2018 EDITION

Starting out right with Xamarin

and other useful tips, tricks and resources
p.s. still just my opinion

RYAN DAVIS

QUEENSLAND C# MOBILE DEVELOPERS MEETUP

2018 09 25

whoami

- Ryan Davis
- Professional ~~Mobile~~ LINQPad Developer

- Blog: ryandavis.io
- Twitter: @rdavis_au
- Github: @rdavisau
  - essential-interfaces – use DI/mocking with Xamarin.Essentials
  - jsondatacontext-linqpad – json data context driver for LINQPad
  - sockets-for-pcl, sockethelpers – socket comms in a PCL
    (today you should use netstandard sockets why are you all still installing this)

to cover

- ~~File -> New~~
- Mobile principles to live by
- Pop Quiz – your first big questions
- Spotlight on some useful libraries
- Roundup of resources.

-= mobile principles to live by =-

#1

know platform

~~love~~ thy ~~neighbour~~

why Xamarin, again?

(yeah this list isn't really MECE)

Xamarin because:

- ✓ Write code in powerful, expressive languages
- ✓ Take advantage of the .NET platform and ecosystem
- ✓ TARGET ALL THE PLATFORMS!!
- ✓ Share as much code as we can
- ✓ Avoid Java and Objective-C and swift

- ✘ ~~Don't want to learn about the native platforms.~~

not all mobile platforms are created equal

Xamarin is not designed to relieve you of the need to understand the underlying platform:

- Android, iOS and UWP differ on many levels – UI, app lifecycle, sandboxing

Each platform might be more or less suited to a task or paradigm

Xamarin.Forms abstracts some of these platform variations, some of the time.

fortunately, we can make use of native docs

Xamarin's iOS/Android frameworks are C# idiomatic, but otherwise near identical to native ObjC/Swift/Java counterparts

- A book or course targeting iOS or Android native is still relevant, especially for frameworks like UIKit and CoreAnimation
- Tutorials and documentation are easily adapted
- The balance of swift – objc samples has tipped over the last couple of years

Probably the biggest deviation from native API naming:

NATIVE iOS		XAMARIN iOS
- tableView:cellForRowAtIndexPath:	→	GetCell(,)

-= mobile principles to live by =-

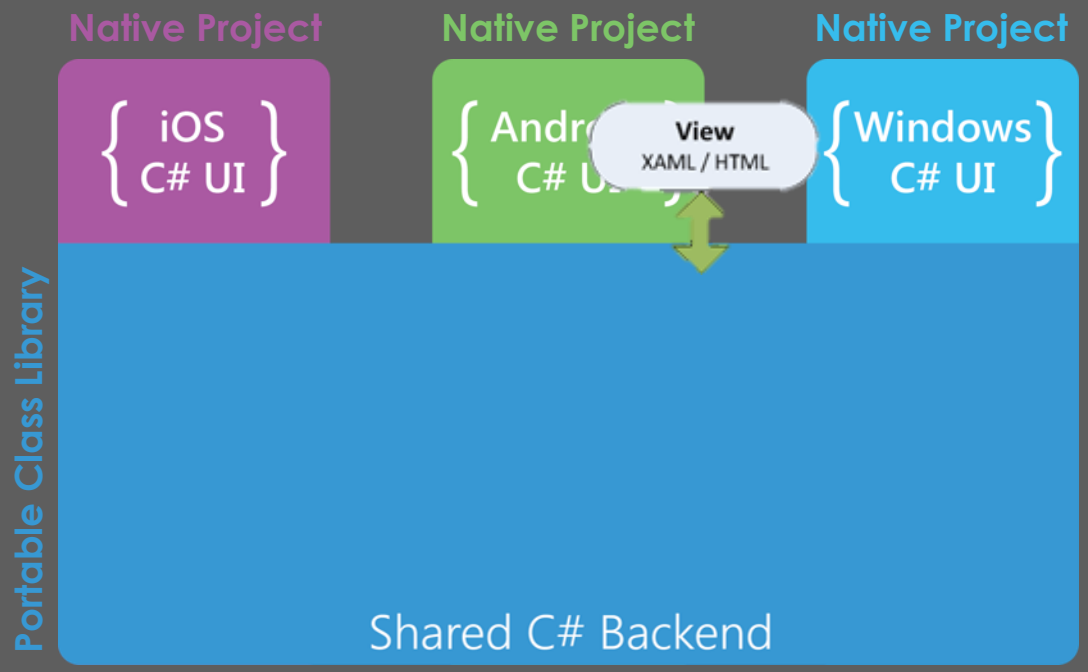
#2

cross-platform

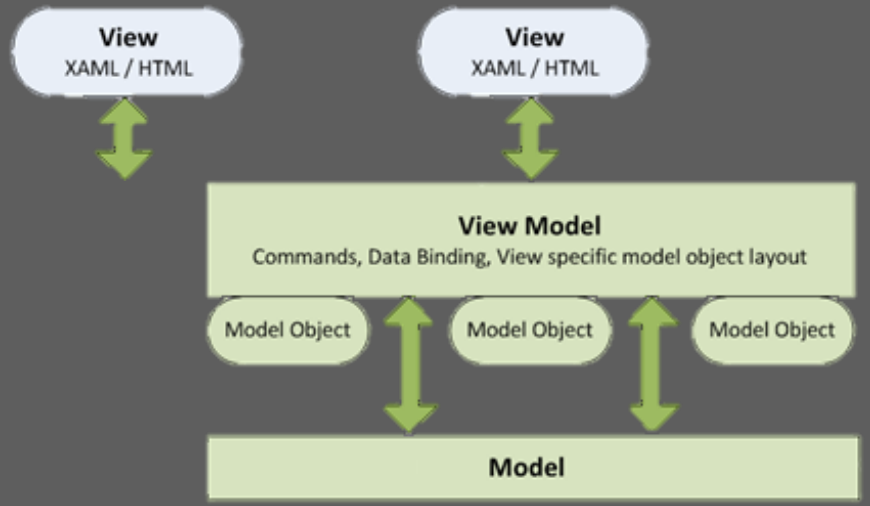
begin with the^end in mind

Forms or otherwise, MVVM is a safe bet

The components of MVVM map closely to the original code sharing architecture in a Xamarin app, and the ecosystem has evolved around that.



From a Xamarin Presentation



From a google image result for MVVM

go MVVM with netstandard from the beginning

(even if you focus on a single platform first)

Keeping your viewmodels in the core project and your views in native projects* encourages Good Habits™

- You cannot accidentally use platform-specific and non-netstandard APIs
- You cleanly separate your viewmodel and views, and your code goes into the viewmodel by default

* In Xamarin.Forms projects, a dedicated UI project gives you the same separation and benefits

-= mobile principles to live by =-

#3

dependencies

keep your ~~liabilities~~ closer

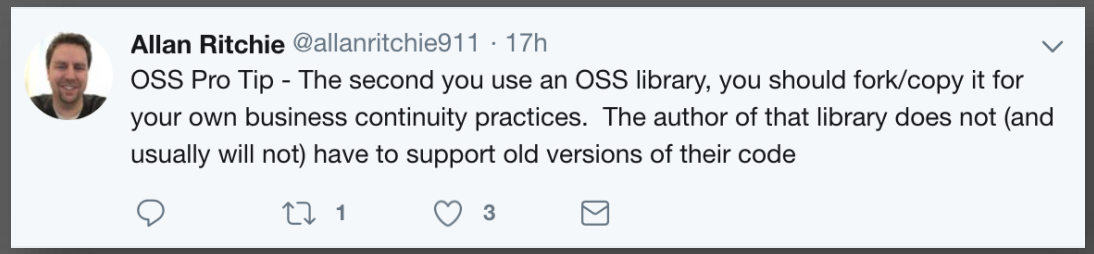
mobile moves fast, libraries age faster

- Xamarin lends itself to libraries and abstractions, and there are many
- Libraries can got stale for lots of reasons, some particular to the Xamarin/Mobile
- Mobile itself moves quickly and changes frequently – something that works today may be deprecated or disallowed in a year's time.

think carefully about adding dependencies

- When the next android SDK deprecates features in your dependencies, or you want to bump your target version, what will you do if your libraries haven't been updated?
- If your dependencies move to newer API versions before you are ready, how can you incorporate important bugfixes or new features?
- If you discover a bug in an abandoned library that you're already shipping, how will you prevent it from affecting your users?

With OSS, we have options



- A private NuGet feed to host builds from forks can help out in a pinch
  - e.g. uptarget or downtarget a dependency
  - Incorporate a bugfix that the maintainer hasn't merged
  - Add InternalsVisibleTo("yourproject") a library (if you are feeling brave)

- Consider how to best contribute fixes or changes back to mainline repos.

-= mobile principles to live by =-

#4 win friends and influence people

aka. get involved in the community

Xamarin has a great core community

- For newcomers, it can be hard to know what's out there, which libraries are reliable, etc. it can help to start following the work of others
- Twitter is a popular platform amongst Xamarin and .NET developers
  - https://twitter.com/kwlothrop/lists/xamarin-mvps
- For a good summary of what's happening, subscribe to the Weekly Xamarin Newsletter

POP QUIZ

Answering the tough questions

Q1: your next cross platform mobile app

Q1: your next cross platform mobile app

NATIVE vs **WEB**

Q2: To Share UI, or not to Share UI

Xamarin

or

Xamarin.Forms

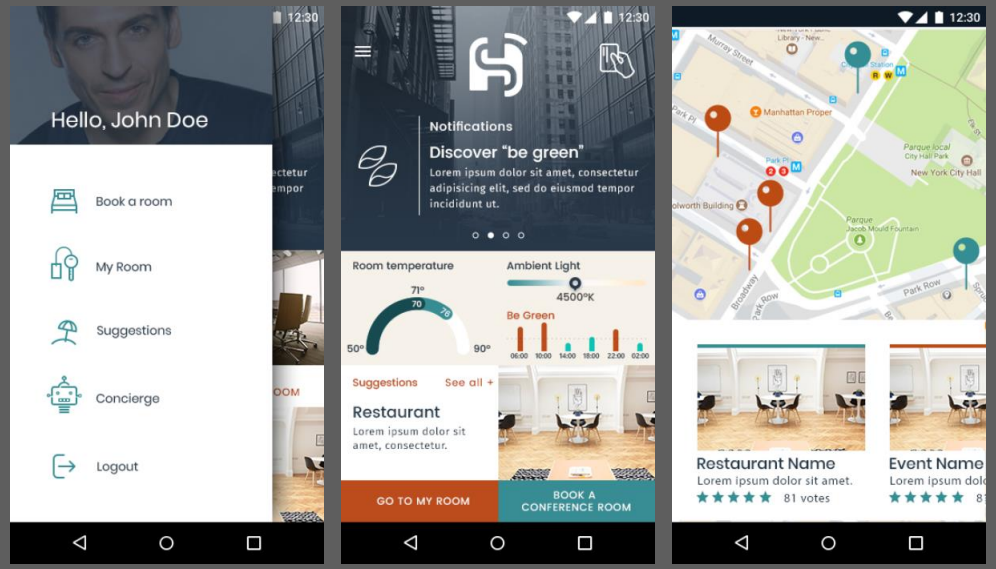
Q2: To Share UI, or not to Share UI

Xamarin or **Xamarin.Forms**

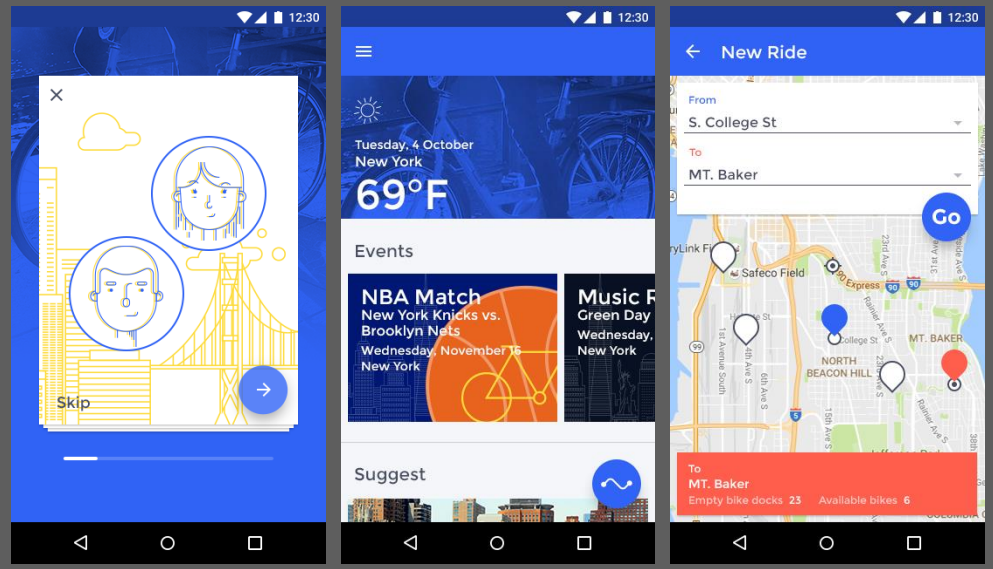
- Forms has been improving steadily over the years and is now open source
- You can do nice UIs with Forms: jsuarezruiz/xamarin-forms-goodlooking-UI
- Forms Embedding is a Great Thing Which You Should Definitely Use
  - see Alex Dunn's posts for some techniques (iOS, Droid)

Q2: To Share UI, or not to Share UI

jsuarezruiz/xamarin-forms-goodlooking-UI



SmartHotel



BikeSharing

Q2: To Share UI, or not to Share UI

Xamarin or **Xamarin.Forms**

- If you want flexibility: use a native shell with first class support for embedding
  - Keep Xamarin.Forms off the startup path
  - Full native control over screen transitions and exterior elements
  - Use viewmodel-first navigation and centralise your forms<->native smarts there (including XF initialisation)
  - Even the most UI intensive apps can likely benefit from a few forms pages

Q3: <UI /> or new UI();

XAML/Designer

or

Coded UI

But either is cool

Q3: <UI /> or new UI(); - iOS

XAML/Designer

- **Storyboards**
  - ✓ Single file containing screens + transitions
  - ✓ Many 'new' features including static tableview content and prototype cells
  - ✗ Source control nightmare
- **XIBs**
  - ✓ File per view – easy reuse, source control
  - ✗ 'New' features not supported
- **Both**
  - ✓ Easy visualisation, easy? constraints
  - ✗ Difficult to model dynamic layouts

Coded UI

- **Coded UI**
  - ✓ "Anything is possible" – dynamic layouts, subclassing/reuse
  - ✓ No magic – you control how and when your view is created
  - ✗ Writing constraints in code can be arduous – DSLs (e.g. FluentLayout, EasyLayout+) help
  - ✗ No design-time/preview unless you use something like Continuous

Q3: <UI /> or new UI(); - Android

XAML/Designer

- **AXML**
  - ✓ The 'right' way to do Android UI
  - ✓ If you have trouble with the Xamarin.Android designer you can use the Android Studio one.

Coded UI

- **Coded UI**
  - × Not impossible, just not advisable
  - × Some discussion here

Q3: <UI /> or new UI(); - Xamarin.Forms

XAML/Designer

- **XAML**
  - ✓ Good tooling support in wrt previewer and live reloader.
  - ✓ On VS4Mac, Mfractor adds many significant QoL and productivity enhancements to the XAML experience in particular
  - ✓ XF developer mindshare tends towards XAML-based views

Coded UI

- **Coded UI**
  - ✓ "Anything is possible"
  - ✗ Some constructs can be more verbose to specify in code
  - ✓ Some constructs can be less verbose to specify in code
  - ✗ No design-time/preview unless you use something like Continuous
  - ✓ Use CSharpForMarkup and thank Vincent later

Q4: Which IDE

Visual Studio

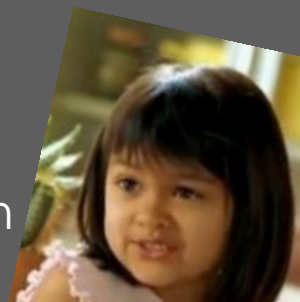
- ⚠ Resharper
- ✓ UWP
- ✓ Generally better editor experience (IMO)
- ✓ Consolidate NuGet packages

vs

VS4Mac

- ✓ MFractor
- ✓ Better iOS experience – designer, build/deploy
- ⚠ Some features come to mac later than win

I use both



Q5: Your dev machine and you

Parallels vs Bootcamp

ESX vs Hyper-V

Q5: Your dev machine and you

The true challenge of cross platform:

- You need a Mac to build/run iOS apps
- You need to be running Windows + Hyper-V to run the UWP device emulator
- Android is chill and runs everywhere, thanks buddy

What can we do?

- If you want to keep your options open, a Mac is a ~~safe~~ expensive bet
  - Use Bootcamp, you can boot Windows direct or use Parallels/VMWare from OSX
    (Parallels seems to perform best, this result has been consistent for several releases now)
  - If you're brave, you can also boot your OSX partition in VMWare when running in Bootcamp: https://pigiuz.wordpress.com/2013/11/22/how-to-make-a-vm-boot-your-osx-partition-from-windows/

Spotlight

on some cool li

how to use `HttpClient` to query a JSON API

- Step 1: **Reconsider.**

hello Refit

Refit gives you automatic type-safe api client generation

```
public interface IGitHubApi
{
    [Get("/users")]
    Task<User> GetAllUsers();

    [Get("/users/{user}")]
    Task<User> GetUser(string user);
}
```

Create an interface for your API, annotated with verb and URL stem

```
var gitHubApi = RestService.For<IGitHubApi>("https://api.github.com");
var ryan = await gitHubApi.GetUser("rdavisau");
```

Type-safe API methods, async, (de)serialization come for free!

Supports all the best verbs, body parameters, authentication, static and dynamic request headers and generic interface definitions.

Publish an internal nuget package with your Refit api definitions on every change and have strongly-typed access to your apis from anywhere (e.g. in linqpad)

persisting data – Akavache

Akavache gives you async-friendly key-value persistence + more

```
BlobCache.ApplicationName = "My Cool App";
await BlobCache.UserAccount.InsertObject("rdavisau", ryan);

// ...later, in another part of town...

var ryanFromCache = await BlobCache.UserAccount.GetObject<User>("rdavisau");
```

- Has a GetAllObjects<T>() method
- Facilitates caching and expiry: GetOrFetchLatest(string key, Func<Task<T>, DateTimeOffset expiry)
- Add encryption using this one simple trick [link]
- This tutorial shows you how to do the same using a randomly generated encryption key stored in the keychain (rather than relying on user input) [link] **this implementation assumes all inputs (serialised values) will be under the (block size – padding length: 256 – 11 = 245 bytes), If you are storing larger payloads you can use the `**CreateEncryptedData**`** **method on iOS and a technique like [this] on android**

super duper animations - lottie

Lottie renders Adobe After Effects animations exported as json with Bodymovin, giving you high performance animations for little effort

- Support for iOS and Android natively, or via Xamarin.Forms
- Many animations available at www.lottiefiles.com for inspiration
- Great for keeping onboarding flows simple but impressive

cross platform apis with no fuss - essentials

Xamarin.Essentials provides easy access to cross-platform APIs for mobile applications.

- Accelerometer
- App Information
- Battery
- Clipboard
- Compass
- Connectivity
- Data Transfer (Share)
- Device Display Information
- Device Information
- Email
- File System Helpers
- Flashlight
- Geocoding
- Geolocation
- Gyroscope
- Launcher
- Magnetometer
- MainThread
- Maps
- Open Browser
- Orientation Sensor
- Phone Dialer
- Power
- Preferences
- Screen Lock
- Secure Storage
- SMS
- Text-to-Speech
- Version Tracking
- Vibrate

If you like interfaces, try Essential.Interfaces too!

y even xaml am I rite

CSharpForMarkup is a small set of extensions designed to make writing Forms UI in code fluent and more declarative.

XAML

```
<Entry Placeholder="E.g. 123456" Keyboard="Numeric"
       Margin="{StaticResource fieldMargin}" HeightRequest = "44"
       Grid.Row="2" Grid.ColumnSpan="2"
       Text="{Binding RegistrationCode, Mode=TwoWay}" />
```

C# (XAML-like)

```
new Entry { Placeholder = "E.g. 123456", Keyboard = Keyboard.Numeric,
            Margin = fieldMargin, HeightRequest = 44 }
           .Row(2) .ColSpan(2)
           .Bind(nameof(vm.RegistrationCode), BindingMode.TwoWay),
```

C# (concise)

```
new Entry { Placeholder = "E.g. 123456", Keyboard = Keyboard.Numeric }
           .Row(2) .ColSpan(2) .Margin(fieldMargin) .Height(44)
           .Bind(nameof(vm.RegistrationCode), BindingMode.TwoWay),
```

Simple helpers that make coded-ui life easier:

Execute code inline

```
new ListView { }.Invoke(l => l.ItemTapped += MyListView_ItemTapped)
```

Assign controls inline

```
new ListView { }.Assign(out MyListView),
```

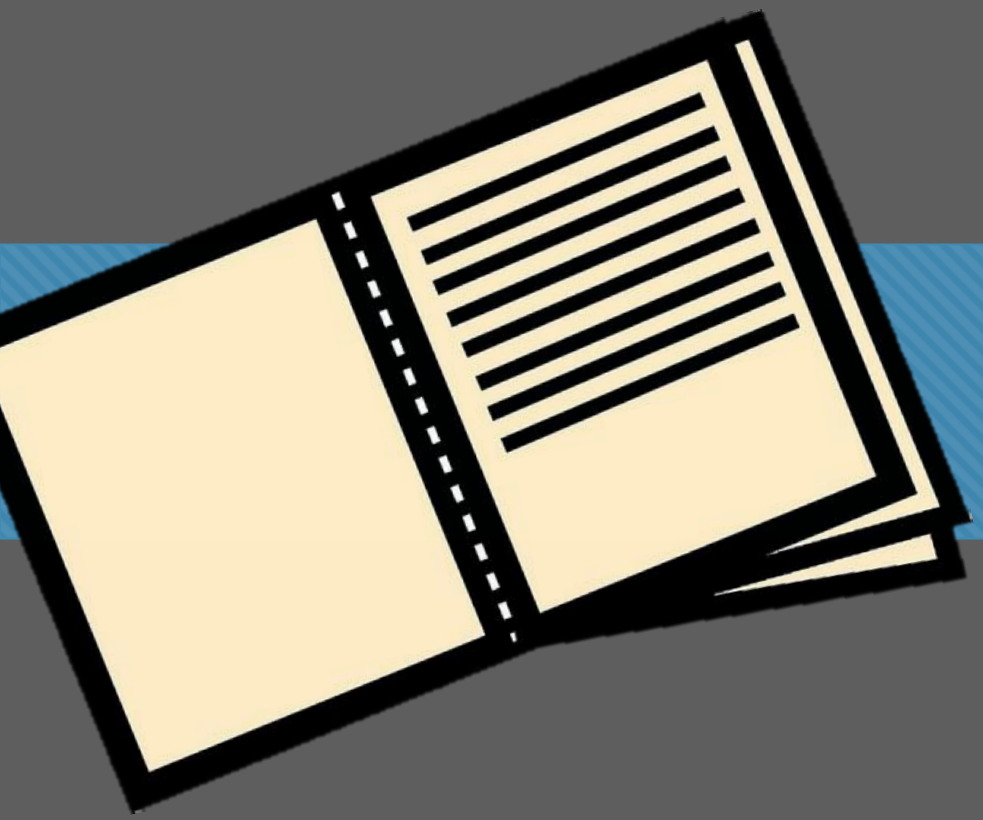
Enum-based row/cols

```
enum Row { Prompt, Header, Entry, Button }
enum Col { Label, Validation }

View BuildContent() => new Grid
{
    RowDefinitions = Rows.Define(
        (Row.Prompt, 170),
        (Row.Header, 75),
        (Row.Entry , GridLength.Auto),
        (Row.Button, GridLength.Auto)),

    ColumnDefinitions = Columns.Define(
        (Col.Label     , 160),
        (Col.Validation, GridLength.Star)),

    Children = {
        new Label { }
                .Row (Row.Prompt) .ColSpan (All<Col>())
                .Bind (nameof(vm.Prompt)),
```



Resource roundup

docs/learning

- Xamarin Docs
  https://docs.microsoft.com/en-us/xamarin/
- Xamarin University
  https://university.xamarin.com
- Books
  - Mastering Xamarin.Forms - https://www.packtpub.com/application-development/mastering-xamarinforms-second-edition
  - Xamarin in Action - https://www.manning.com/books/xamarin-in-action
  - You, I and ReactiveUI - https://kent-boogaart.com/you-i-and-reactiveui/
  - Creating Mobile Apps with Xamarin.Forms - https://docs.microsoft.com/en-us/xamarin/xamarin-forms/creating-mobile-apps-xamarin-forms/

(social) media

- Xamarin MVP List
  https://twitter.com/kwlothrop/lists/xamarin-mvps
- Public Xamarin Slack
  https://xamarinchat.herokuapp.com/
- Planet Xamarin – blog aggregation
  https://www.planetxamarin.com/
- Xamarin Gitter channels
  - iOS/MacOS: xamarin/xamarin-macios
  - Android: xamarin/xamarin-android
  - Forms: xamarin/Xamarin.Forms

Subscribe to the Weekly Xamarin Newsletter!
http://www.weeklyxamarin.com

frameworks/libraries

- MVVM
  - MvvmCross – https://github.com/MvvmCross/MvvmCross - Popular cross-platform MVVM library, Native/Forms
  - ReactiveUI – https://github.com/reactiveui/ReactiveUI - Reactive Extensions and FRP principles, Native/Forms
  - MVVM Light Toolkit – http://www.mvvmlight.net/ - Light MVVM, Native/Forms
  - Prism - https://prismlibrary.github.io/docs/ - Forms
  - FreshMvvm - https://github.com/rid00z/FreshMvvm – Forms
- Libraries from the previous section
  - Akvache - https://github.com/akavache/Akavache - An asynchronous, persistent key-value store
  - Refit - https://github.com/paulcbetts/refit - The automatic type-safe REST library for Xamarin and .NET
  - Lottie - https://github.com/martijn00/LottieXamarin - Beautiful animations in mobile apps
  - Xamarin.Essentials - https://docs.microsoft.com/en-us/xamarin/essentials/ - Cross platform APIs for mobile
  - CSharpForMarkup - https://github.com/VincentH-Net/CSharpForMarkup - Fluent, declarative coded UI for XF

frameworks/libraries
(continued)

- Library Roundups
  - Awesome Xamarin - https://github.com/XamSome/awesome-xamarin
  - Awesome Xamarin.Forms - https://github.com/jsuarezruiz/awesome-xamarin-forms
  - Goodlooking Xamarin.Forms - https://github.com/jsuarezruiz/xamarin-forms-goodlooking-UI

game dev

- UrhoSharp
  - **UrhoSharp** [repo]
    https://github.com/xamarin/urho
  - **Docs** [official Xamarin documentation]
    https://docs.microsoft.com/en-us/xamarin/graphics-games/urhosharp/
  - **Introduction to UrhoSharp presentation**
    https://ryandavis.io/introduction-to-urhosharp/
  - **Virtual/Mixed Reality Apps with C#** [Build 2017 session]
    https://channel9.msdn.com/Events/Build/2017/T6052
  - **Urho3d** [official website]
    https://urho3d.github.io/

ci/cd

- App Center
  - **appcenter.ms** [official website]
    https://appcenter.ms/
  - **Docs** [official Xamarin documentation]
    https://docs.microsoft.com/en-us/appcenter/
  - **Introduction to Visual Studio App Center presentation**
    https://ryandavis.io/introduction-to-visual-studio-app-center/
- Azure DevOps
  - **Step by step guides for setting up Xamarin Builds** [james montemagno's blog posts]
    https://montemagno.com/tag/continuous-integration/
- Misc
  - **Mobile.BuildTools** [helper library for CI tasks and Xamarin Projects]
    https://github.com/dansiegel/Mobile.BuildTools

native platforms

- iOS
  - **Ray Wenderlich's site** - http://www.raywenderlich.com/
  - **iOS Dev Weekly** - https://iosdevweekly.com/
  - **iOS Dev Center** [apple's own] - https://developer.apple.com/ios/
- Droid
  - **developer.android.com** [it's official] - http://developer.android.com/index.html

+ approximately 999,999,999,999 courses on Pluralsight

podcasts/videos

- The Xamarin Show - https://channel9.msdn.com/Shows/XamarinShow
- Merge Conflict - https://www.mergeconflict.fm
- Gone Mobile - https://www.gonemobile.io

questions / thanks