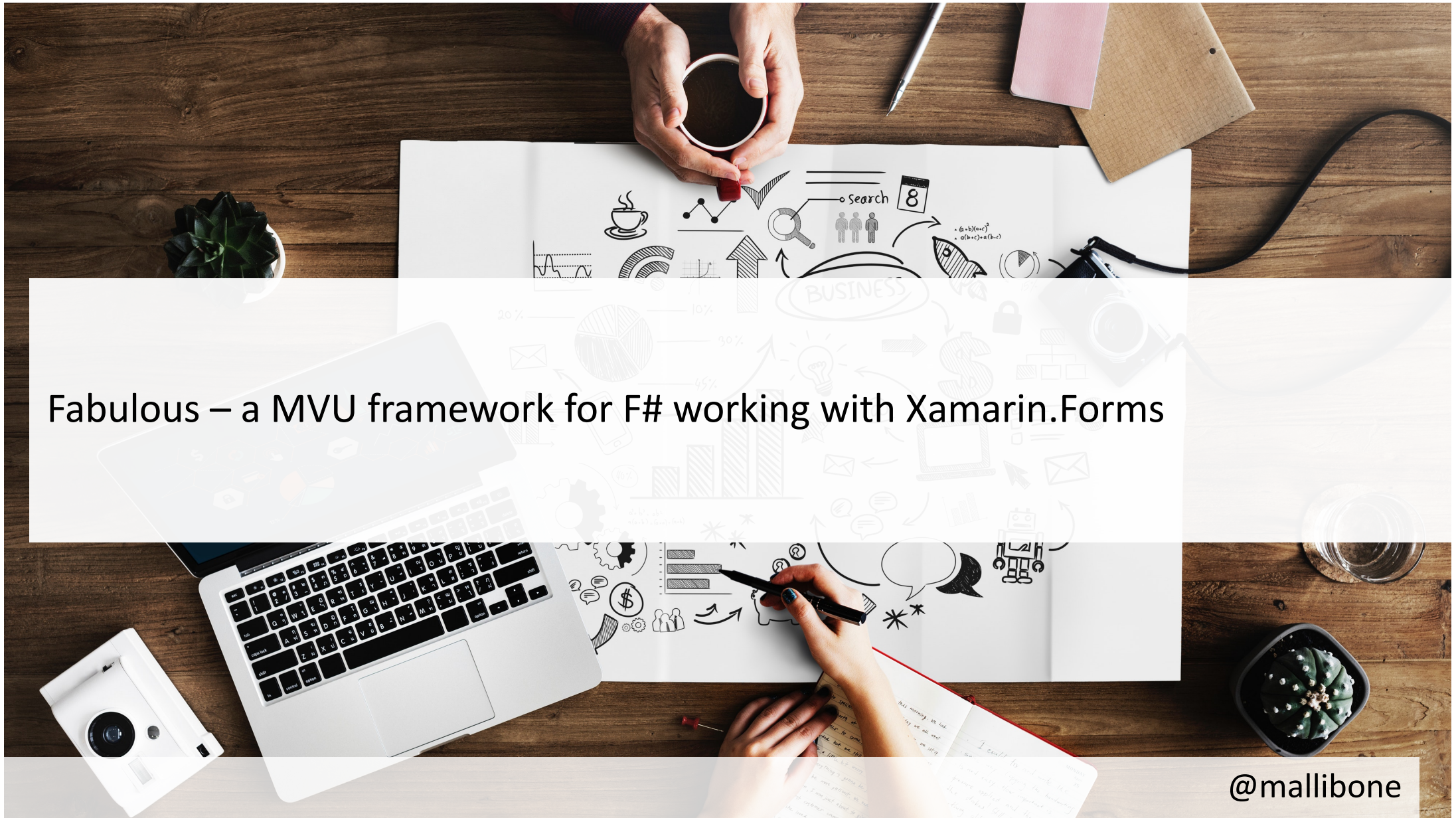
@mallibone

Why not just use Xamarin Forms?!

@mallibone

F# can do OO programming but it is functional first

@mallibone



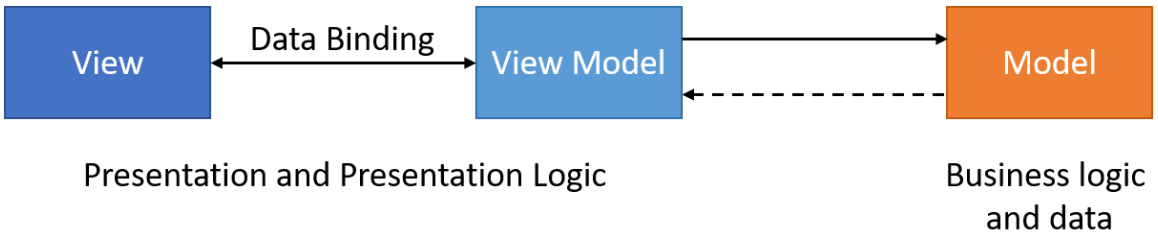
Fabulous – a MVU framework for F# working with Xamarin.Forms

@mallibone

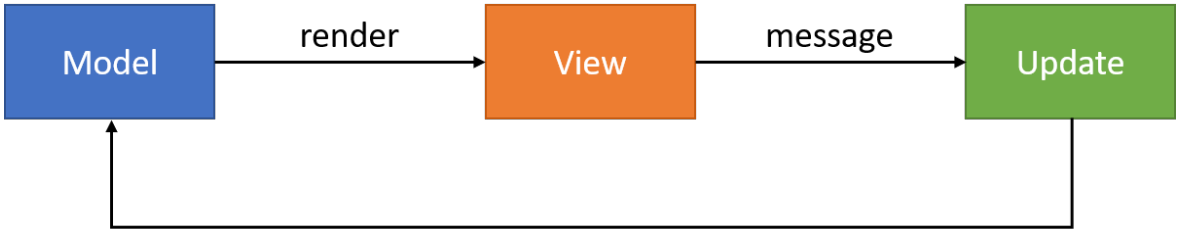
MVU the functional MV* Pattern

@mallibone

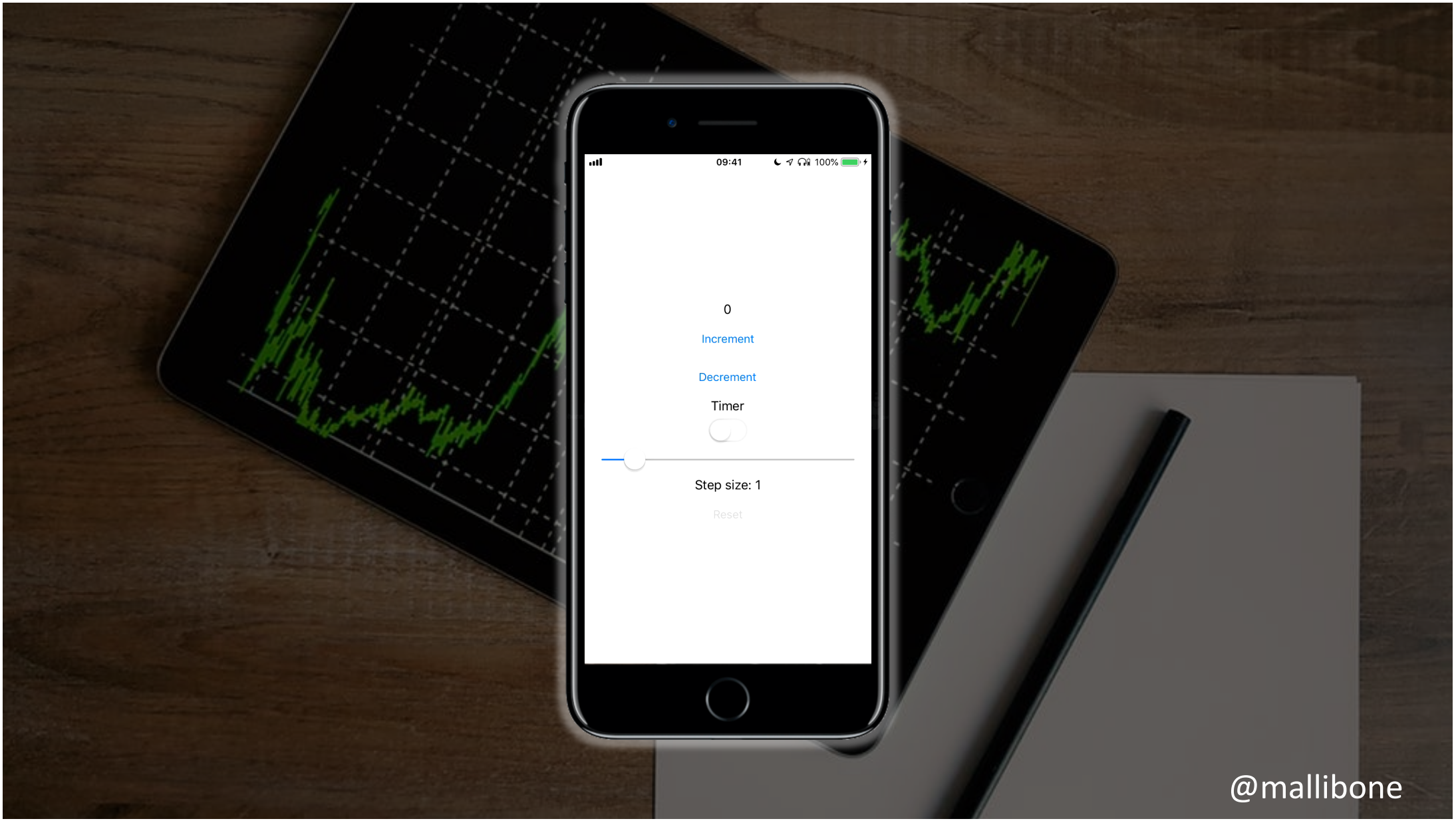
Model View View Model

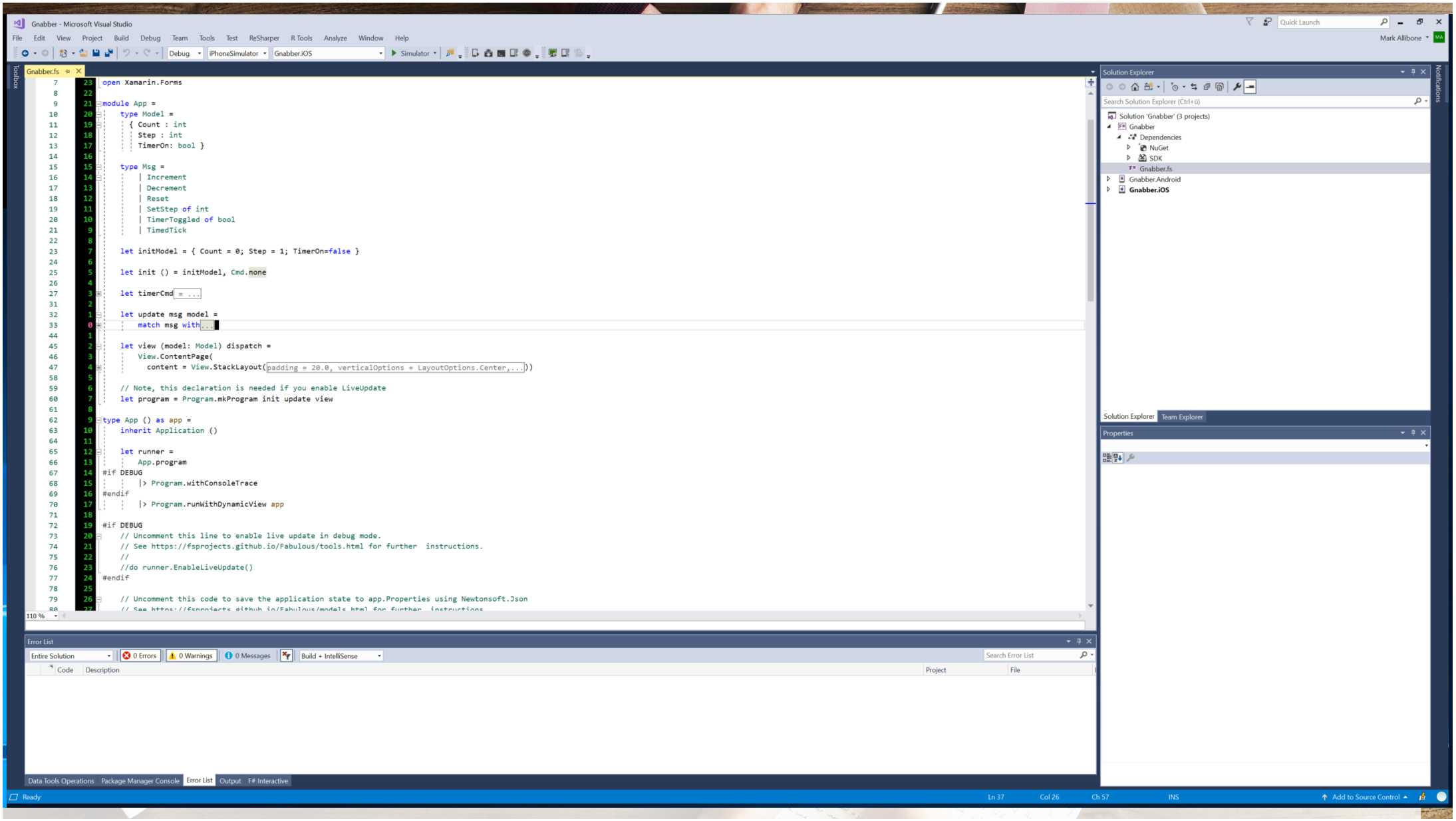


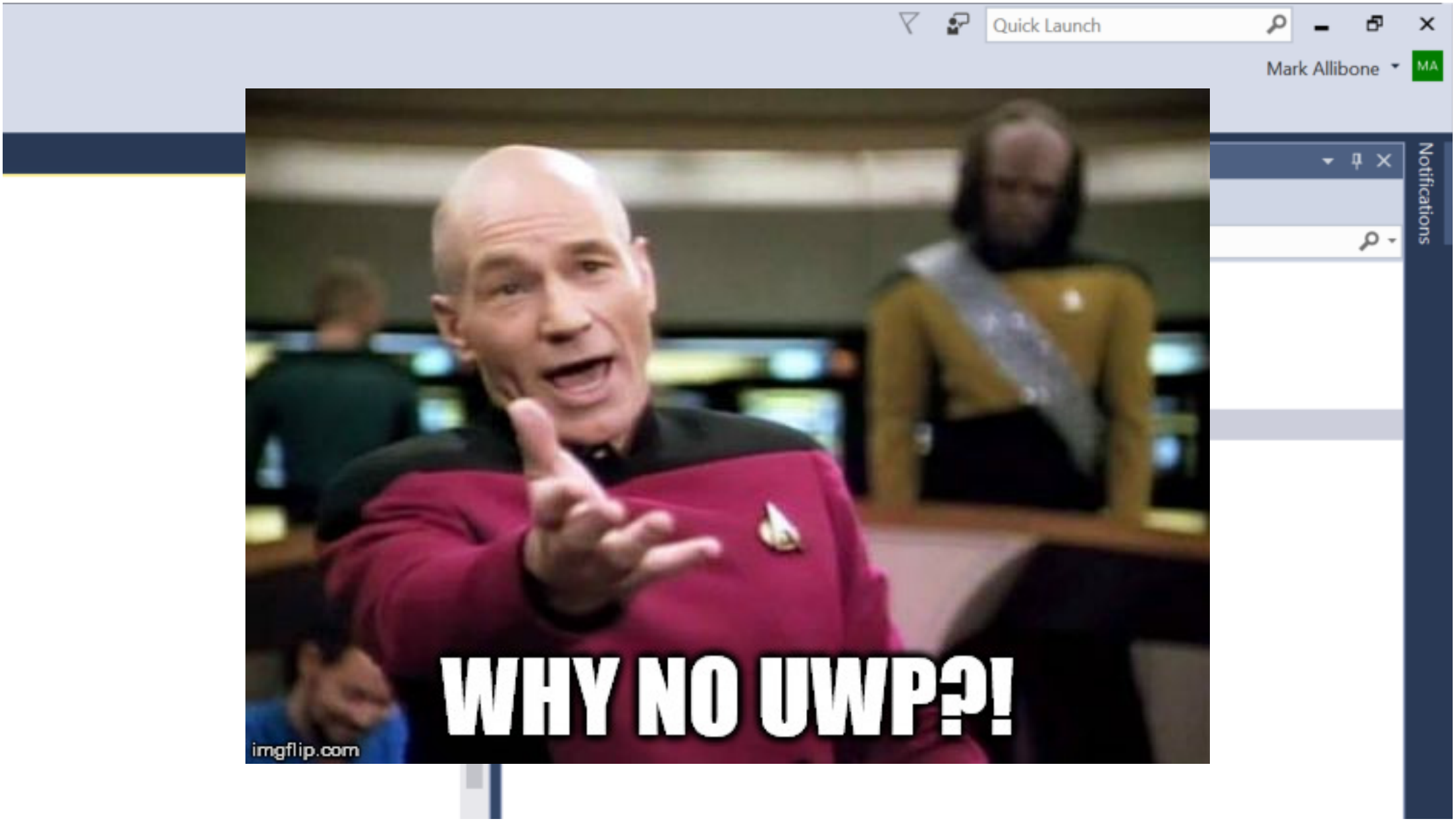
Model View Update

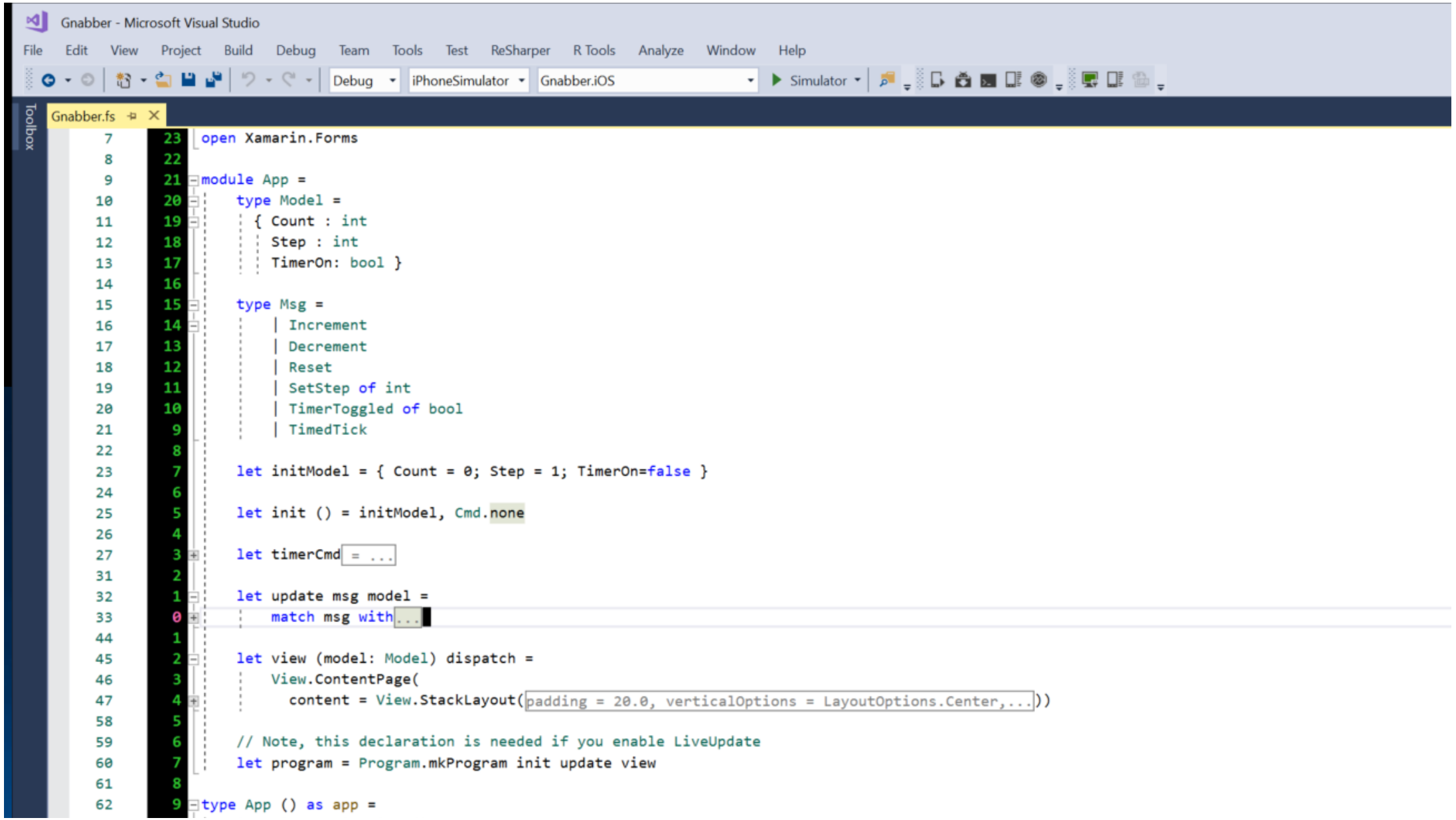


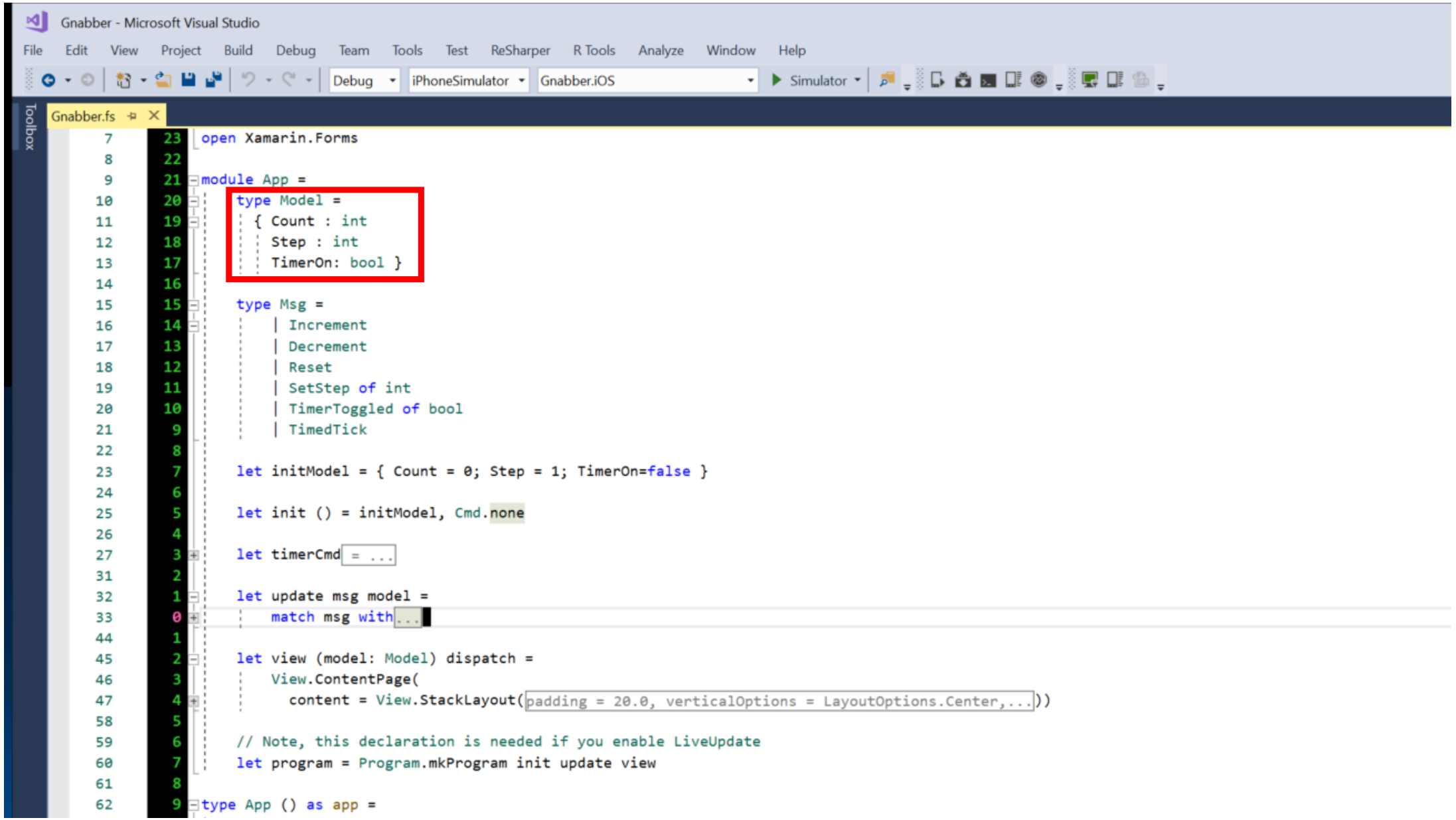
Functional Frontends

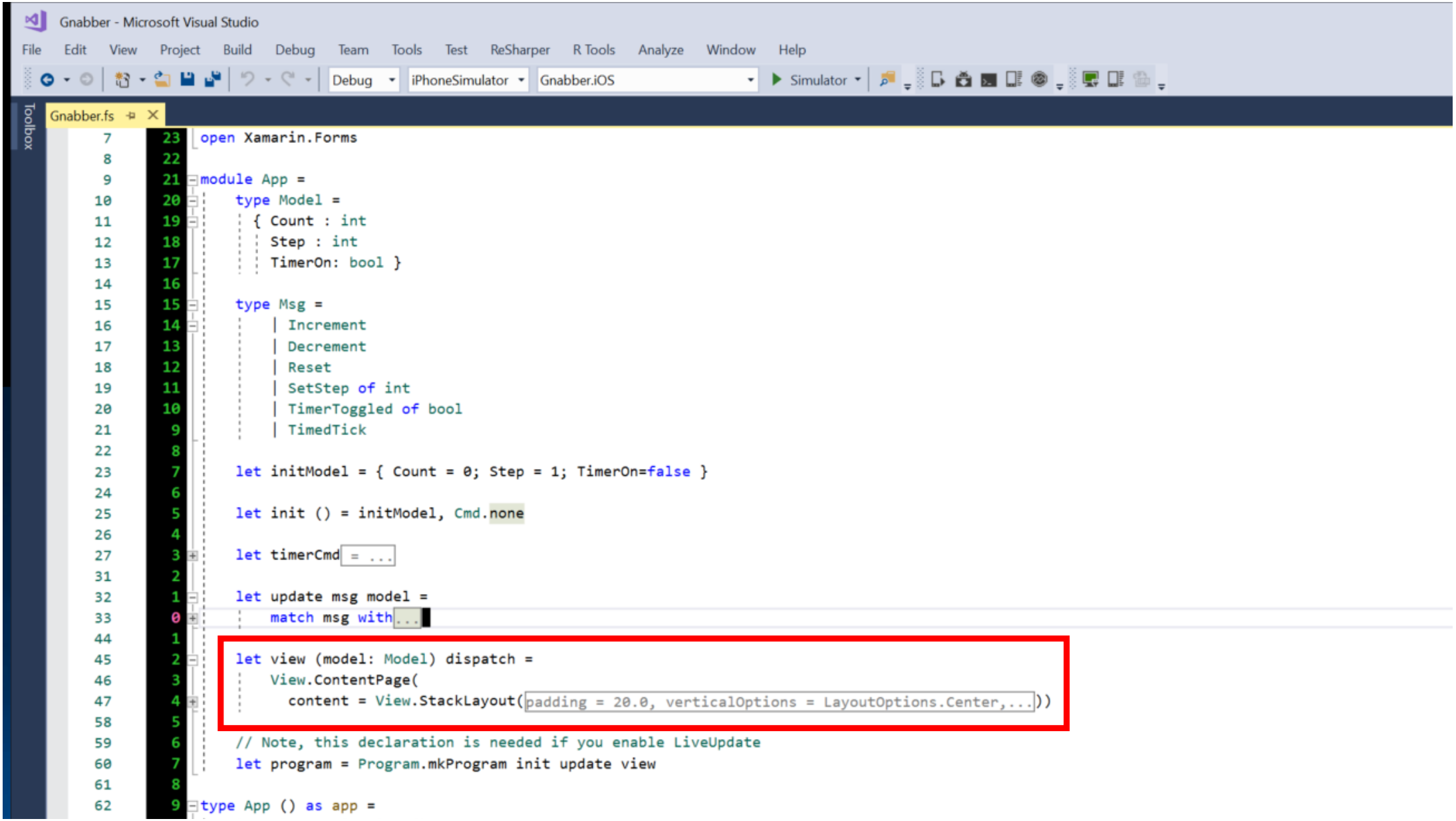

09:41
100%
0
Increment
Decrement
Timer
Step size: 1
Reset
@mallibone

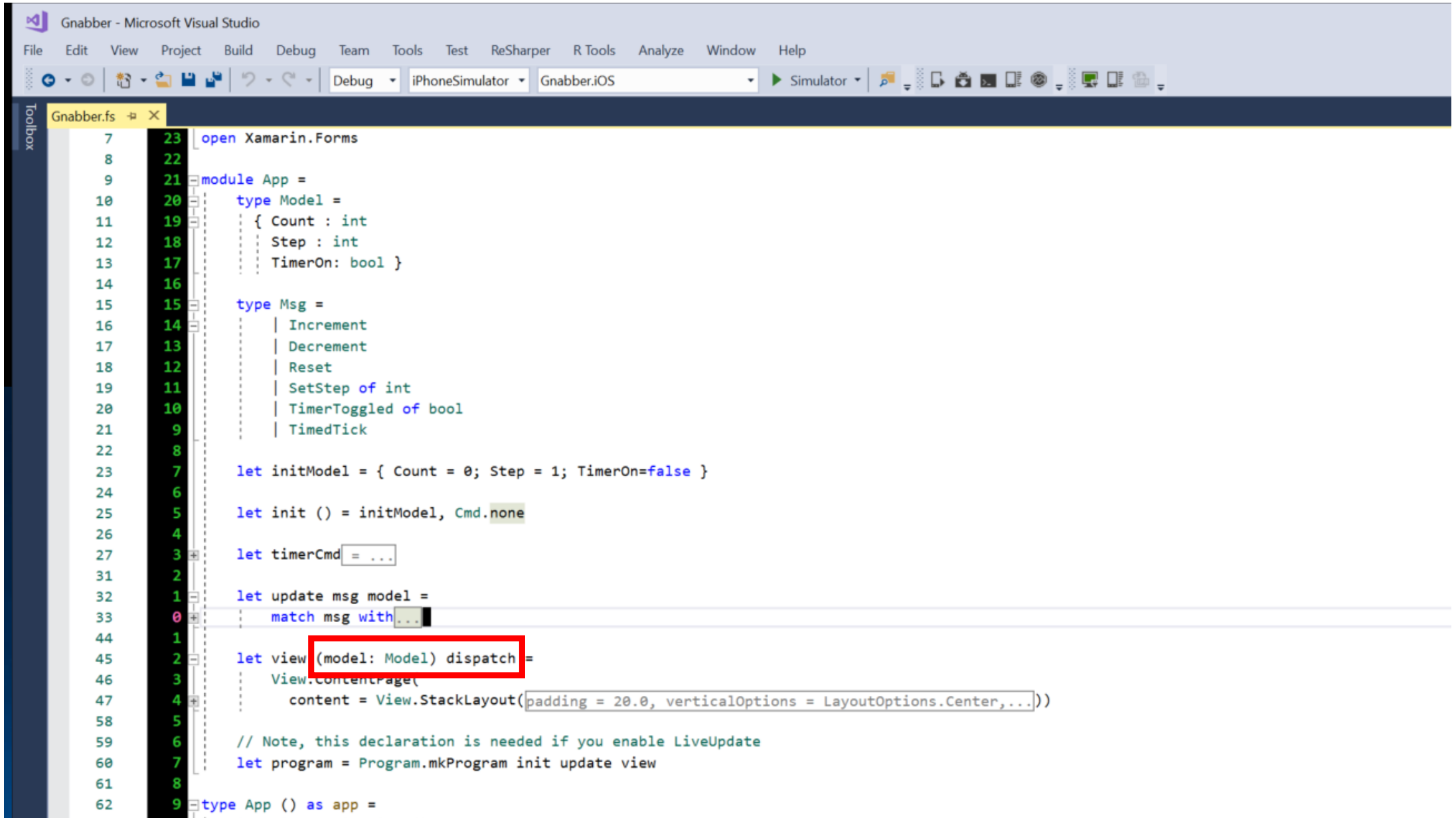

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Mark Allibone
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment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

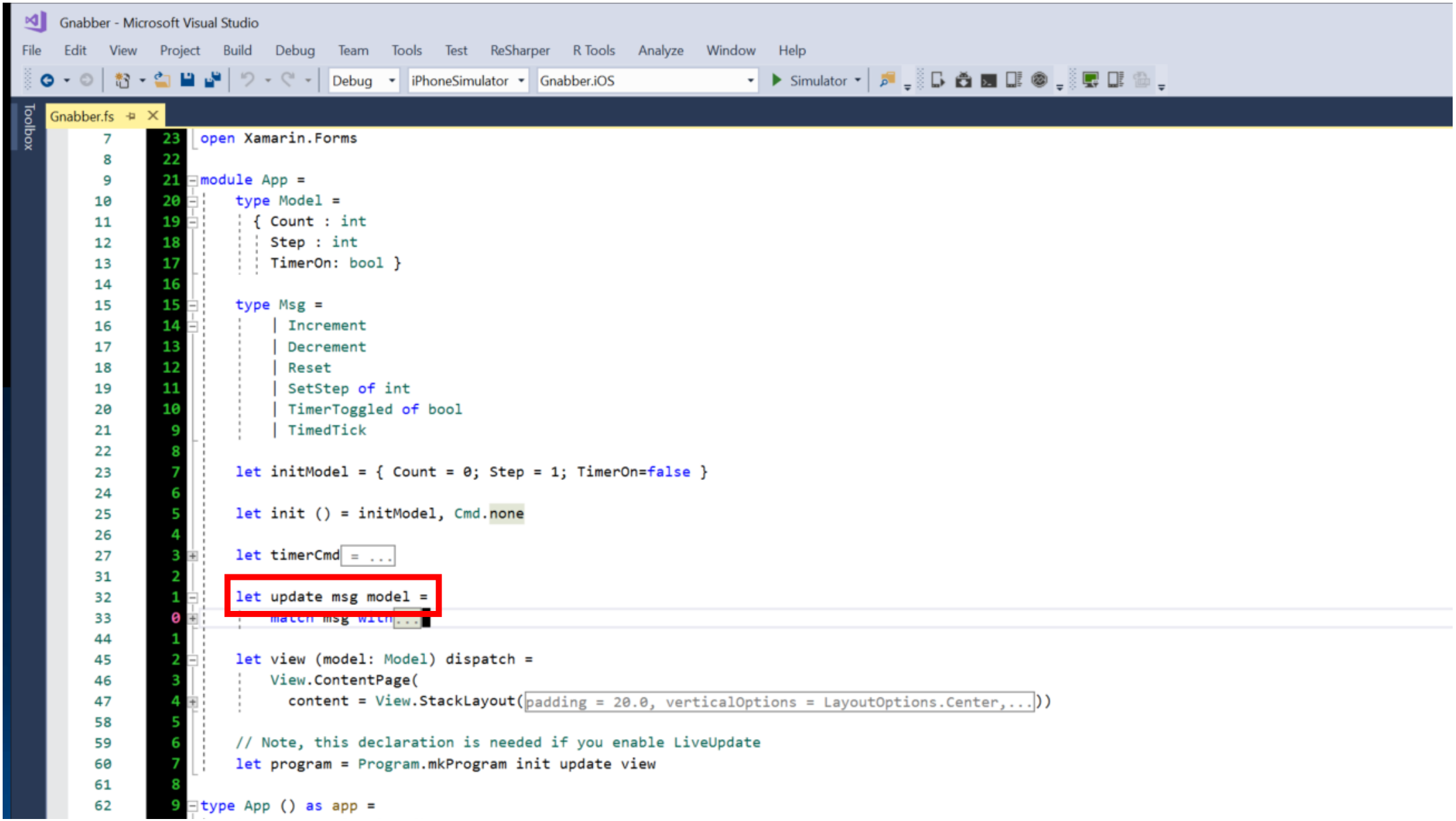

Quick Launch
Mark Allibone
MA
Notifications
WHY NO UWP?!
imgflip.com

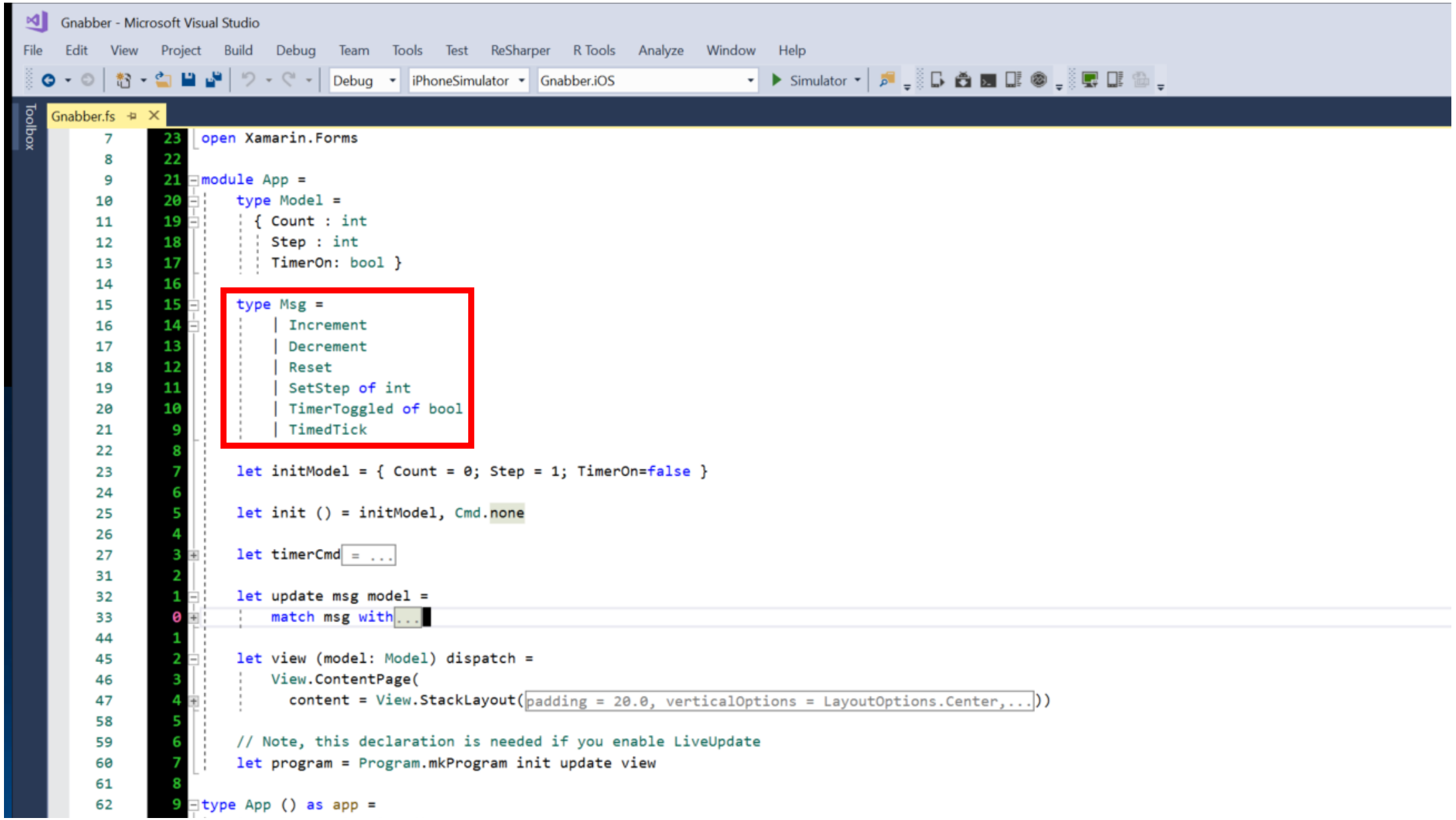

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug
iPhoneSimulator
Gnabber.iOS
Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement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

- Driven from the model not the view
- State changes are defined with messages
- Single place for change - the update method
- Uses Xamarin Forms

@mallibone

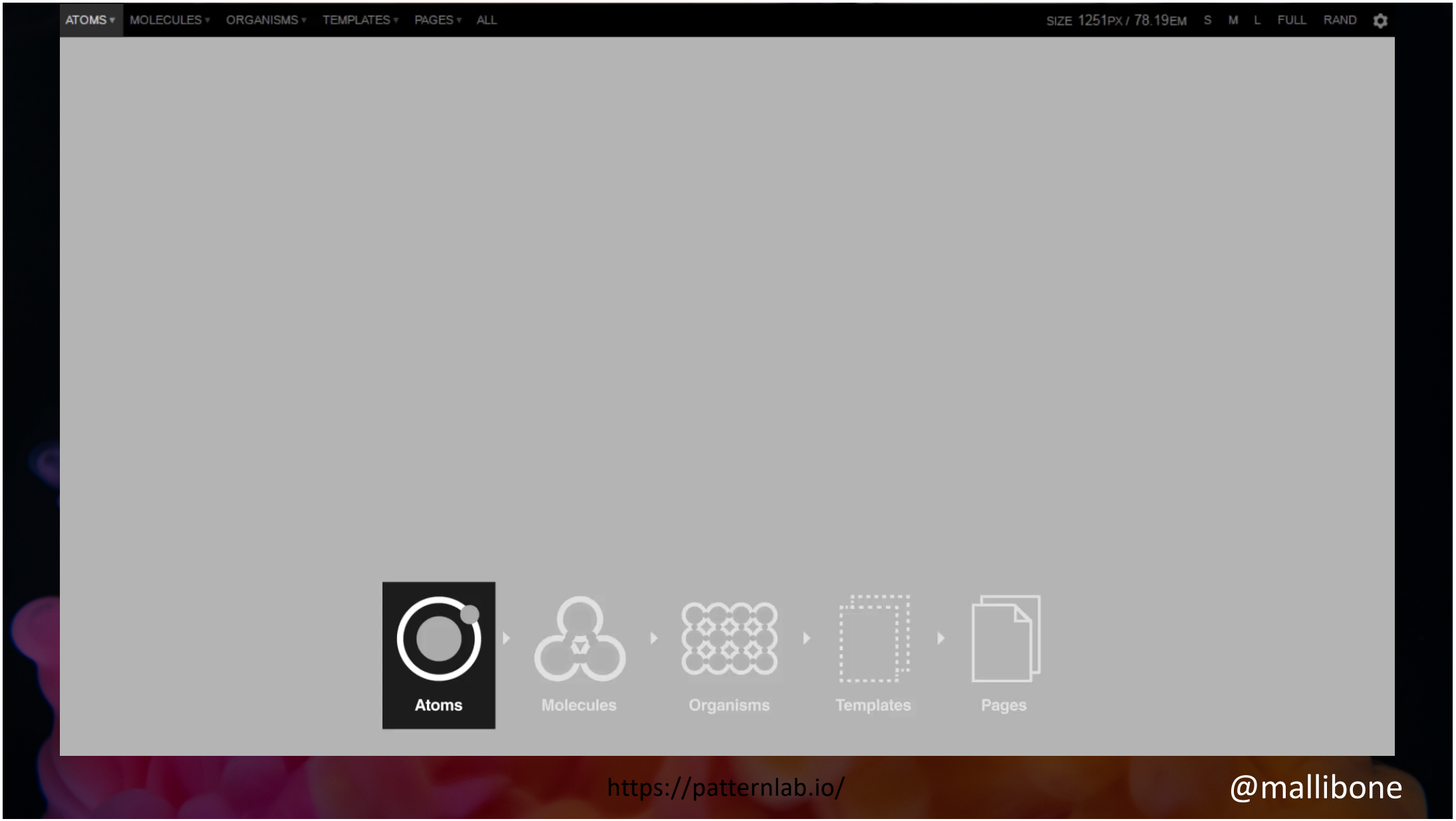
Doing it with style

@mallibone

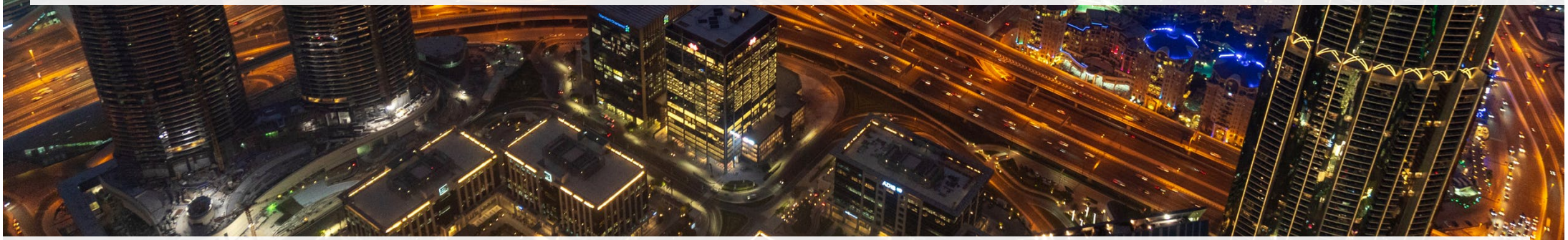

XAML vs Coded UI


Demo
@mallibone

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

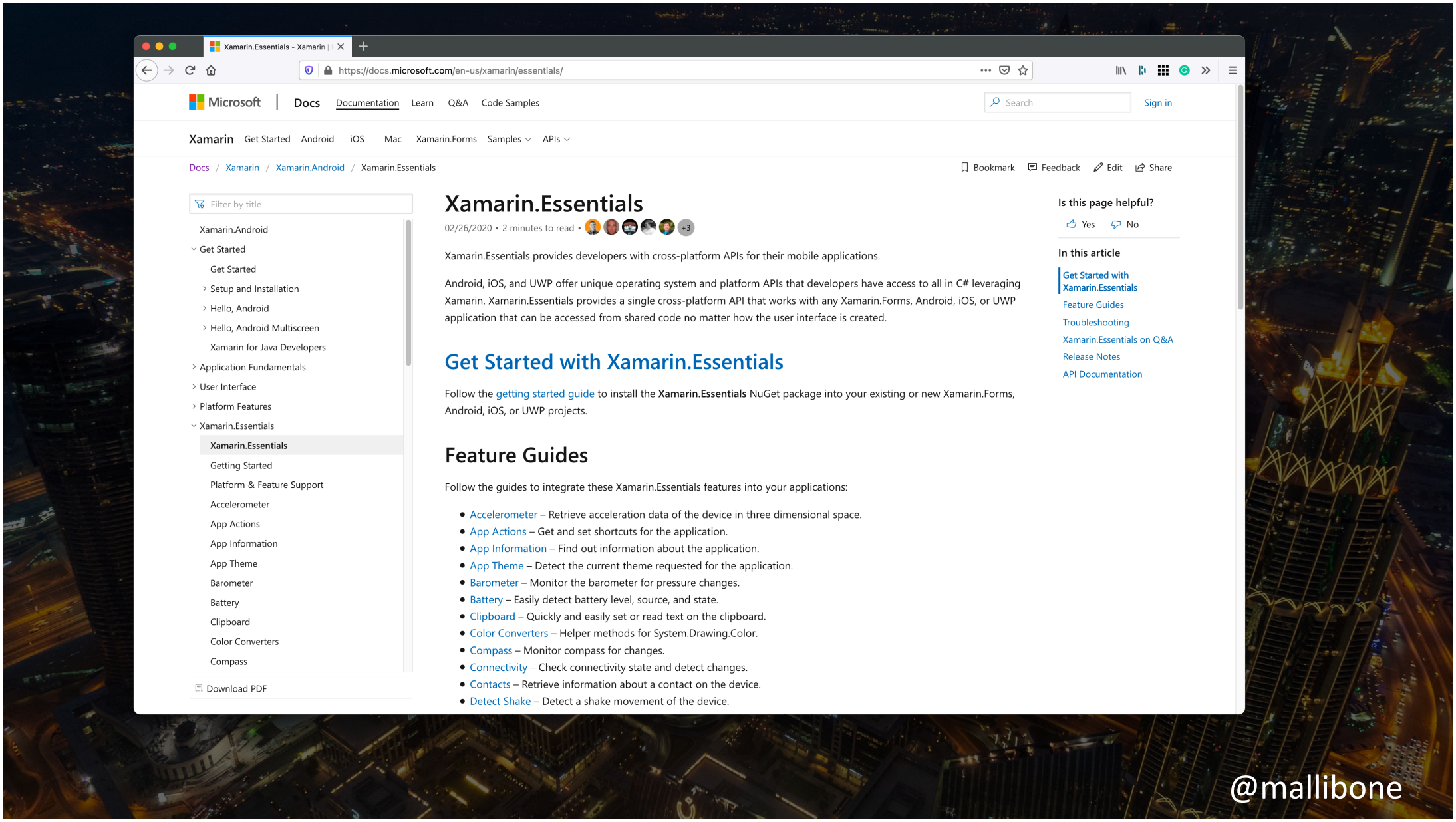

ATOMS
MOLECULES
ORGANISMS
TEMPLATES
PAGES
ALL
SIZE 1251PX / 78.19EM
S
M
L
FULL
RAND
Atoms
Molecules
Organisms
Templates
Pages
https://patternlab.io/
@mallibone

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

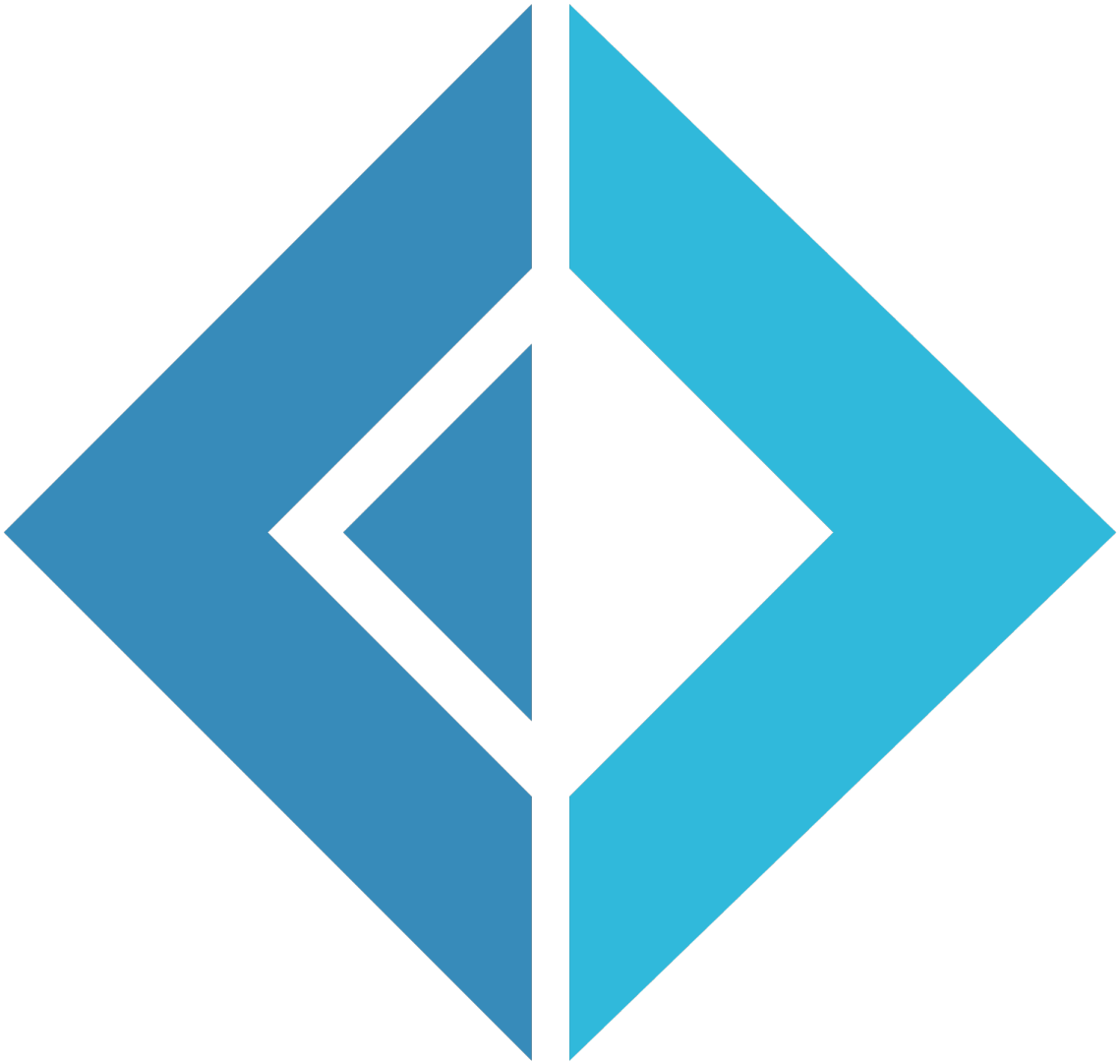


@mallibone

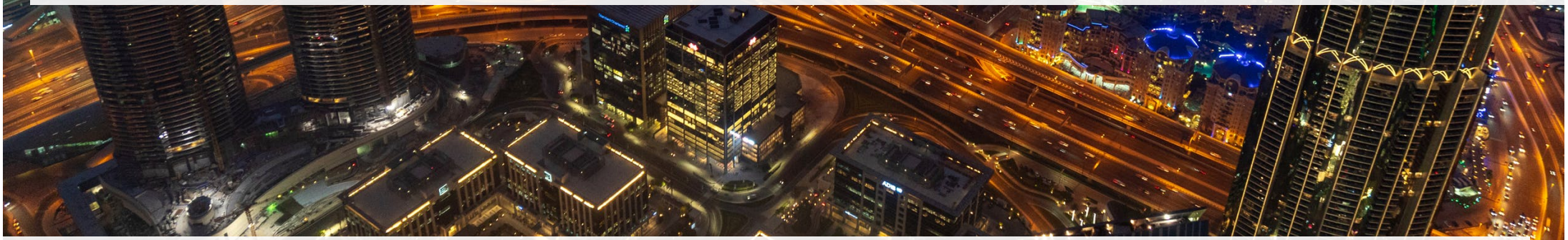

NuGet
@mallibone


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 •
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

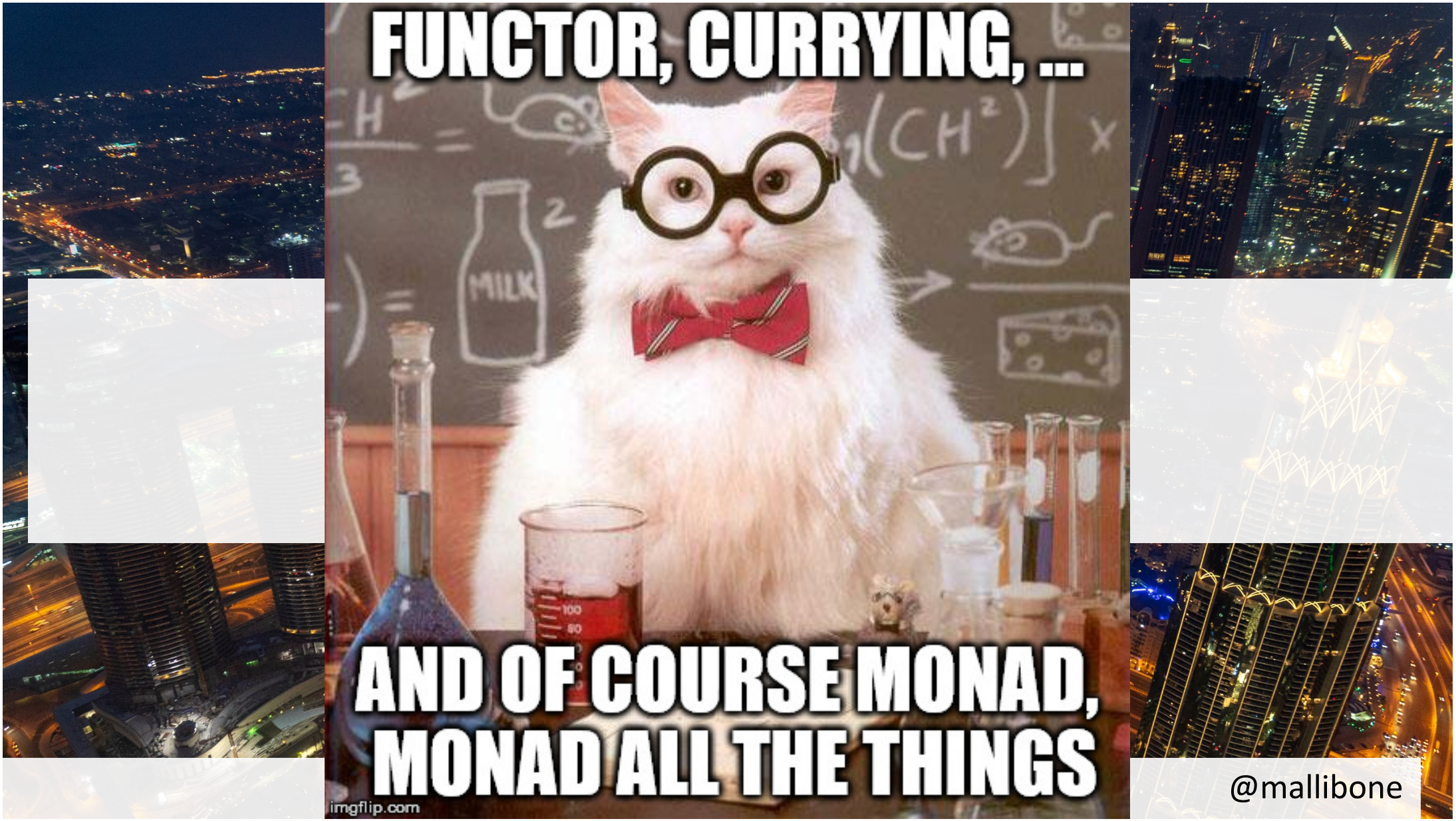
@mallibone

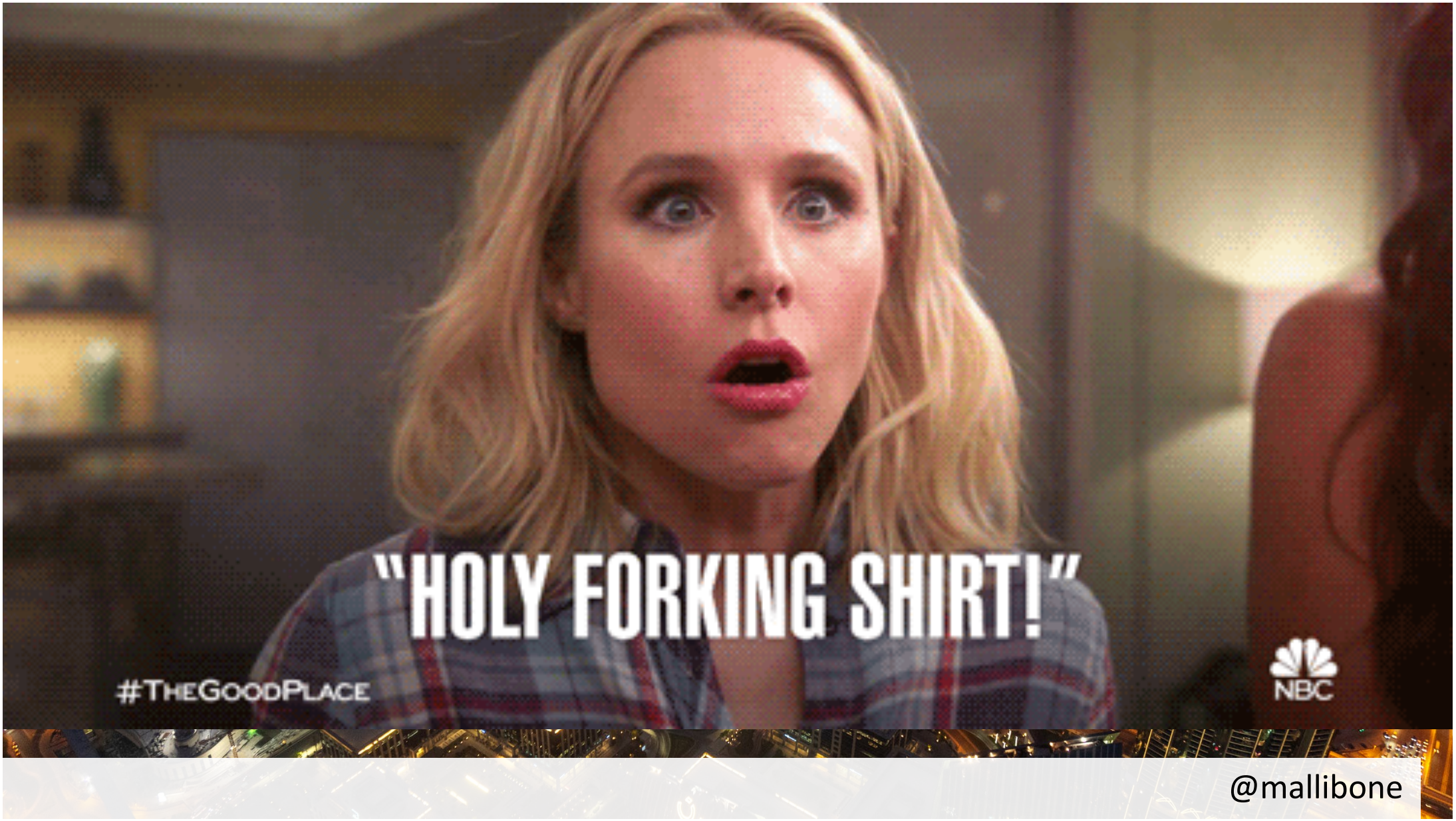


How to get started with F#?

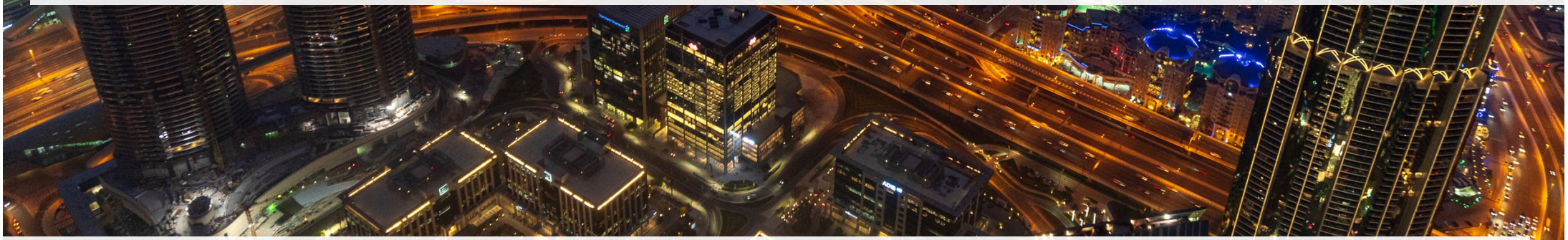


@mallibone

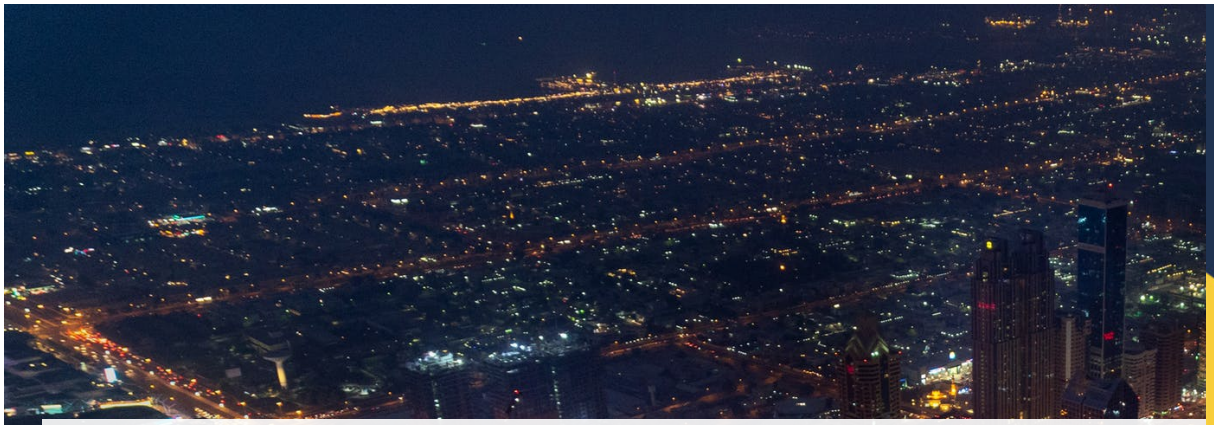

FUNCTOR, CURRYING, ...
MILK
AND OF COURSE MONAD,
MONAD ALL THE THINGS
imgflip.com
@mallibone


"HOLY FORKING SHIRT!"
#TheGoodPlace
NBC

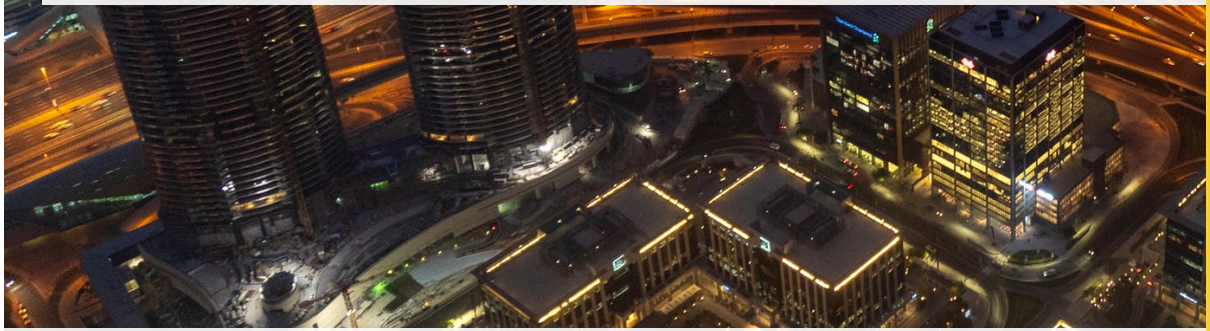
Learn Design patterns after learning the language

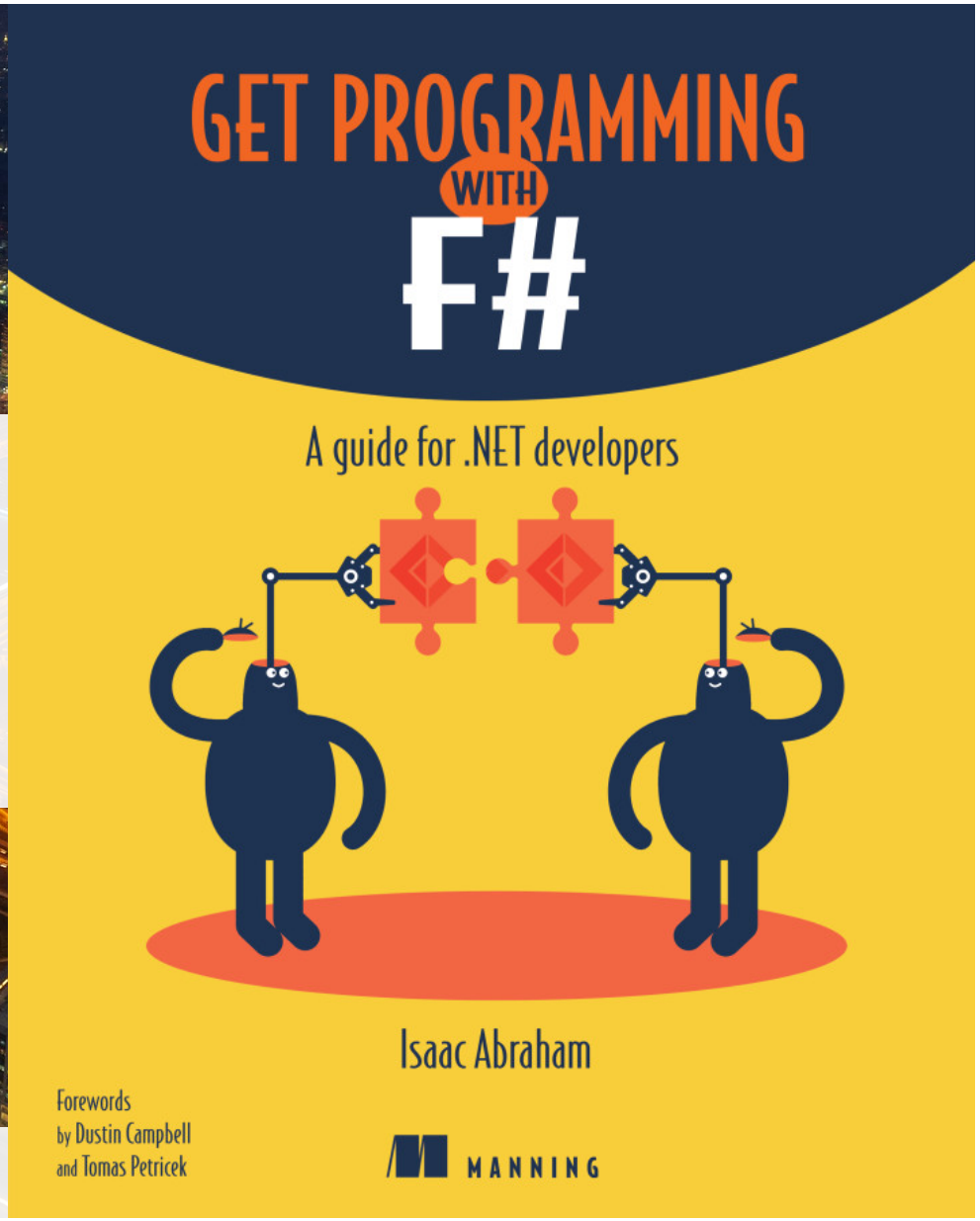


@mallibone



How to get started with F#?





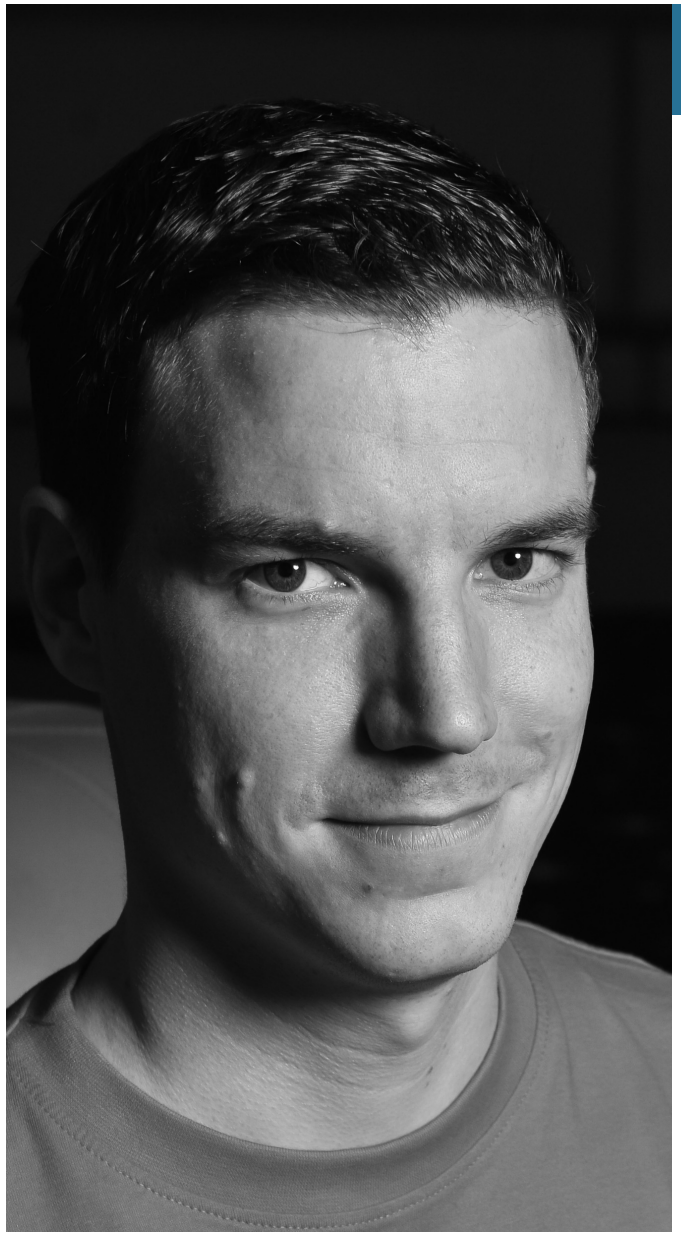
REY
TECHNOLOGY

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

Microsoft® MVP Most Valuable Professional