MULTIPLATFORM MOBILE APPLICATIONS DEVELOPMENT

dr Michał Kuciapski
Higher Banking School

AGENDA

1. Why cross-platform application development is important?
2. Approaches to mobile apps development
3. Strategies for developing cross-platform applications
4. User Interface design considerations
5. Cross-platform applications developemnt frameworks

Referenced materials:
1. Henning Heitkotter, Sebastian Hanschke, and Tim A. Majchrzak, Evaluating Cross-Platform Development Approaches for Mobile Applications, Web Information Systems and Technologies. 8th International Conference, WEBIST 2012, Springer Berlin Heidelberg 2013
2. Norbert Haberl, Cross Platform Development
3. Possibilities and drawbacks of the Xamarin platform, FH JOANNEUM University of Applied Sciences
4. Hammoudeh Alamri, Balsam Abdul Jabbar Mustafa, Software Engineering Challenges in Multi Platform Mobile Application Development, Journal of Computational and Theoretical Nanoscience
5. Ramya Balaraman, ROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT
6. Dinis Vieira, Strategies for Developing Cross-Platform Applications
7. Rabi Satter, .NET Cross Platform Development Strategy for Mobile, Cloud and Desktop Apps
8. Rohit Ghatol, Cross Platform Mobile Applications
9. Naveen Danturi, Pranay Mahendra, Mobile Application Development

Why cross-platform application development is important?

Who cares about Cross Platform?

- **2013** App Economy was **68 billion USD** according to DeveloperEconomics.com or roughly 10 USD per person
- **2016** estimated App Economy will be **143 billion USD** according to DeveloperEconomics.com or roughly 20 USD per person
- Problem is that there is **no OS monoply**. What is a developer to do?

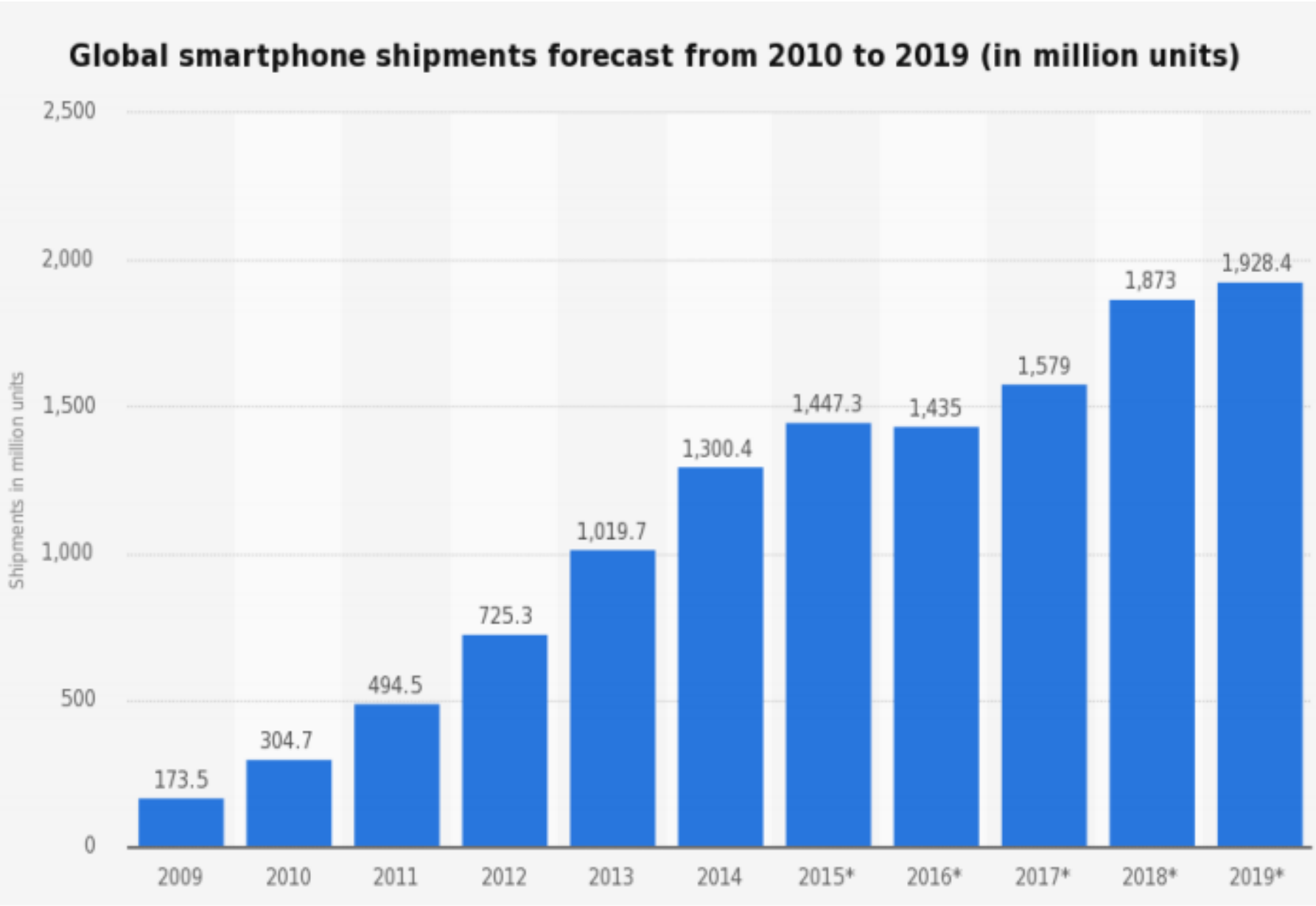


Figure 1 Global smartphone shipments forecast from 2010 to 2019 (in million units) [6]

85% of Facebook's users access the **site on mobile devices** , with more than one third exclusively using mobile. The money is following suit: in 2014, mobile provided 69% of Facebook's $3.6 billion advertising revenue[4]

Mobile is projected to account for 46.6% of global e-commerce by 2018.[5]

By 2016, **70% of the mobile workforce will have a smartphone** , with BYOD employees purchasing half of them; **90% of enterprises will have two or more platforms to support.** [6]

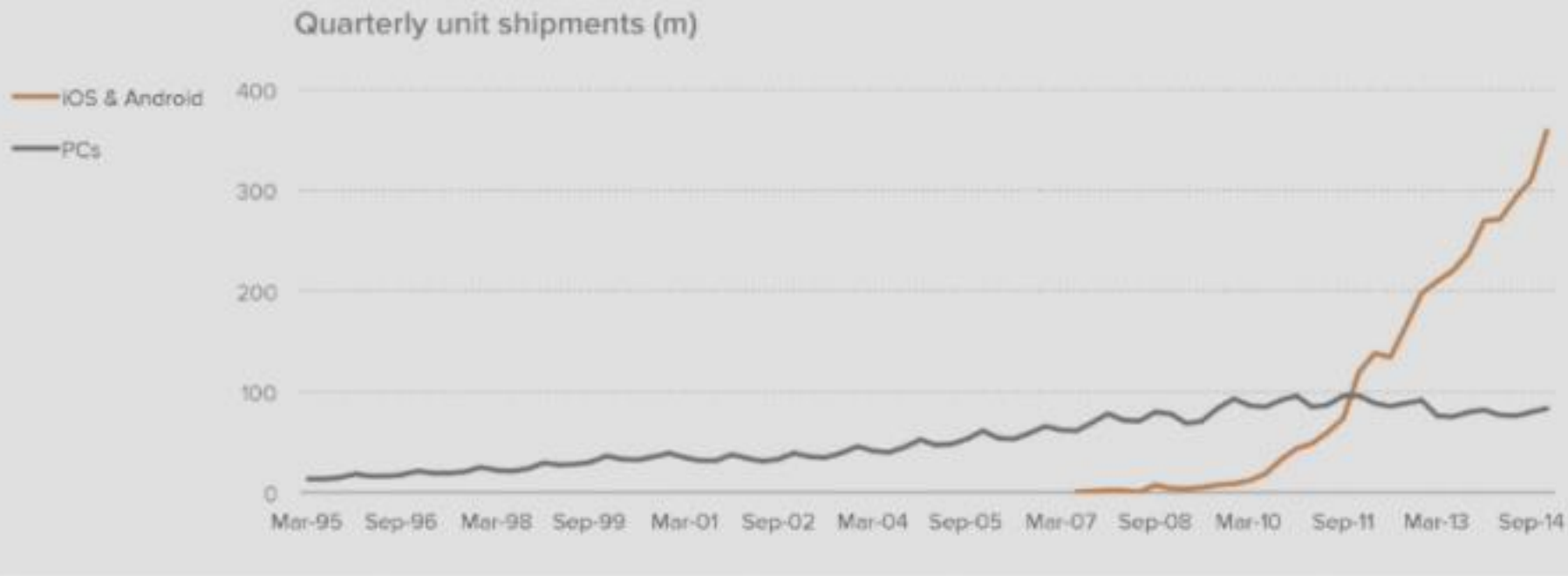
Figure 4 Key facts on going mobile [9]

Table 1 Smartphone OS Market Share, Q1 2012 – Q1 2015 [7]

Period		Android	iOS	Windows Phone	BlackBerry OS	Others
Q1 2015		78.0%	18.3%	2.7%	0.3%	0.7%
Q1 2014		81.2%	15.2%	2.5%	0.5%	0.7%
Q1 2013		75.5%	16.9%	3.2%	2.9%	1.5%
Q1 2012		59.2%	22.9%	2.0%	6.3%	9.5%

The smartphone industry dwarfs PCs

4bn people buying phones every 2 years instead of 1.6bn buying PCs every 5 years



14

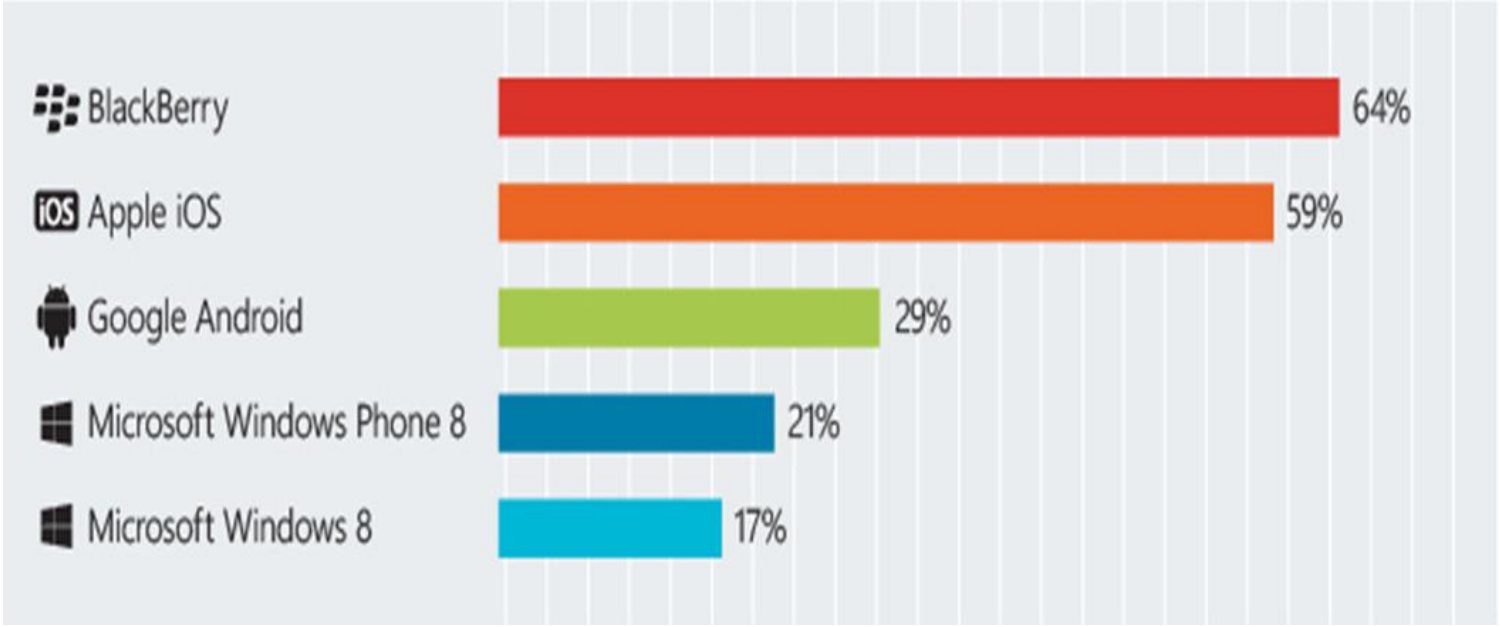
ANDREESSEN HOROWITZ

Source: Gartner, Apple, Google, a16z

No you can't just target Android!

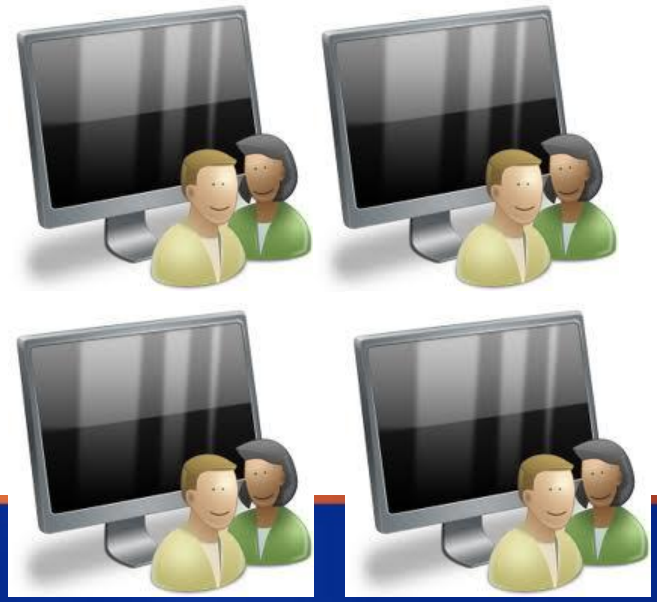
- Android dominates the phone market 81% shipping in Q3 2013 vs 13% iOS
- iOS dominates the tablet market 52% primary target is iPad only 28% primary target is Android
- Most devs (60%) make less than 500 USD per month per app
- iOS devs make 500-1000 USD per month per app vs. Android devs make 100-200 USD per month per app
- WP devs make < 50 USD per month per app
- In 2013 Contracting brought in 38 billion USD representing 56% of app economy

Development Difficulty



Blackberry and iOS are the most difficult platforms to develop for Windows 8 and Windows Phone 8 ranked as the easiest with Android falling in the middle.

Reaching Mobile Users



Mobile Features

Mostly Feature Sub Set

Complete Feature Set

Tablet Features

Almost Complete Feature Set

Complete Feature Set

User Interaction

Touch based

Accelerometer

Traditional

Compass

LOCATION AWARE

Location Aware and highly accurate

Can be Location Aware but approximate

SENSORS

Handy Camera and Voice Recording

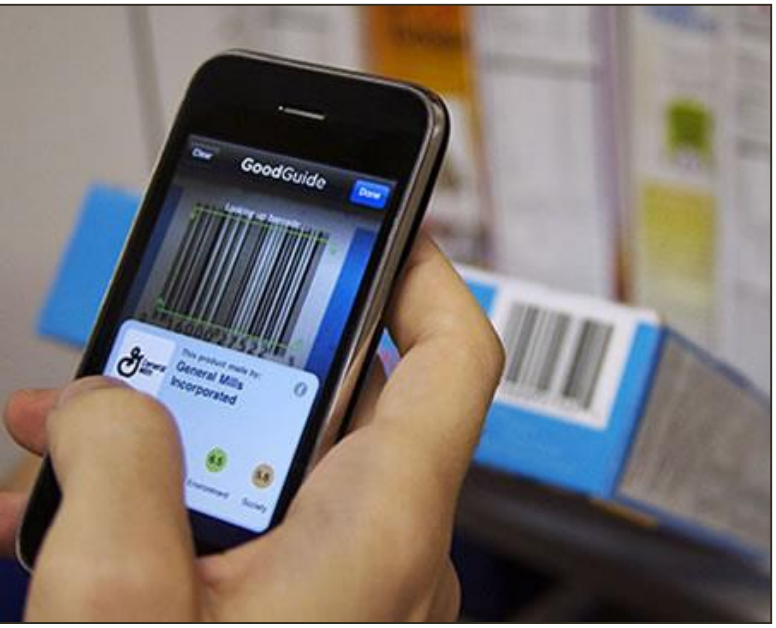
Upcoming NFC (Near Field Communication) turning phone into Credit Card, Access Card, Business Card Exchanger

Push Notifications

Push Notification

Notifying the User proactively

E.G SHOPPING APPLICATIONS



Scan a product's barcode to know if it has the lowest price.

If not, then navigate to a store which has the lowest price

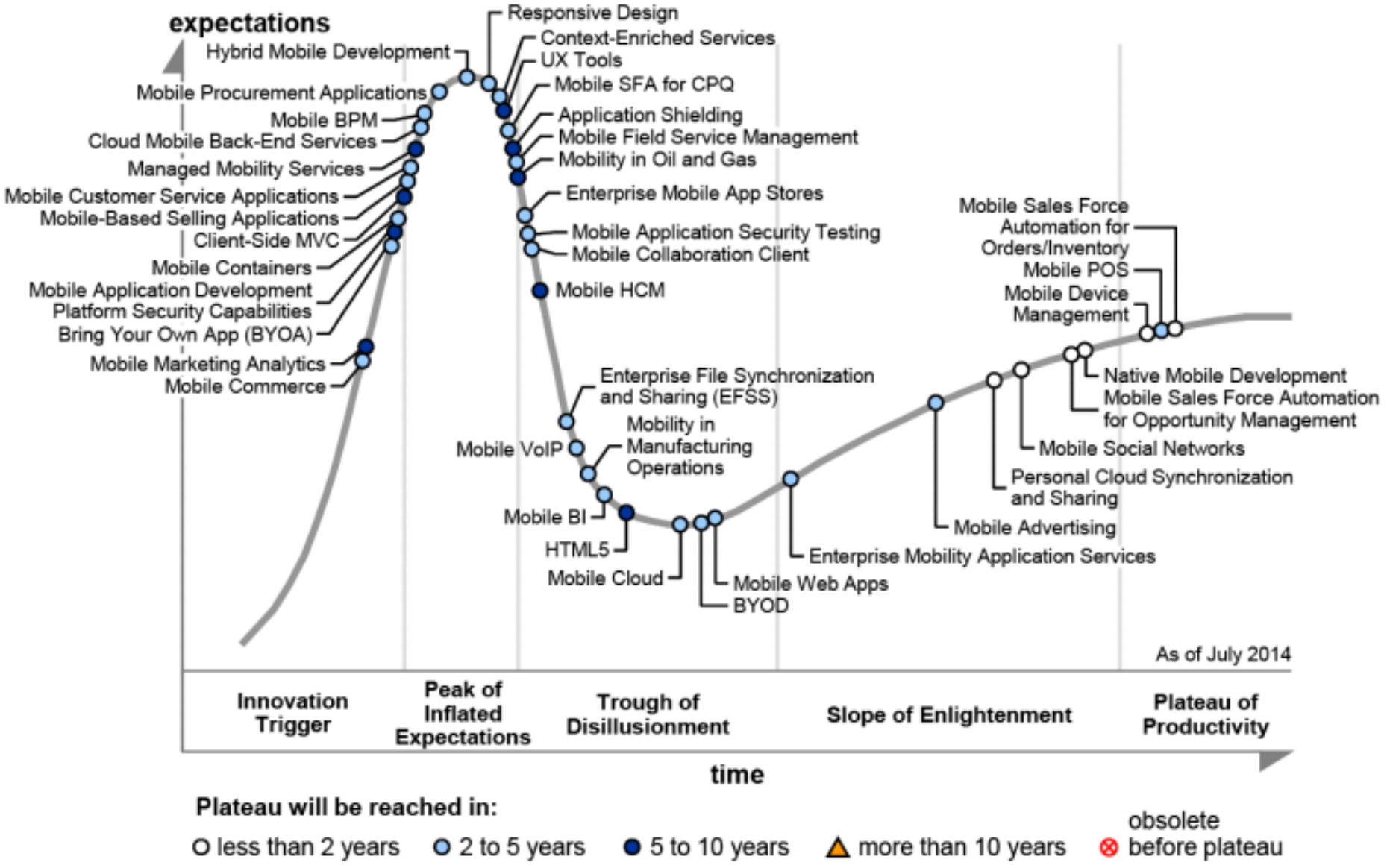


Figure 5 Hype Cycle for Mobile Applications [12]

OS Fragmentation

Fragmentation

Multiple Teams/Product

Multiple Teams/Products

Uniform User Experience

Uniform User Experience

Approaches to mobile apps development

Native apps

- Binary executable files on the device.
- Can access all API's made available by OS vendor.
- SDK's are platform-specific.
- Each mobile OS comes with its own unique tools and GUI toolkit.
- Different tools, languages and distribution channels associated with leading mobile operating systems

Native apps

PROS	CONS
Easy low-level hardware access services.	Code Reusability : Low
Easy access to high level services important to personal mobile experience.	Development & maintenance: Time-consuming & expensive.
Full use of all functionalities that modern mobile devices have to offer.	Designers are required to be familiar with different UI components of each OS.
High usability.	Upgrade flexibility: Low.

CROSS-COMPILATION

- Separates build environment from target environment.
- Platform-independent API using a mainstream programming language like JavaScript, Ruby or Java.
- The cross-compiler then transforms the code into platform-specific native apps.
- The software artifact generated can be deployed and executed natively on the device.

ADVANTAGES:

- Improved performance and User Experience.
- Full access to functionalities of underlying mobile OS and device specific capabilities.

DISADVANTAGES:

- Highly complex as cross-compilers are difficult to program.
- Need to be kept consistent with fragmented mobile platforms and operating systems available.

VIRTUAL MACHINE APPROACH

- A virtual machine is used to abstract the target platform details from the application's running code.
- The framework provides both the API and runtime environment.
- The runtime executes on the mobile device and enables interoperability between the device's OS and the mobile application.

ADVANTAGES:

- Improved performance and User Experience.
- Full access to functionalities of underlying mobile OS and device specific capabilities.
- Portability: VM's are easier to maintain & more flexible to extend.

DISADVANTAGES:

- Slower due to runtime interpretation latency.

MOBILE WEB APPS

- Use standard web technologies such as HTML 5, CSS 3 & JavaScript.
- Features of HTML 5 - Advanced UI components, access to rich media types, geolocation services & offline availability.
- Increasing popularity of HTML 5 in rendering engines such as WebKit.
- Runs on a standalone mobile web browser.
- Installed shortcut, launched like a native app.
- UI logic resides locally; makes the app responsive and accessible offline.

ADVANTAGES:

- Multiplatform support.
- Low development cost.
- Leverage existing knowledge.

DISADVANTAGES:

- Limited access to OS API's.

HYBRID APPS

- Combines native development with web technology.
- The web app runs inside a thin wrapper native app.
- The wrapper native app uses the OS API's to create an embedded HTML rendering engine which provides a bridge between the browser and device API's.
- The communication between web app and native app normally happens over JavaScript via custom built API's.

ADVANTAGES:

- Flexibility of web apps combined with feature richness of native apps.
- Simplified deployment and immediate availability.
- Leverage existing knowledge.

DISADVANTAGES:

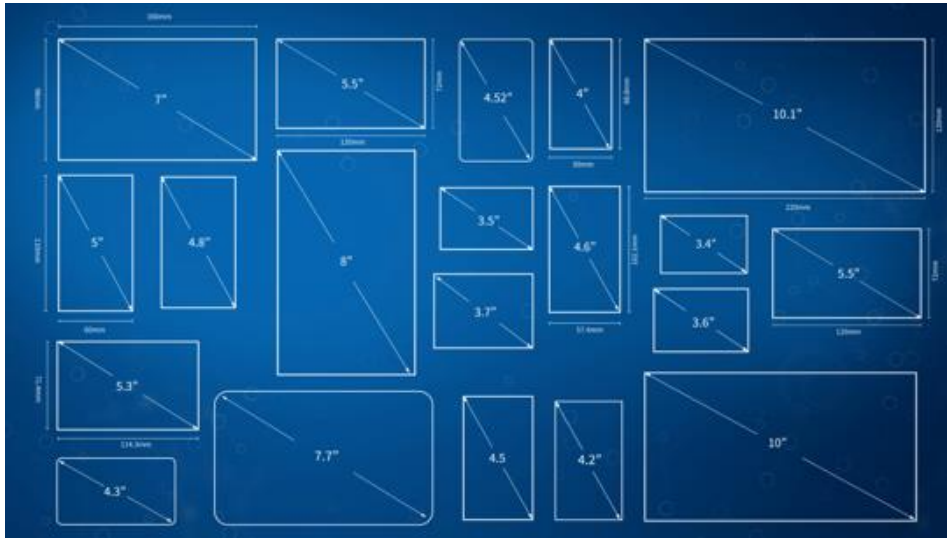
- Poorer user experience as compared to native apps.
- Access to advanced device capabilities normally restricted.

Strategies for developing cross-platform applications

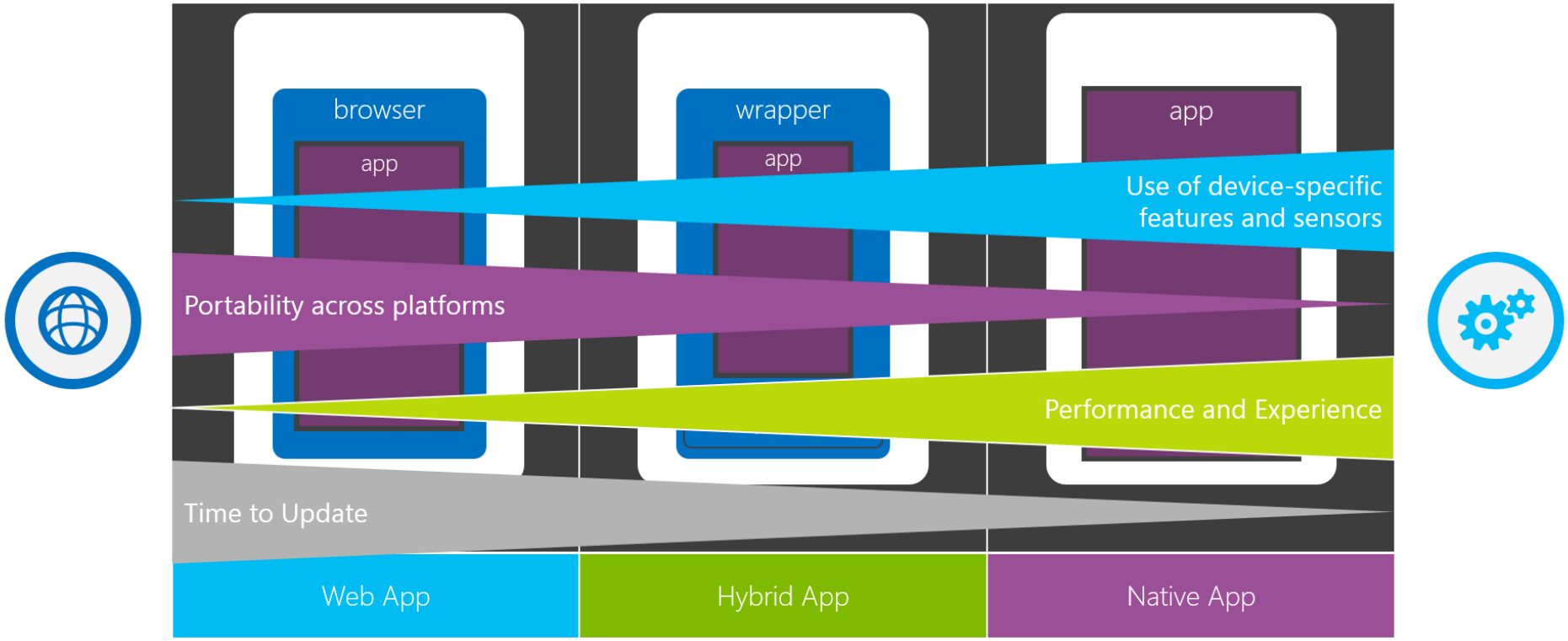
Cross-Platform Mobile Development

Building high-quality Apps is hard:

- Different presentation styles, interaction styles and software stacks
- Devices have different screen sizes, input modes and hardware capabilities
- New devices and OS versions are introduced multiple times per year
- Network connectivity and power levels fluctuate widely in typical usage scenarios
- New consumer applications regularly extend and revise the standards and set the bar higher for good mobile applications

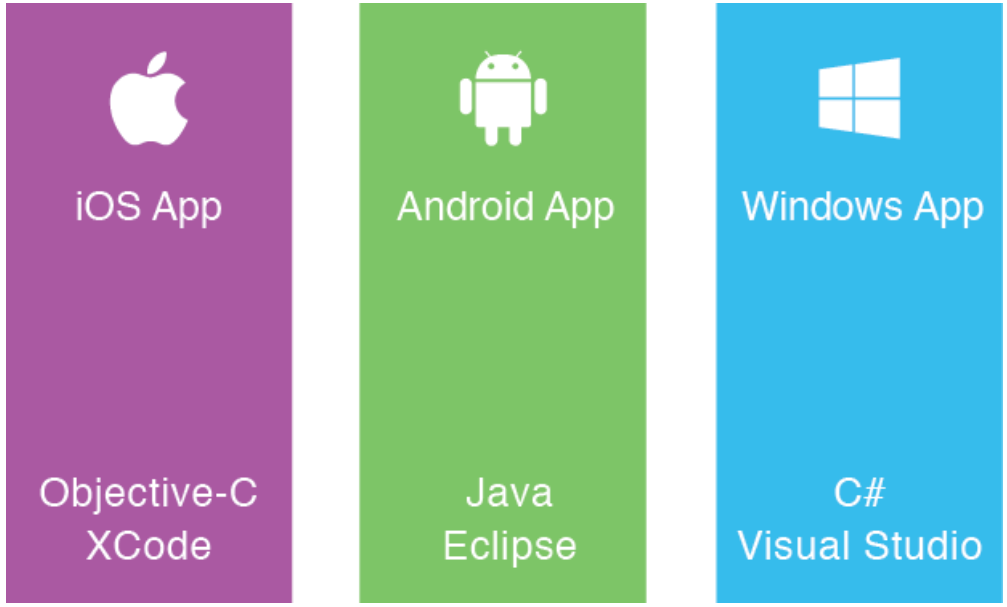


Client Technology Choices



The "Siloed" Approach: Build App Multiple Times

- **Expensive to staff multiple platform-specific teams**
- **Expensive to maintain multiple code bases**
- **Slows innovation**



Silo – Write App on every Target

Benefits

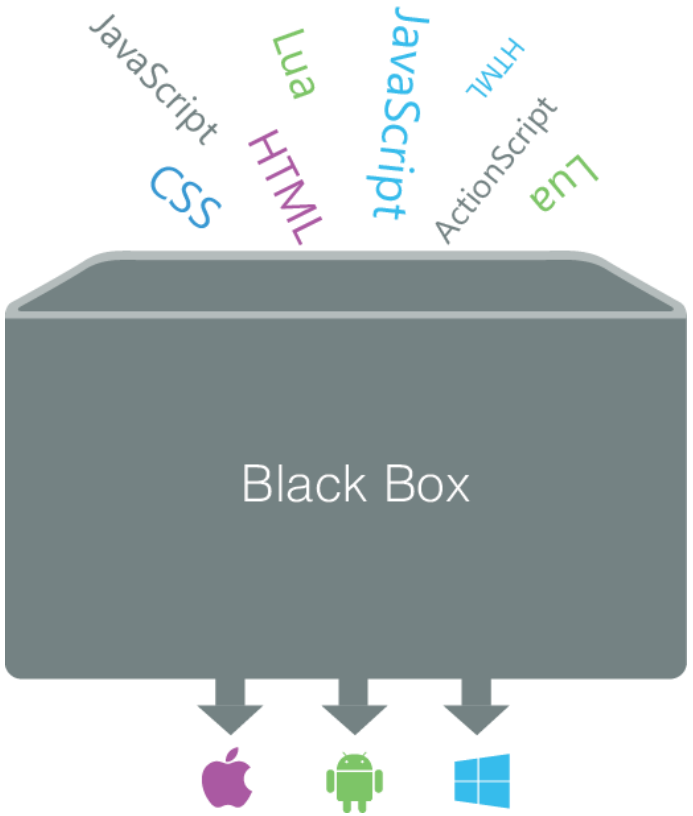
- Full native experience
- Total access to the device as provided by SDK
- Share Web API

Negatives

- Minimal re-use mostly on back end Web API
- Higher development cost from multiple teams (silo teams) or expensive multi-device developers
- Multiple codebases to maintain and extend
- One platform rules the others are subservient

The Write-Once-Run-Anywhere Approach

- **Poor user experience**
  - API coverage
  - Performance
- **High abandonment rates**
- **Wasted investment**



HTML – Write App using Mobile Web

Benefits

- Provide consist experience regardless of target
- Cheap as it is just HTML
- Single codebase to maintain and extend
- No need for revenue sharing as no need to be in app stores

Negatives

- User experience tends to be webish and not native
- Need to still test and debug multiple targets
- Features tend to be a subset common to all targets

MEAP – Write App using Mobile Enterprise Application Platform

Benefits

- Provide consist experience regardless of target
- Cheaper as App is developed once for all targets
- Single codebase to maintain and extend
- Apps can be in app store if needed

Negatives

- User experience tends to be webish and not native
- Need to still test and debug multiple targets even when MEAP only thing updated
- Features tend to be a subset common to all targets
- Vendor risk and lock in

MDAP – Write App using Mobile Development Application Platform

Strategies:

- Tool generating target app
- Write app in single language and compile multiple targets

MDAP – Write App using Mobile Development Application Platform

Benefits

- Provide consist experience regardless of target
- Single codebase to maintain and extend
- Hit a lot of targets at once

Negatives

- Need to still test and debug multiple targets even when MEAP only thing updated
- Features tend to be a subset common to all targets
- Vendor risk and lock in
- May have to wait on new targets

MDAP – Write App using Mobile Development Application Platform

- Tools
  - Appcelerator
  - Embarcadero
  - Rhomobile
  - RubyMotion
  - **Unity**
  - **Xamarin**
  - **Ionic**

Native Mobile Apps

When To

- High Performance Apps
- Heavy on OS and Device Features
- Complex N/W comm. Canvas based Apps
- Only Few Platforms

When Not To

- Performance is not the main criteria
- More or less Replicates Web Apps with few device feature
- Standard Restful
- Widget based apps
- Many Platforms

Cross Platform Mobile Apps

When To

- Performance is not the main criteria
- More or less Replicates Web Apps with few device feature
- Standard Restful
- Widget based apps
- Many Platforms

When Not To

- High Performance Apps
- Heavy on OS and Device Features
- Complex N/W comm.
- Canvas based Apps
- Only Few Platforms

Cross Platform Mobile Apps

When To

- Time to market is critical
- Saving Cost is critical

When Not To

Hybrid Mobile Apps

When To

- Fairly Simple UI
- Complex Backend
- Quite few platforms
- E.g ShareFile

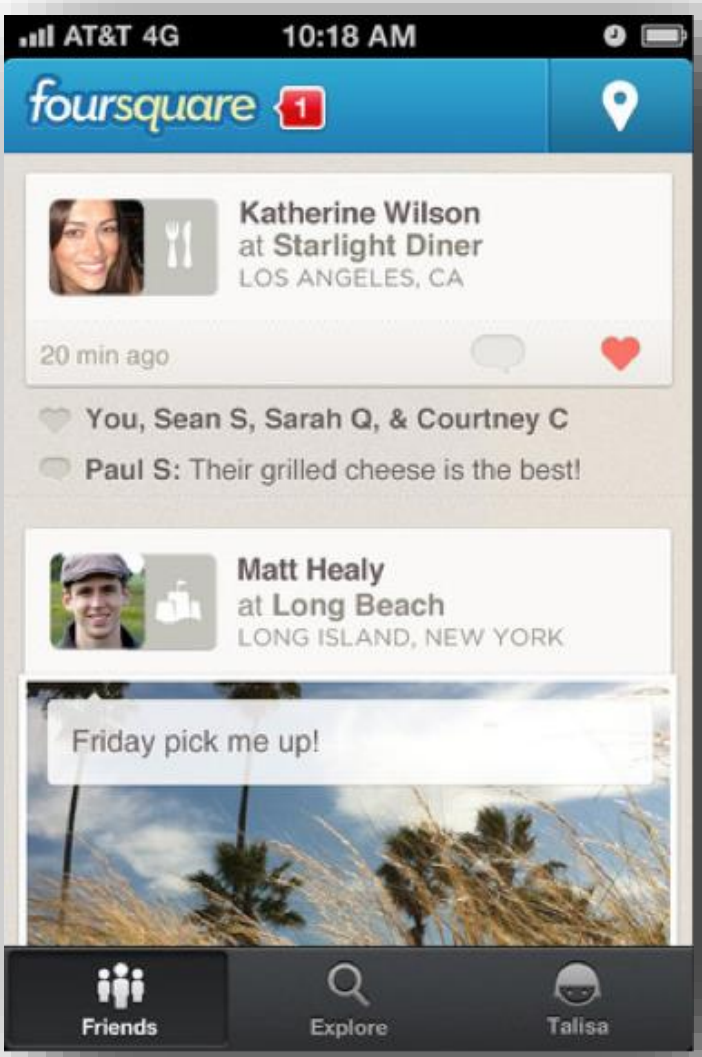
Why To

- Some parts of app are common
- Rest parts are different
- Use Cross Platform to develop common part
- Use Native to develop the weight lifting parts

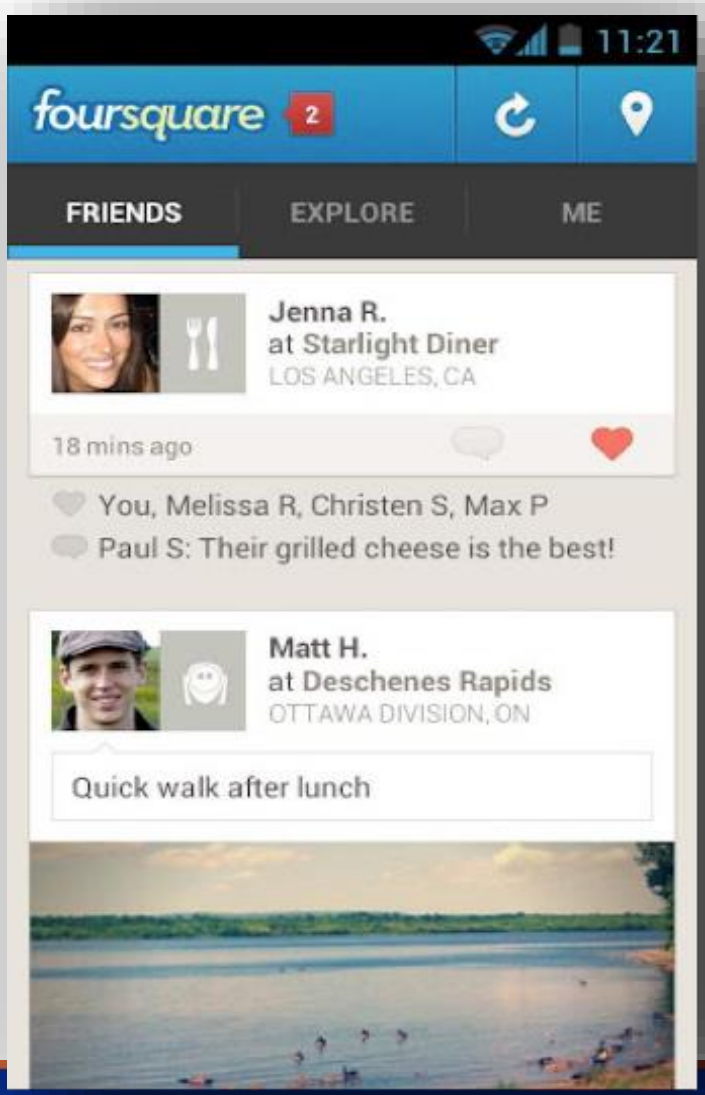
User Interface design considerations

UI DESIGN CONSIDERATION

iOS

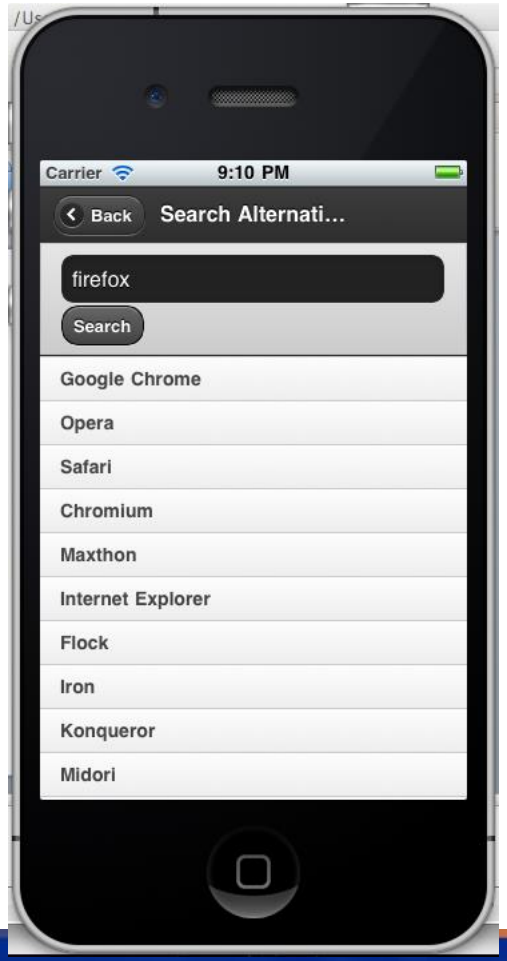


Android

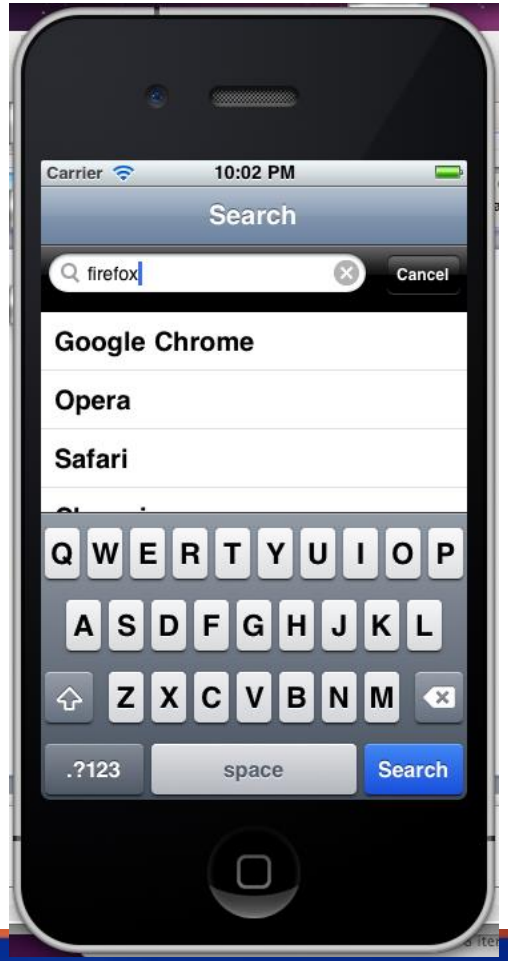


Compare Screens (iPhone)

PhoneGap

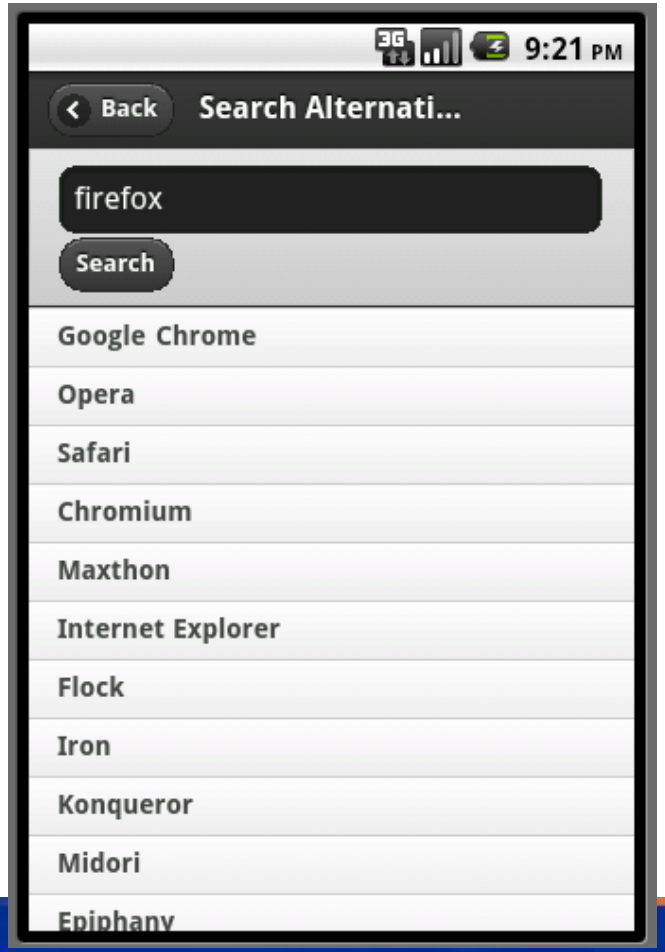


Titanium Mobile

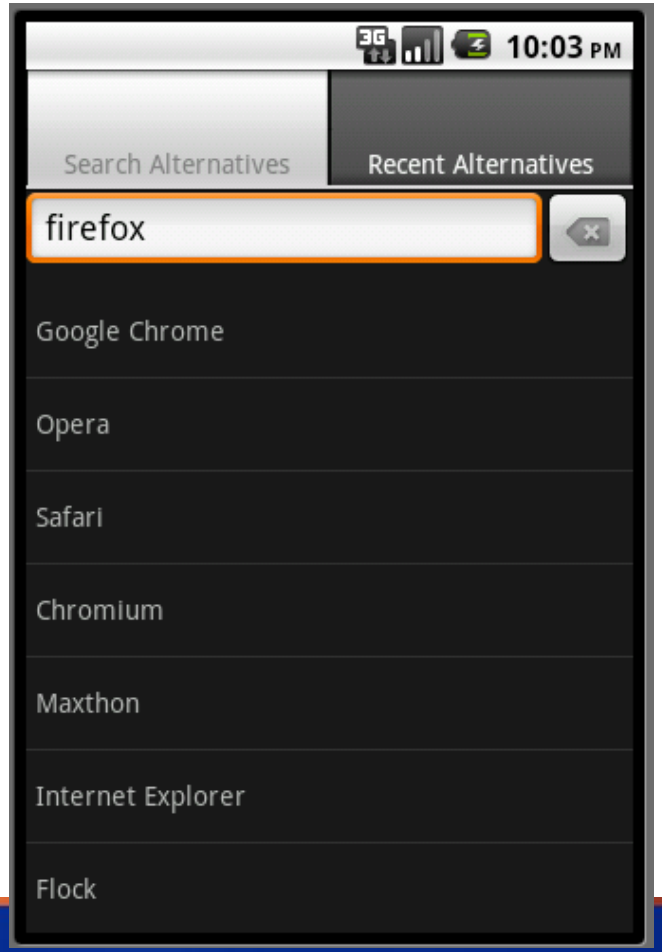


Compare Screens (Android)

PhoneGap



Titanium Mobile



Cross-platform applications developemnt frameworks

Concept

- **Common API** set is provided by framework
- Application is written using this common api set
- HTML/CSS may be supported or may not be supported. Titanium prefers native UI instead of HTML/CSS UI, Rhodes prefers HTML/CSS UI
- Phone Features are access liked common api set (this is similar to that in PhoneGap)

CROSS-PLATFORM FRAMEWORKS

PROS	CONS
Code Reusability	Might not support every feature of OS
Plugins	Cannot use own tools/IDE
Easy for web developers	Slower.
Reduced development costs	High end graphics & 3D support limited
Support for enterprise & cloud services	Vendor lock-in
Easy Deployment	

CROSS-PLATFORM FRAMEWORKS

RhoElements – RhoMobile Suite From Motorola Solutions

TECHNICAL ARCHITECTURE:

- Cross compilation using Virtual Machine.
- Single source codebase written in Ruby and UI constructed using HTML 5, CSS 3, JavaScript running on Ruby interpreter on the device.
- Support for SQLite enables the local storage of relational data, enabling offline capabilities for both hybrid and native HTML 5 applications.

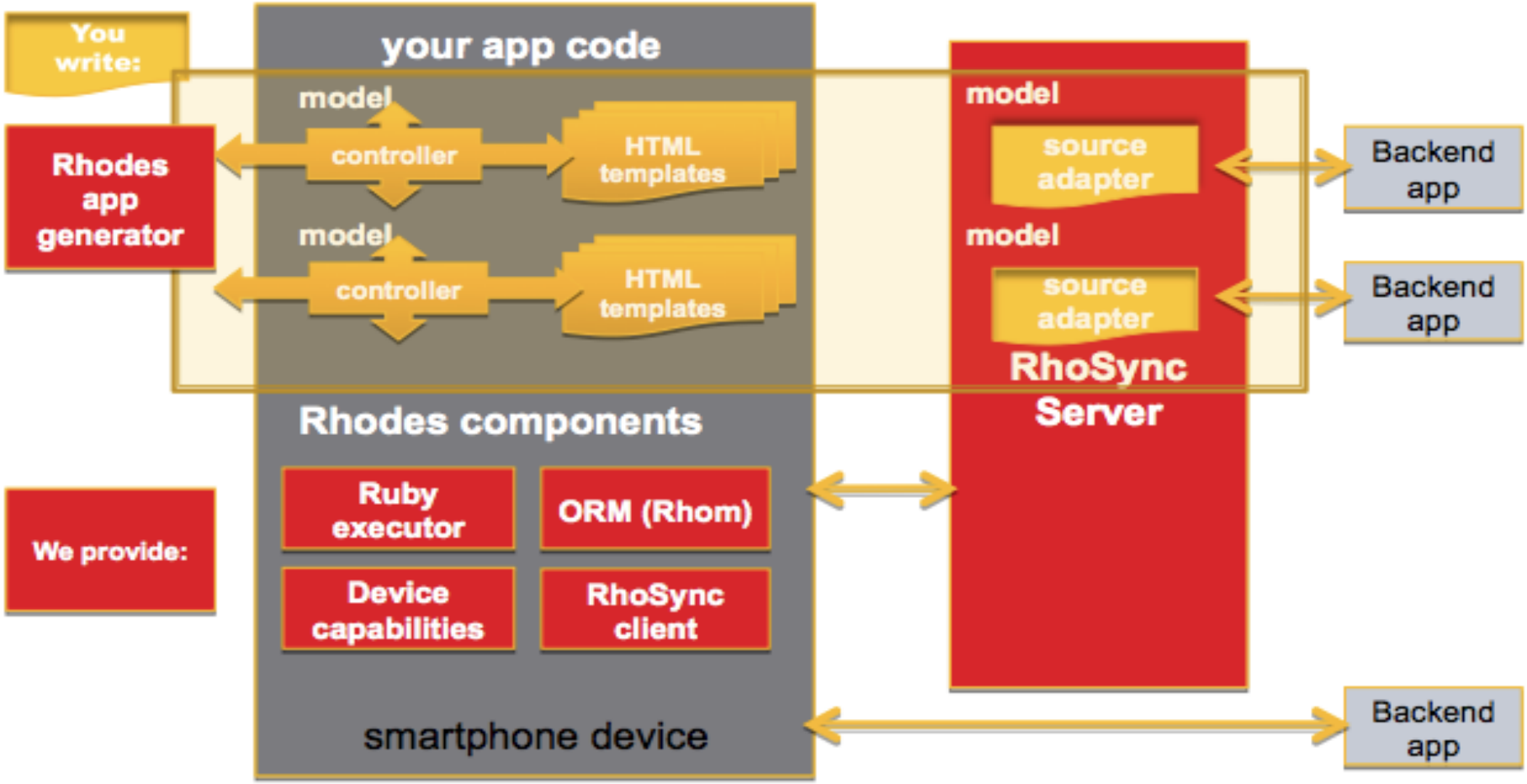
DESIGN PATTERNS:

- Model-View-Controller pattern for maintainability and best practices.
- Object Relational Mapper design for easy data manipulation.

SUPPORTED PLATFORMS:

- WM /WEHH , WinCE5.0+, Android 2.1+, iOS 3.0+, BB 4.6+, WP7+

Rhodes Architecture



RhoElements – RhoMobile Suite From Motorola Solutions

HTML 5 FEATURES:

- App Caching, WebSockets, WebWorkers, Local & Session Storage, SQLite, Semantic Elements, Form Attributes

IDE USED:

- RhoStudio – An Eclipse based IDE

STRENGTHS:

- Design patterns used.
- Applications look and behave identically on all devices.

WEAKNESSES:

- Updating HTML/JavaScript code needs a complete rebuild.
- Need to know Ruby well, which is not as popular as other programming languages.
- Doesn't generate source code, only native package which can restrict any further tweaking of the app.

RhoElements – RhoMobile Suite From Motorola Solutions

Score (out of 3)

Category	Score	Details
Device Compatibility	3	Supports most mobile platforms including iOS, Android, and BlackBerry.
Native UI Components	1	Its easy to get some native looking elements, but actually implementing the native elements takes extra effort.*
Access of Device Features	3	http://docs.rhomobile.com/rhodes/device-caps
General Performance	2	Suffers from an occasional view flicker or white screen.
Community	2	Pretty active Google Group but not a lot of activity on Twitter.
Documentation	1	The documentation, while existant, feels very disorganized.
Sample Code	2	Code samples embedded within documentation; good, clean samples, but good luck finding them.
Data Handling	3	Only cross-platform framework with full support for an MVC.
Animation	1	Really doesn't handle animation; need to use JavaScript for any animation.
View Handling	3	The MVC structure makes building/managing views a breeze.

PHONEGAP (ADOBE)

TECHNICAL ARCHITECTURE:

- Web approach using hybrid model.
- Single source codebase written HTML 5, CSS 3, JavaScript running on a mobile browser embedded in a native app wrapper.
- Device capabilities accessed through device-independent JavaScript API.

SUPPORTED PLATFORMS:

- iOS, Android, Blackberry, WP7, Symbian, Palm, Samsung Bada.

IDE USED:

- MAC OS X & XCODE for iPhone & iPad.
- Google Android SDK, Eclipse ADT Plugin, Ant as well as Eclipse IDE for Android.

PHONEGAP (ADOBE)

ARCHITECTURE:

PHONEGAP (ADOBE)

STRENGTHS:

- Native wrapper source code is provided so it can be customized further.
- Simple ‘drop-in libraries’ concept makes it easier to develop.
- Lowers barriers of adoption for web developers.

WEAKNESSES:

- Lack of support for native UI components, design patterns & development tools.
- The capabilities offered by the framework is limited to what a “WebView” can do.
- Different projects for different platforms
- Different JavaScript files on each platform for PhoneGap itself and plugins
- No native UI support
- Java, Objective-C or C# requirement to create new plugins
- No built-in support for push notifications

PHONEGAP (ADOBE)

Category	Score	Details
Device Compatibility	3	Supports most common OSes including iOS, Android, and BlackBerry.
Native UI Components	0	No native UI support. There are forks that do offer some support, however.
Access of Device Features	3	JavaScript provides great abstraction class for all common device functionality.
General Performance	3	PhoneGap itself performs great; performance issues arise from poorly written apps and slow devices.
Community	3	Very vibrant community; lots of activity on forums, Twitter, and blog articles.
Documentation	2	API reference has gotten a lot better; could still stand to clean up wiki.
Sample Code	2	Good sample code for PhoneGap API, but not a lot of support from PhoneGap for building good mobile apps. However, there are plenty of blog articles.
Data Handling	1	Left completely up to JavaScript and device's implementation.
Animation	1	CSS animation works great on iOS devices; leaves a lot to be desired elsewhere.
View Handling	0	Completely in the hands of the developer how the app is going to manage views.

TITANIUM FROM APPCELERATOR INC.

TECHNICAL ARCHITECTURE:

- Cross compilation technique – Pre-compilation, front-end compilation, platform & package compilation.
- Single source codebase written in JavaScript, compiled into native code and packaged for different target platforms.
- Does not use browser engine to render user interface on mobile devices.
- Instead the UI elements are converted to true native UI elements when deployed to the phone.

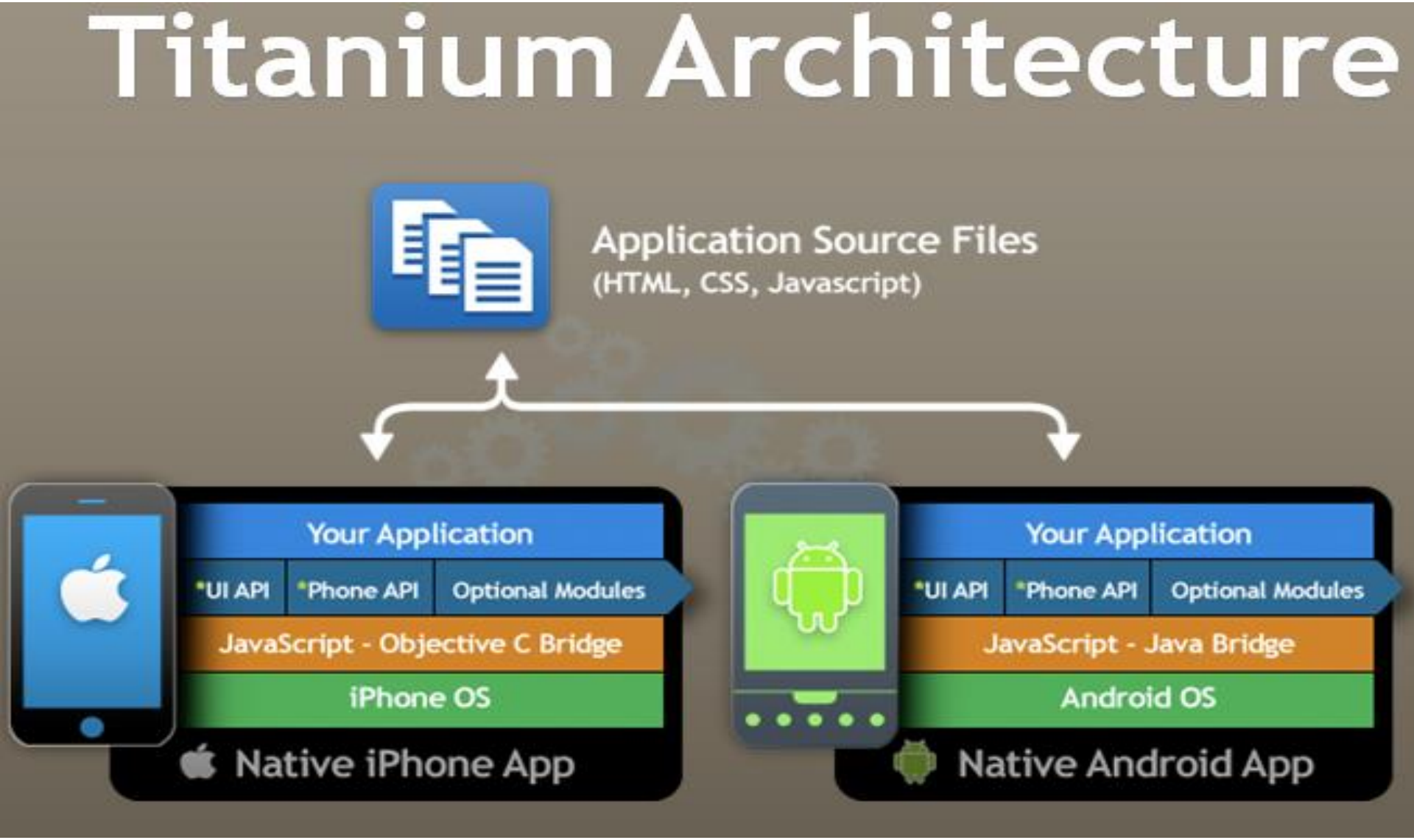
SUPPORTED PLATFORMS:

- iOS, Android, Windows & Blackberry

IDE USED:

- Studio, an Eclipse-based IDE

Titanium Architecture



TITANIUM FROM APPCELERATOR INC.

STRENGTHS:

- Native code output very quick and fluid on the phone.
- Easy setup and startup for developers.
- Excellent documentation & examples.
- Strong community forum to find out answers.
- Intuitive app management environment.
- Support for desktop and tablet development

WEAKNESSES:

- Potentially restrictive API's
- Tries to solve too many problems in one shot supporting phones, tablets & desktops.

TITANIUM FROM APPCELERATOR INC.

Category	Score	Details
Device Compatibility	1	Only works with Android 2 and iOS; doesn't work in Honeycomb. Support for BlackBerry in beta.
Native UI Components	3	Supports nearly every native device UI component.
Access of Device Features	3	Provides JavaScript abstraction for all the common features; includes some lower level network control.
General Performance	2	Occasionally suffers from blank views while loading.
Community	2	Active community (although a handful of questions go unanswered).
Documentation	3	Great API documentation.
Sample Code	3	The Kitchen Sink app is a great example of all the features of Titanium.
Data Handling	2	Easily parse through JSON and XML support is pretty good; easy to build views based on data.
Animation	2	Can animate most UI elements, but don't expect very advanced animations.
View Handling	3	Effortlessly manage and customize different views of the application; each window can have its own namespace.

MoSync From MoSync AB

TECHNICAL ARCHITECTURE:

- Cross compilation using Virtual Machine.
- Single source codebase written in C/C++ or HTML/JavaScript or a combination of both.
- C++ source code → platform-independent intermediate code → application package

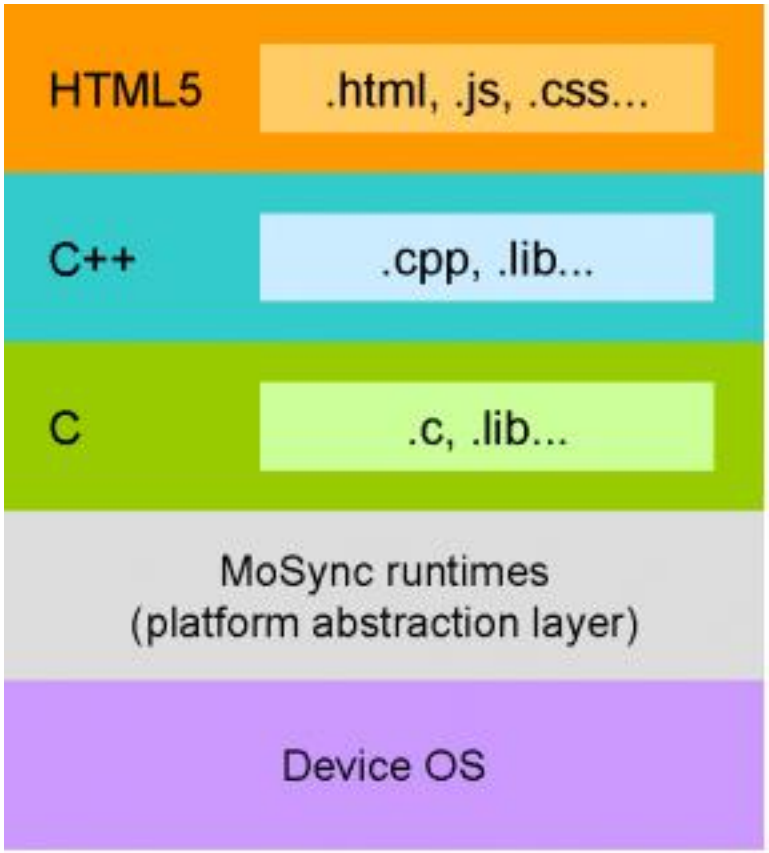
SUPPORTED PLATFORMS:

- iOS, Android, Windows Mobile, Moblin/MeeGo, Symbian & Blackberry

IDE USED:

- MoSync IDE based on Eclipse.

MoSync From MoSync AB



MoSync mobile App layers

MoSync From MoSync AB

STRENGTHS:

- Only one project structure for all the platforms.
- The same JavaScript file.
- Extend JavaScript functionality using C++ or Java and Objective-C
- Native UI support
- Built-in support for push notifications
- Target group: Both web developers looking to enter the mobile space, as well as the ordinary PC/Mac desktop developer with knowledge of C/C++.

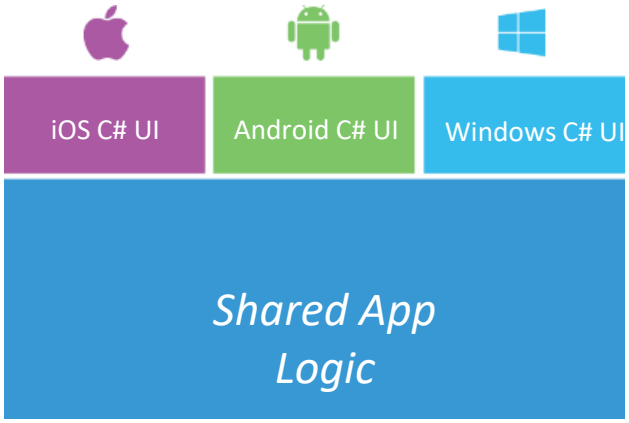
WEAKNESSES:

- No support for accelerometer or camera in most phones.
- Contains XML parsing libraries but lacking support for JSON or other data formats.
- Doesn’t provide support for MVC; requires little extra effort to create views for data.

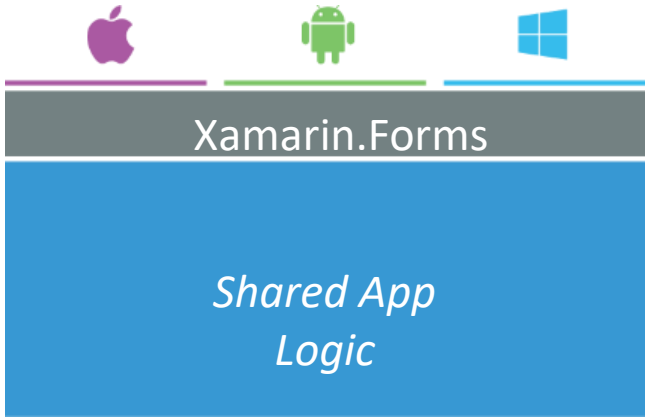
MoSync From MoSync AB

Category	Score	Details
Device Compatibility	2	Doesn't have full support for BlackBerry; iPhone support is still limited in some regards.
Native UI Components	1	Only supports iPhone and Android; doesn't work in MoSync emulator.
Access of Device Features	1	Supports some lower level network control, but no support for accelerometer or camera in most phones.
General Performance	3	Runs smoothly; get a lot of control over how fonts are rendered to the screen.
Community	1	Hardly any Twitter activity; a lot of registered users in forums, but not a lot of posts.
Documentation	3	Lots of documentation about framework and an excellent API reference.
Sample Code	2	Provides a decent amount of sample code; could really benefit from a "Kitchen Sink" type app.
Data Handling	1	Contains XML parsing libraries, but lacking support for JSON or other data formats.
Animation	2	There are plans to include support for OpenGL; because its written in C, there is support for some drawing and simple physics libraries.
View Handling	1	Doesn't provide support for an MVC; requires a little extra effort to create views for data.

Xamarin Approach

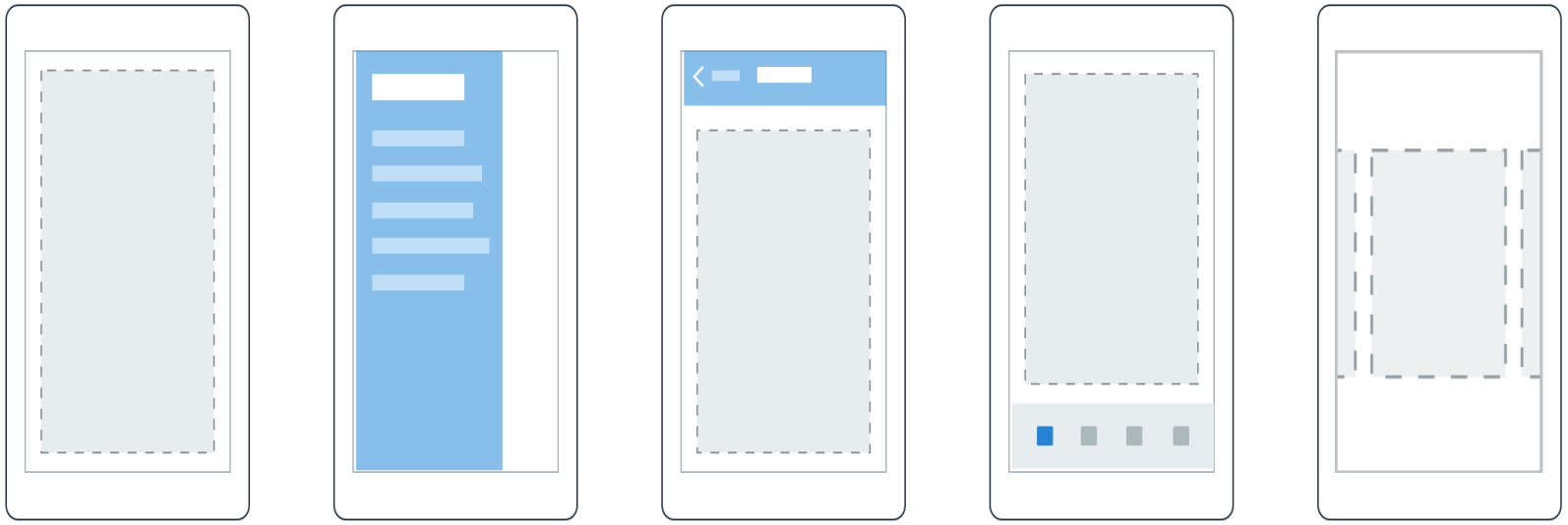


Traditional



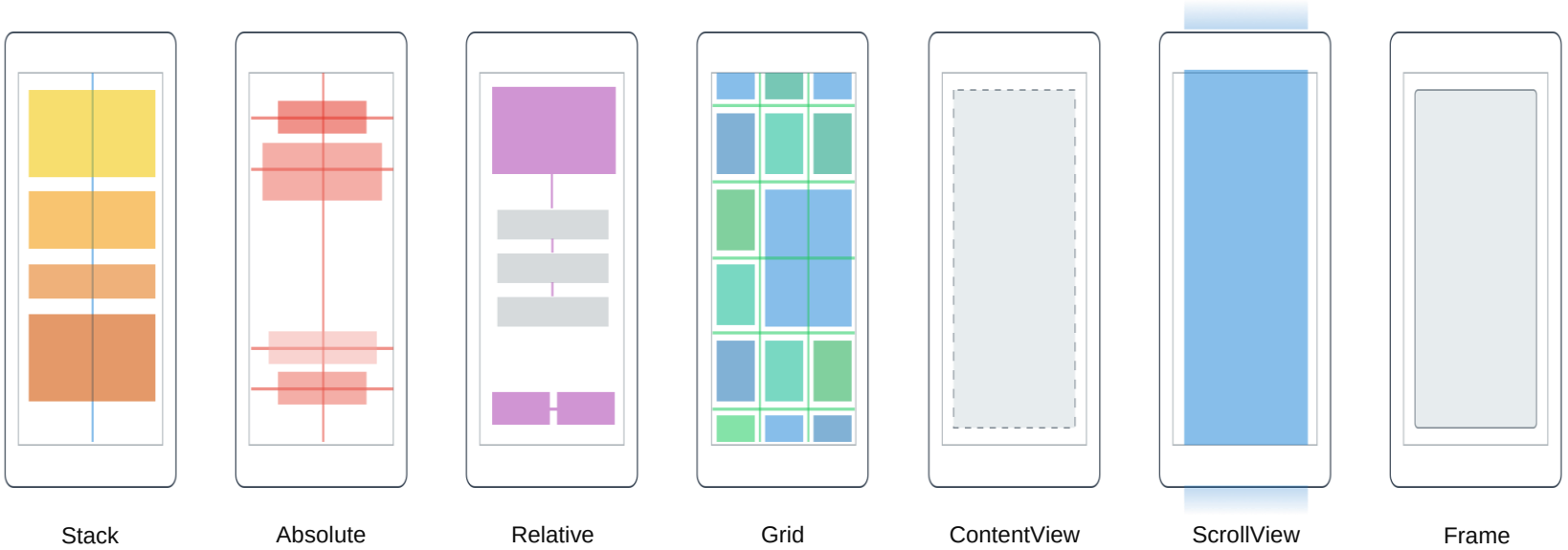
Xamarin.Forms

Xamarin - Pages



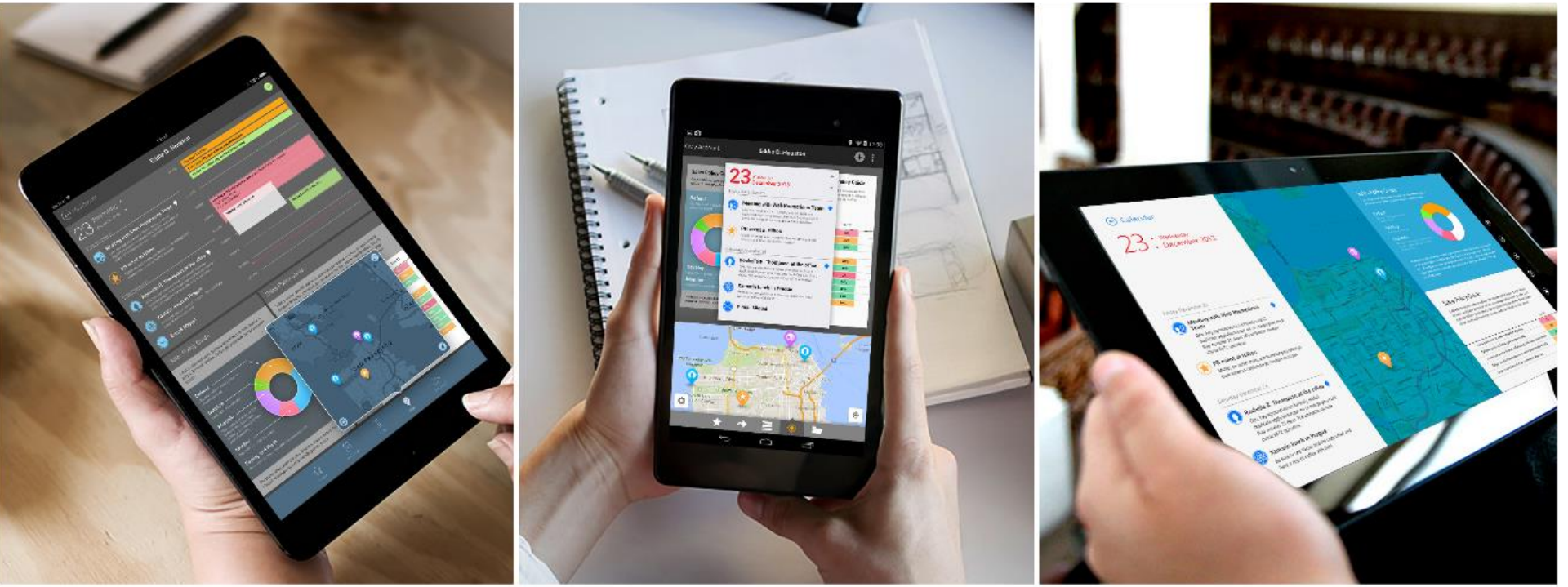
Content MasterDetail Navigation Tabbed Carousel

XAMARIN - LAYOUTS



Xamarin - Controls

ActivityIndicator	BoxView	Button	DatePicker	Editor
Entry	Image	Label	ListView	Map
OpenGLView	Picker	ProgressBar	SearchBar	Slider
Stepper	TableView	TimePicker	WebView	EntryCell
ImageCell	SwitchCell	TextCell	ViewCell	



Xamarin exposes 100% of the native APIs for iOS, Android and Windows

Xamarin - Native Performance

Xamarin.iOS does full Ahead Of Time (AOT) compilation to produce an ARM binary for Apple's App Store.

Xamarin.Android takes advantage of Just In Time (JIT) compilation on the Android device.

Xamarin - Suported Native APIs

iOS 5 … iOS10

… Android 6 / API level 23

… Windows 10

Also:

- Google Glass
- Android Wear
- Amazon Fire TV
- Outros…

Xamarin

Benefits:

- Re-use .NET skills
- Leverage existing .NET technology
  - JSON.NET
  - OAUTH.NET
  - SignalR
- High code re-use 80+%
- Tailor UI/UX to target

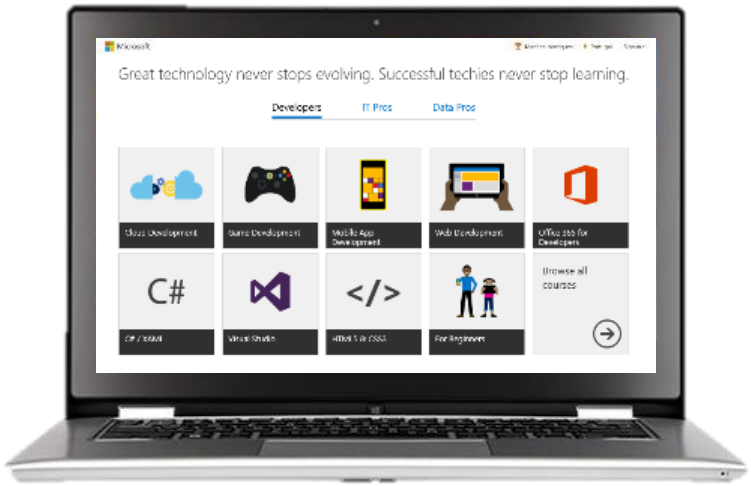
Negatives:

- Need to still test and debug multiple targets even when MEAP only thing updated
- Multiple codebase for UI
- No sharing of UI
- Vendor risk and lock in although Xamarin is a strategic partner for MS
- May have to wait on new targets like Android

MVA

Microsoft Virtual Academy

www.microsoftvirtualacademy.com



Cross-Platform Development with Xamarin & Visual Studio

Cross-Platform Development with Visual Studio

RHODES

- Developed by Motorola.
- Native app like feel.
- Apps written in Ruby and recently extended for JavaScript
- Support to build Android, iOS, Blackberry Apps, Windows phone and Mobile.
- Source code organization
- Device Specific Functionality – No built in support for Bluetooth and NFC.
- Rich web service support built in.
- Free but not for commercial users.
- RhoHub is their MBaaS

CORONA

- Developed by Corona Labs.
- Apps written in Lua.
- Free until app isn't published.
- Support to build Android, iOS, NOOK and Kindle Fire Applications.
- Application is compiled using Lua libraries mashed with OpenGL and OpenAL.
- Native controls using underlying library
- Device Specific Functionality – No built in support for Bluetooth and NFC.
- Web services – HTTP,HTTPS,SOAP, JSON
- Cloud service is called Corona Cloud.
- Targeted for game developers.

Marmalade

- Developed by Ideaworks3d.
- Upfront licensing.
- Apps written in C++
- Support to build iOS, Android, BlackBerry PlayBook OS, and bada.
- Binary combined with Segundo Embedded Execution Environment (S3E)
- All device specific functionality except Bluetooth and NFC.
- Web services – SOAP, XML, JSON
- Marmalade Juice – plan to port Objective C source code into Marmalade.

MonoCross

- Part of the Mono Project
- Built on the .NET framework.
- C# is used to build apps.
- Support to build Android and iOS.
- Specific platform tools – Xamarin Mono and Xamarin MonoTouch.
- Interpreter for Android and BlackBerry is MozillaRhino, for iOS JavascriptCore
- Native experience – ‘not quite there’
- Source code organization
- Device Specific Functionality – No built in support for Bluetooth and NFC.
- Windows Communication Foundation– Bing Maps API

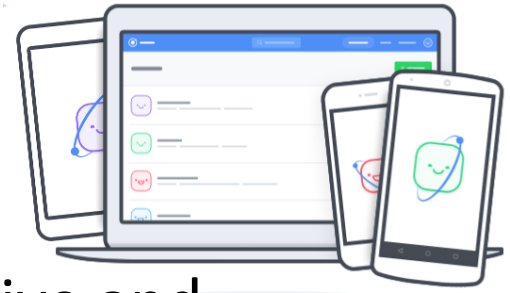
Sencha Touch

- Over 50 built-in components.
- Built-in MVC system
- Apps written in HML5 and CSS3.
- Sencha Touch 2.2 is the latest version
- Faster, Cheaper and highly customizable
- PC developers can now create iOS applications without needing a Mac.
- More than 500,000 developers
- Rich set of documentation

jQuery Mobile

- Built on the rock-solid jQuery and jQuery UI foundation
- Its lightweight size makes it a speed freak
- JQuery Mobile 1.3.1 recently launched
- AJAX-powered navigation system
- Extensions are easy to make
- No established architecture
- Easy to debug
- Markup-based and is backed by a smart community

Ionic



- Open source mobile SDK for developing native and progressive web apps.
- Uses Angular 2 for data and actions binding and Sass for CSS generation.
- Uses Apache Cordova for programming mobile devices native functionalities.
- Enterprise commercial technologies (eg. notifications, cloud starage) offered.
- Version 2.0 uses own, not HTML notation for designing interface.
- Dedicated web-based tool (Ionic Creator) apps developement.

Ionic

- Web apps have adapted interfaces accordingly to the platform.