

Xamarin Overview

CSE 686 (Internet Programming)

Spring 2019

Mike Corley

Background

- Microsoft supported open source .Net based (multiplatform) mobile development framework
 - Android
 - iOS
 - Mac
 - Windows (UWP)
- Enables (native) cross platform application development on Android and iOS
- Originated in 2011 as an open source framework, Microsoft owned and supported since 2016
 - Available for free
 - ships with every edition of Visual Studio (we use the free community edition).
 - <https://visualstudio.microsoft.com/xamarin/>
 - <https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/>
 - https://www.theregister.co.uk/2016/03/31/xamarin_tools_code_free_and_open_source/

Why Xamarin?

- Developing for multiple platforms: Android, iOS, UWP
 - Mobile Platforms are different!
 - Tools and developer expertise
 - Development environment
 - Android Studio Versus Xcode
 - Java versus Objective-C (programming languages)
 - Android SDK versus Cocoa (User Interface frameworks)
 - ART (Android Runtime) versus iOS runtime
 - Platform constraints: app deployment/management/look and feel
 - Absolutely separate code bases
 - No portability or reuse between code / platform
 - Higher developer cycle/overhead to develop/maintain/manage
 - Xamarin solves many of these issues by a single paradigm (C#), while enabling (native) Android and iOS application development

Development Comparison

Actual Development

Platform	Android	iOS	Windows
SDK	Android SDK	iOS SDK	Windows SDK
Language	Java	Objective-C / Swift	C#/VB
UI Definition	XML	XIB (Static XML Format) / Storyboard	XAML
IDE	Android Studio by IntelliJ	XCode	Visual Studio

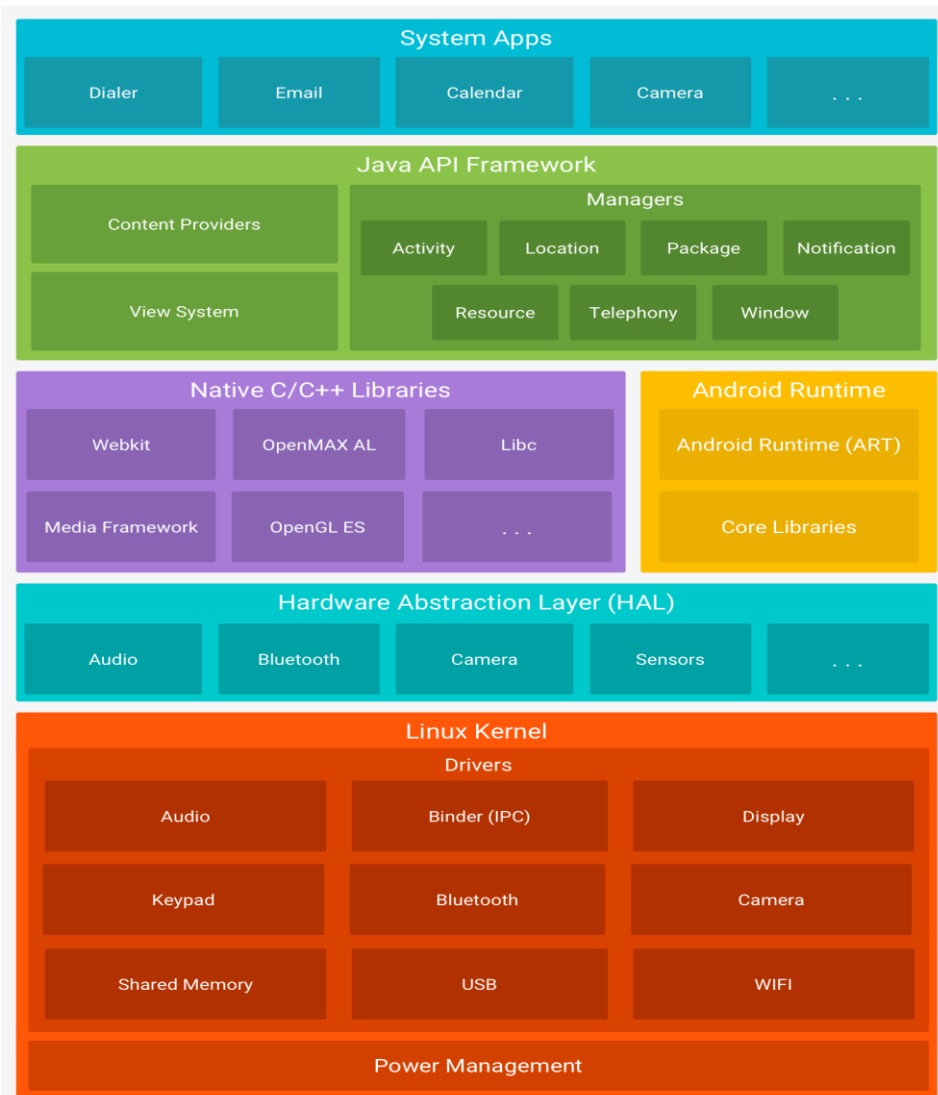


Written by : Mr Hery

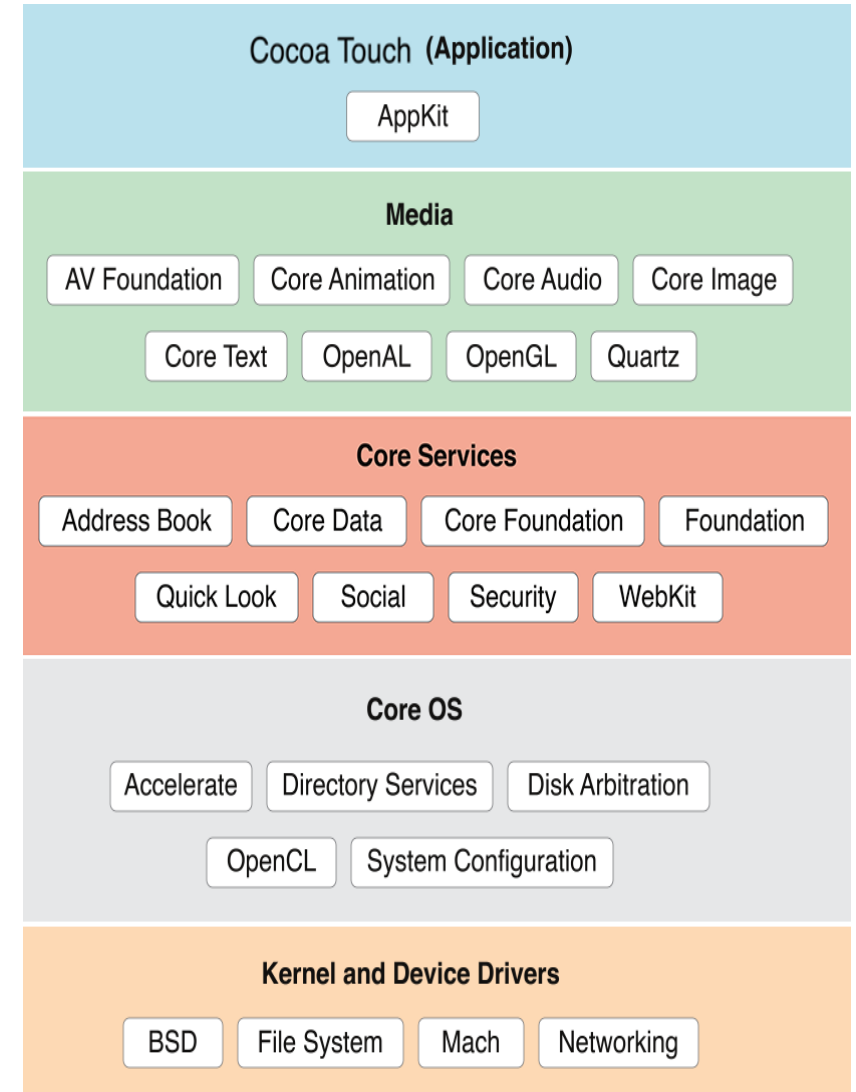
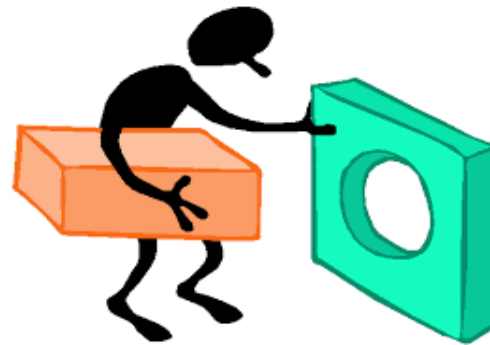
Copyright* HeryIT* JM0670283-X (2017)

Source: <https://www.slideshare.net/iamsharper/overview-to-xamarin-understanding-xamarin-architecture>

Android Versus iOS (Architecture)



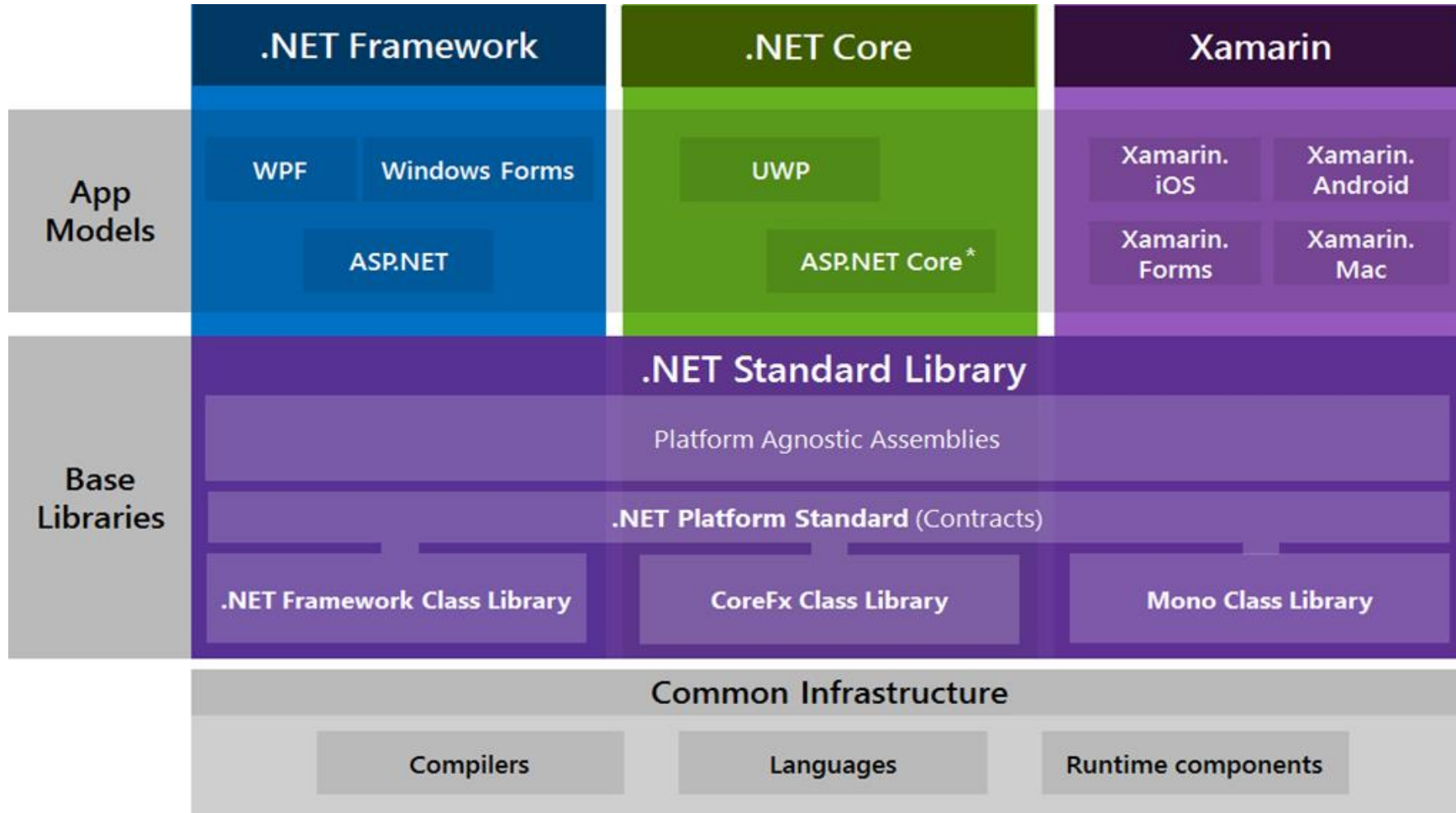
Portability / reuse??



https://developer.android.com/guide/platform/images/android-stack_2x.png

<https://www.dotnettricks.com/learn/xamarin/understanding-xamarin-ios-build-native-ios-app>

.Net Implementations: Architecture Overview



<https://blogs.msdn.microsoft.com/cesardelatorre/2016/06/27/net-core-1-0-net-framework-xamarin-the-whatand-when-to-use-it/>

Xamarin Overview

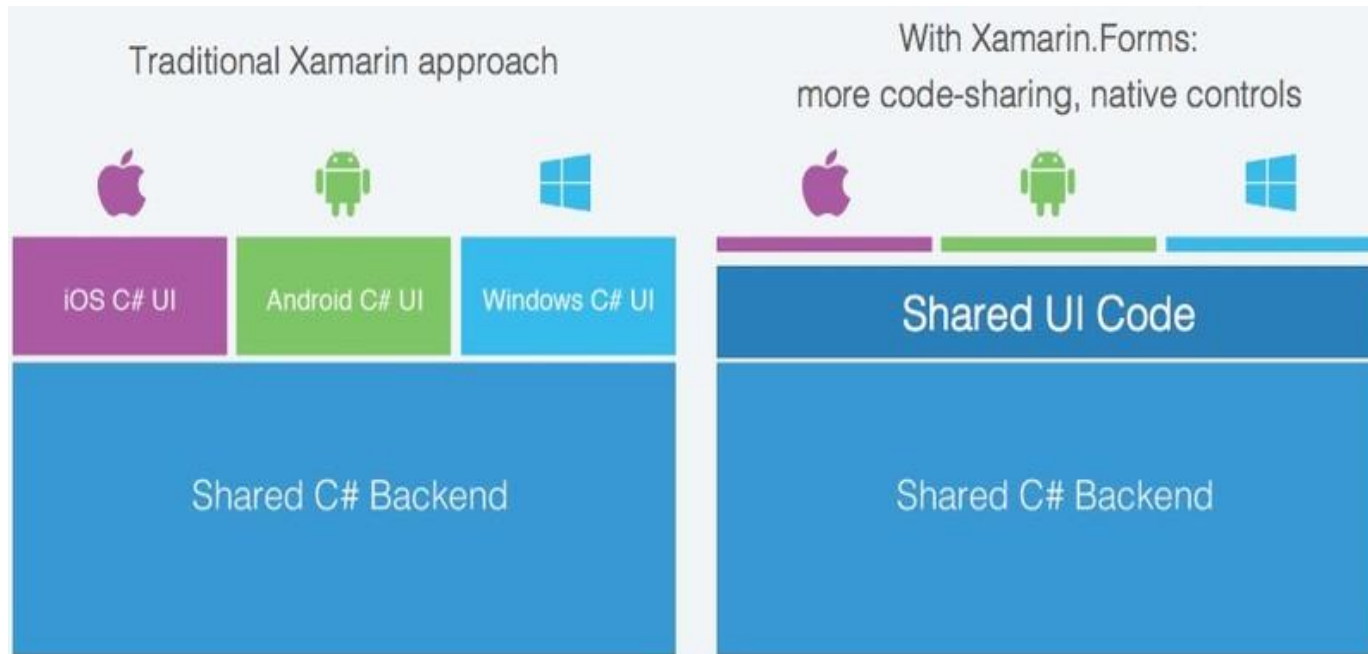
- Xamarin supports two foundational App development approaches

1. Xamarin Native

- <https://docs.microsoft.com/en-us/xamarin/cross-platform/get-started/introduction-to-mobile-development>

2. Xamarin Forms

- <https://docs.microsoft.com/en-us/xamarin/get-started/>



<https://applikeysolutions.com/blog/xamarin-forms-vs-xamarin-native-what-fits-you-best>

1. Xamarin Native

- Develop native Android and iOS mobile applications!
 - How??
 - Xamarin provides complete Bindings and Interop for the underlying platform SDKs
 - *Xamarin.iOS.dll* -- compiles into native iOS using the native tools and runtime
 - Objective-C, etc.
 - *Xamarin.Android.dll* -- compiles into native Android assembly code using the native tools and runtime,
 - Java, etc
 - Interop enables access full spectrum of the underlying platform and device libraries
 - Xamarin apps are 100% native, consumed/installed/managed on the native platform with native tools

<https://docs.microsoft.com/en-us/xamarin/android/internals/architecture>

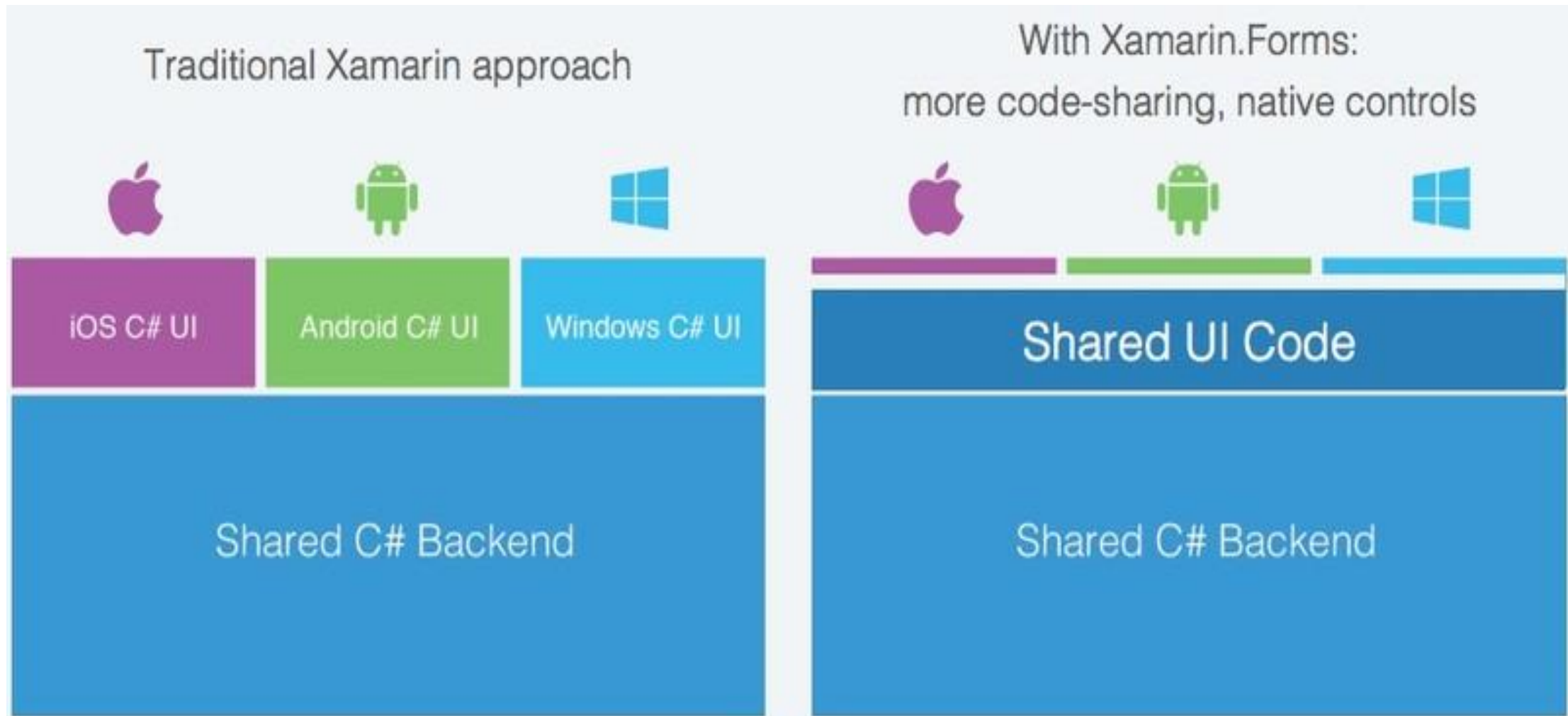
2. Xamarin Forms

- Portable UI framework for mobile apps.
 - Uses native libraries to enable front ends development using portable/shared code.
 - How??
 - Shared application code is built as shared (.Net standard) library
 - .Net standard is formal API specification supported by all .Net framework implementations “write once run everywhere”.
 - Android and iOS “native” projects consume the shared library as the portable mobile application.
 - Interop services enable access full spectrum of the underlying platform and device libraries

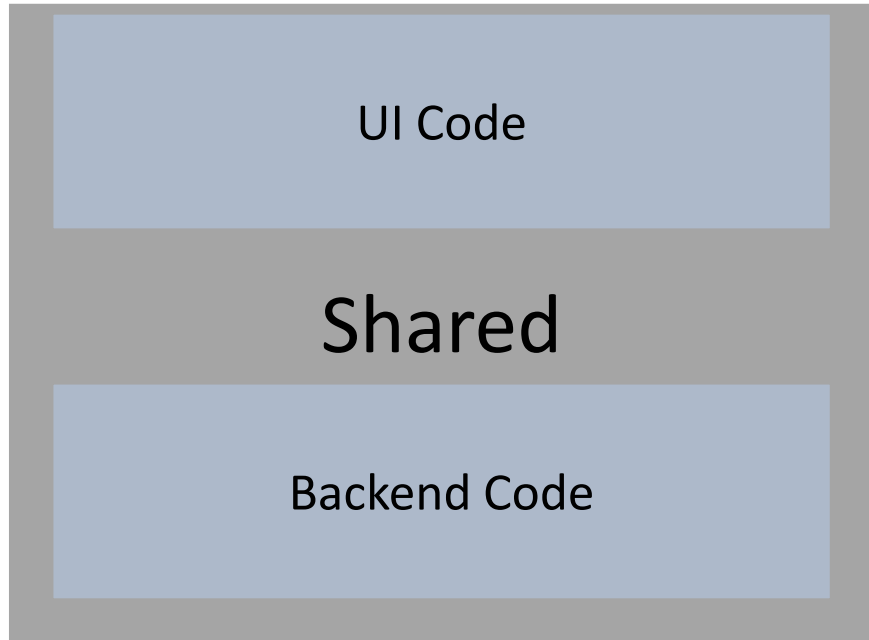
Xamarin Native apps allow for between (80-95%) cross-platform code [1]

<https://docs.microsoft.com/en-us/dotnet/standard/net-standard>

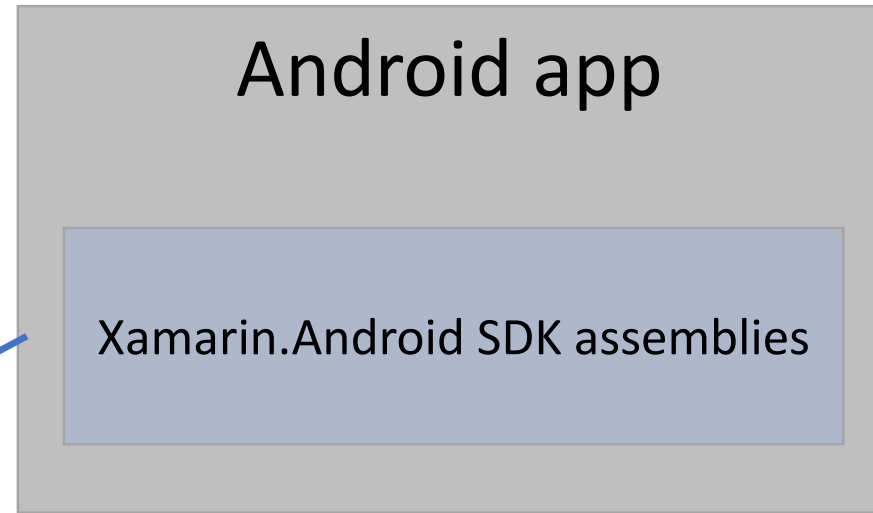
Xamarin Native versus Xamarin Forms



<https://images.app.goo.gl/rJG6mqommi9ePNyn9>



.Net standard (shared library)

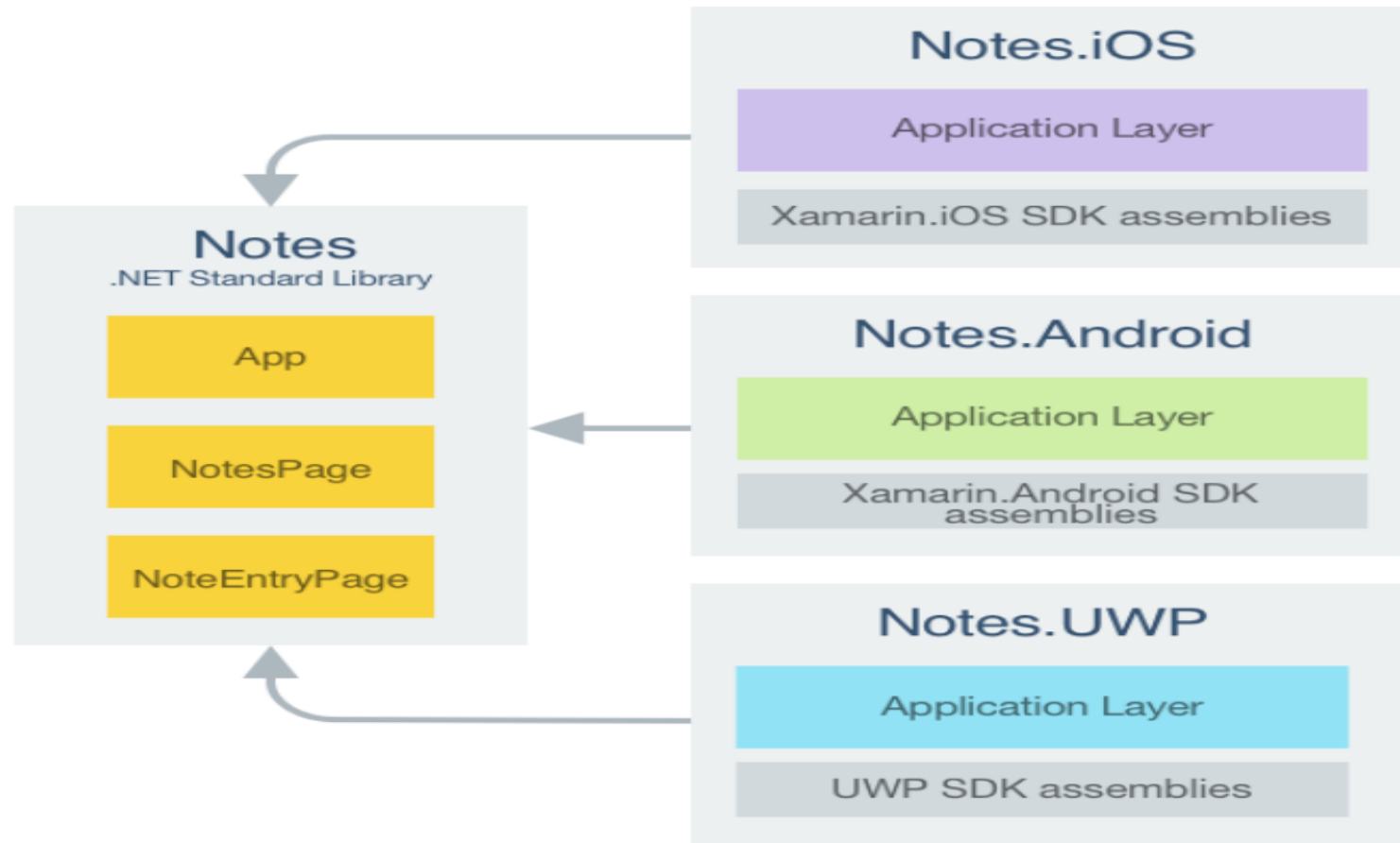


Native apps consume shared code



Architecture and application fundamentals

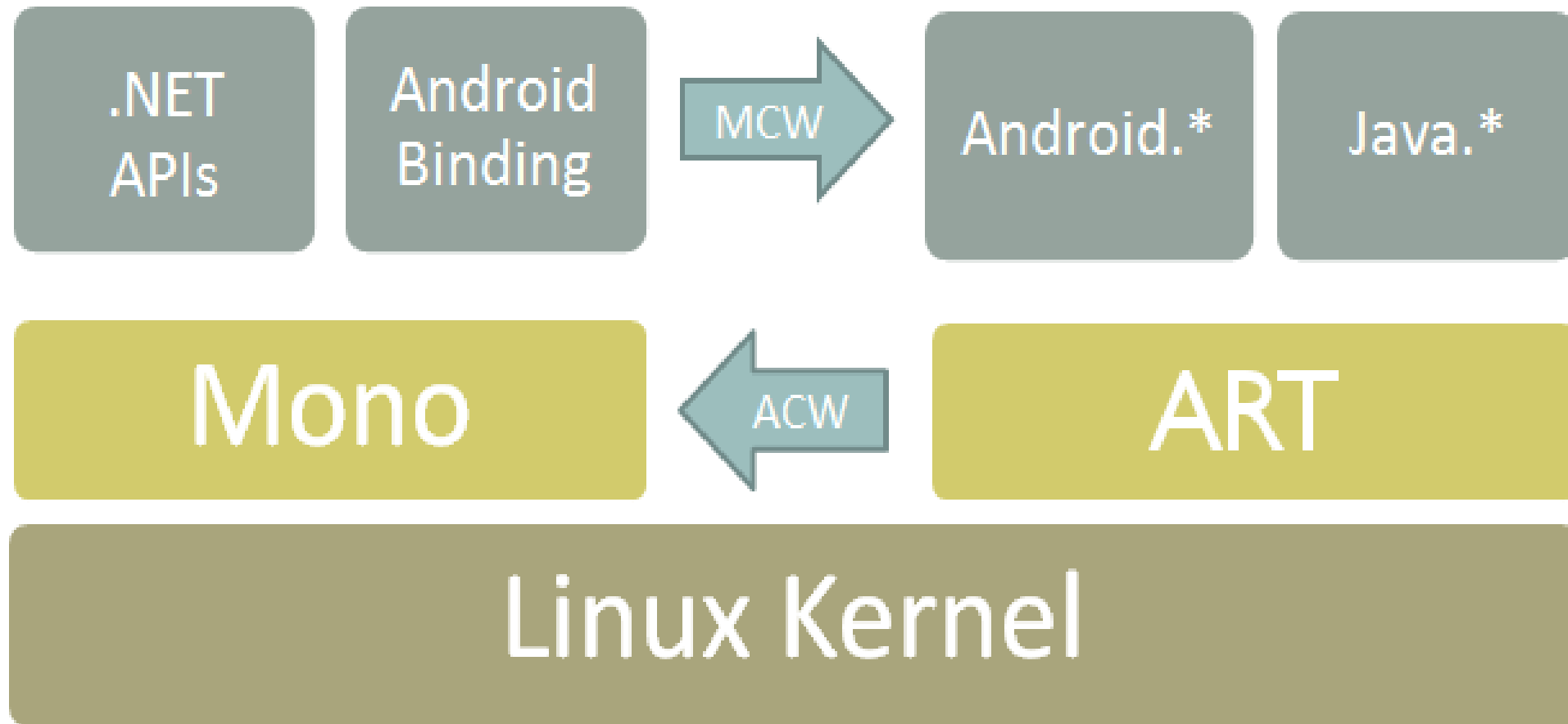
A Xamarin.Forms application is architected in the same way as a traditional cross-platform application. Shared code is typically placed in a .NET Standard library, and platform-specific applications consume the shared code. The following diagram shows an overview of this relationship for the Notes application:



To maximize the reuse of startup code, Xamarin.Forms applications have a single class named `App` that is responsible for instantiating the first page that will be displayed by the application on each platform, as shown in the following code example:

<https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/code-sharing>

Xamarin Architecture (Android SDK bindings)



Benefits?

- Different platforms use different code, different development paradigms/frameworks, different tool sets, and provide different environments with constraints on deployment and use.
 - No portability/sharing across platforms
 - More expensive: need skilled resources with expertise in respective tools and environments
 - Longer development cycles and maintenance of multiple code bases
- Xamarin solves much of these issues
 - Lots of code reuse and portability
 - Single (shared) code base
 - Single set of development tools/environment
 - Shorter development cycles.
 - Access to advanced development constructs: e.g. dependency injection
 - Xamarin Native offers a high degree of “code reuse” – (75-80%) [1]
 - Xamarin Forms offers a high degree of “code portability” – (80-95%) [1]
- <https://arctouch.com/blog/xamarin-forms-more-capable-than-you-think/>

Risks?

- Xamarin Native
 - SDK bindings and interop utilize the native Android and iOS platform SDKs
 - Lots of reported code reuse (75-80%) [1]
 - Similar App performance to native tools/environments
 - Larger App footprint
- Xamarin Forms
 - Traditionally, performance/responsiveness constraints for certain applications with high graphics requirements (games), highly dynamic content (animation etc.)
 - Performance had much improved with Xamarin Forms [2]
 - Applications that require minimalist App size requirements
 - Xamarin Forms Apps and larger than Native Apps
 - Shared Code is Not boxed in!
 - Shared code and can access native platform SDKs.

<https://docs.microsoft.com/en-us/xamarin/android/deploy-test/app-package-size>

Useful Links

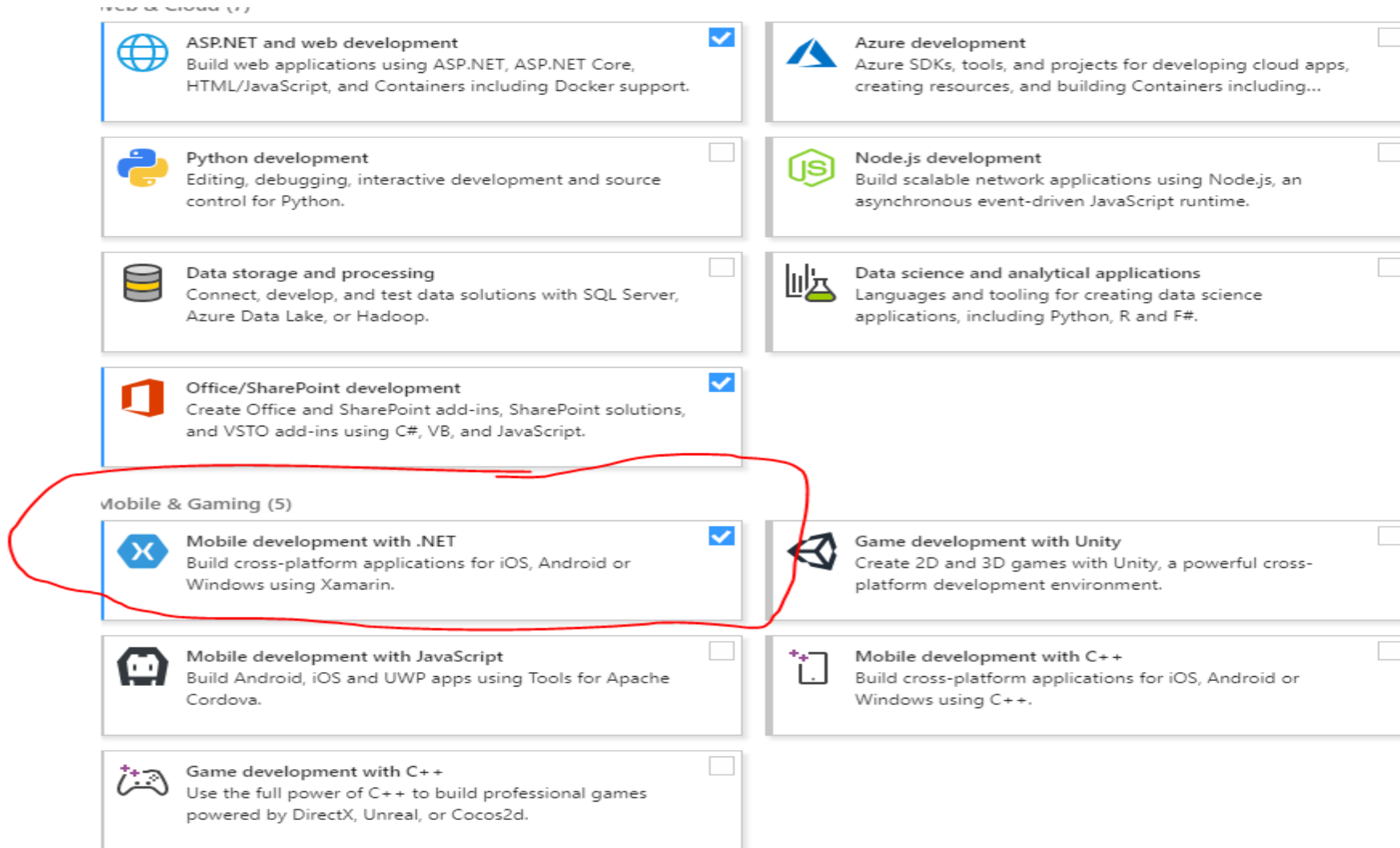
- <https://docs.microsoft.com/en-us/xamarin/>
- <https://docs.microsoft.com/en-us/xamarin/android/get-started/>
- <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>
- <https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform>
- <https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/architecture>
- <https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/code-sharing>
- <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/map>

Sources for additional information

- [1] <https://applikeysolutions.com/blog/xamarin-forms-vs-xamarin-native-what-fits-you-best>
- [2] <https://arctouch.com/blog/xamarin-forms-more-capable-than-you-think/>
- <https://insanelab.com/blog/mobile-development/xamarin-mobile-application-development/>
- <https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/>
- <https://hackernoon.com/the-pros-and-cons-of-xamarin-for-cross-platform-development-2a31c6610792>

Visual Studio 2017 (latest update)

- Start->Run: Type “Visual Studio Installer



The screenshot shows the Visual Studio Installer workload selection screen. The workloads are organized into categories. The 'Mobile & Gaming (5)' category is circled in red. The 'Mobile development with .NET' workload is selected, indicated by a blue checkmark in the top right corner of its box. Other workloads have either a blue checkmark or an unchecked checkbox.

Workload	Description	Selected
ASP.NET and web development	Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.	Yes
Python development	Editing, debugging, interactive development and source control for Python.	No
Data storage and processing	Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.	No
Office/SharePoint development	Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.	Yes
Mobile & Gaming (5)		
Mobile development with .NET	Build cross-platform applications for iOS, Android or Windows using Xamarin.	Yes
Mobile development with JavaScript	Build Android, iOS and UWP apps using Tools for Apache Cordova.	No
Game development with C++	Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.	No
Azure development	Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including...	No
Node.js development	Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.	No
Data science and analytical applications	Languages and tooling for creating data science applications, including Python, R and F#.	No
Game development with Unity	Create 2D and 3D games with Unity, a powerful cross-platform development environment.	No
Mobile development with C++	Build cross-platform applications for iOS, Android or Windows using C++.	No

Live Demo: Cross platform Prototype (UWP, Android, iOS)

