Mobile Application Design

Ferruccio Damiani

Università di Torino
www.di.unito.it/~damiani

Mobile Device Programming
(Laurea Magistrale in Informatica, a.a. 2018-2019)

Outline

Outline

Mobile is different

[http://www.creativebloq.com/mobile/10-principles-mobile-interface-design-4122910]

Mobile devices are *not* underpowered versions of 'real' computers:

- Smartphones and desktop computers are very different
- Smartphones are actually more complex than desktops in many ways

Mobile is different

[http://www.creativebloq.com/mobile/10-principles-mobile-interface-design-4122910]

Mobile devices are *not* underpowered versions of 'real' computers:

- Smartphones and desktop computers are very different
- Smartphones are actually more complex than desktops in many ways

Designing for mobile is very different than designing for the desktop [1]

- There are many more issues

Mobile is different

[http://www.creativebloq.com/mobile/10-principles-mobile-interface-design-4122910]

Mobile devices are *not* underpowered versions of 'real' computers:

- Smartphones and desktop computers are very different
- Smartphones are actually more complex than desktops in many ways

Designing for mobile is very different than designing for the desktop [1]

- There are many more issues

1: What about designing for tablet vs smartphone vs smartwatch?

Mobile application design

- A mobile app should do one thing and do it well
- A mobile app should be as simple as possible, but not simpler
  - Minimalist design (e.g., no quit button)
  - Usability design
  - Standards compliance
  - ...
- Different versions (families of applications)
- ...

Some Key choices

1. What is the targeted set of users?
2. Device(s)
3. Operating system(s)
4. Legacy components
5. Deployment/distribution model
6. Complete process

1. What is the targeted set of users?

- Controlled set of users
  - Employees
  - Customers
- Open set of users
  - Market

2. Device(s)

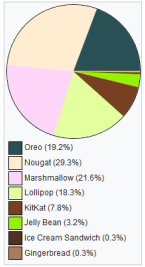
- More then 1000 Android devices
- Other devices

Different versions of Android [https://en.wikipedia.org/wiki/Android_(operating_system)]

Charts in this section provide breakdowns of Android versions, based on devices accessing the Google Play Store in a seven-day period ending on September 28, 2018.[359][lx] Therefore, these statistics exclude devices running various Android forks that do not access the Google Play Store, such as Amazon's Fire tablets.

Version	Code name	Release date	API level	Runtime	Distribution	First devices to run version
9.0	Pie	August 6, 2018	28	ART	N/A	Essential Phone, Pixel, Pixel XL, Pixel 2, Pixel 2 XL, Nokia 7 Plus, OnePlus 6, Oppo R15 Pro, Sony Xperia XZ2, Vivo X21UD, Vivo X21, Xiaomi Mi Mix 2S [360]
8.1	Oreo	December 5, 2017	27	ART	5.8%	Pixel, Pixel XL, Nexus 6P, Nexus 5X
8.0		August 21, 2017	26	ART	13.4%	N/A
7.1	Nougat	October 4, 2016	25	ART	10.3%	Pixel, Pixel XL
7.0		August 22, 2016	24	ART	19.0%	Nexus 5X, Nexus 6P
6.0	Marshmallow	October 5, 2015	23	ART	21.6%	
5.1	Lollipop	March 9, 2015	22	ART	14.7%	Android One
5.0		November 3, 2014	21	ART 2.1.0	3.6%	Nexus 6, Nexus 9
4.4	KitKat	October 31, 2013	19	Dalvik (and ART 1.6.0)	7.8%	Nexus 5
4.3	Jelly Bean	July 24, 2013	18	Dalvik	0.5%	Nexus 7 2013
4.2		November 13, 2012	17	Dalvik	1.6%	Nexus 4, Nexus 10
4.1		July 9, 2012	16	Dalvik	1.1%	Nexus 7
4.0	Ice Cream Sandwich	October 19, 2011	15	Dalvik	0.3%	Galaxy Nexus
2.3	Gingerbread	February 9, 2011	10	Dalvik 1.4.0	0.3%	Nexus S

Legend: Old version | Older version, still supported | **Latest version** | Latest preview version | Future release



See also [https://developer.android.com/about/dashboards/index.html]

Different versions of iOS [https://en.wikipedia.org/wiki/IOS_version_history]

Version	Build	Processor support	Application support	Kernel	Release date	Device end-of-life		
						iPad	iPhone	iPod Touch
3.1.3	7E18	32-bit ARM			Feb 2, 2010	N/A	1st gen	1
4.2.1	8C148				Nov 22, 2010		3G	2
5.1.1	9B206				May 7, 2012	1st gen	N/A	3
6.1.6	10B500				Feb 21, 2014	N/A	3GS	4
7.1.2	11D257	32/64-bit ARM[1][2]			Jun 30, 2014		4	N/A
9.3.5	13G36				Aug 25, 2016	2, 3, Mini 1	4S	5
10.3.3	14G60				Jul 19, 2017	4	5, 5C	N/A
12.0.1	16A404/16A405	64-bit ARM[3]			Oct 8, 2018	N/A		
12.1 Beta 3	16B5077c				Oct 9, 2018			
Legend: Discontinued Current Beta								

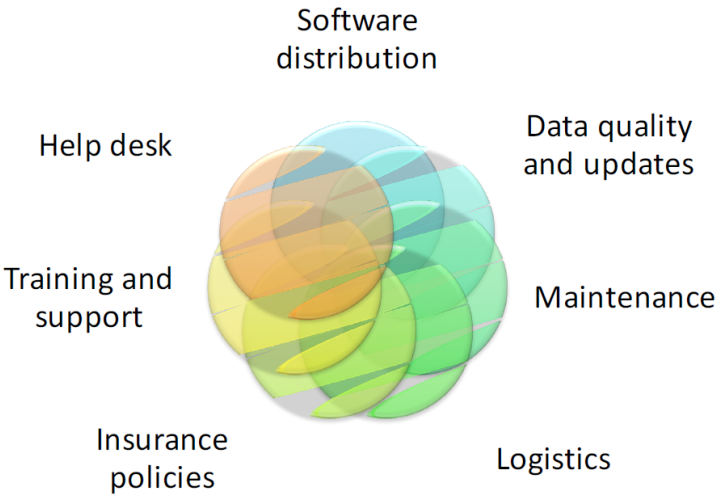
and some debatable choices—see, e.g., Apple's "A Message to Our Customers about iPhone Batteries and Performance" (December 28, 2017) [https://www.apple.com/iphone-battery-and-performance/]

- Where do we came from?
- Where do we want to go?
- What assets do we want to keep?
- Which systems are we supposed to integrate?

5. Deployment/distribution model

- Custom systems
- Closed world
  - Centralized distribution
  - Dedicated sites
- Market place

6. Complete process



Outline

Main development activities

QUESTION: What are the main development activities?

- ....
- ....
- ....

Main development activities

1. Think/Prototype
2. Design
3. Develop

Main development activities

1. Think/Prototype
2. Design
3. Develop

Some key characteristics:

- Mixed teams: designers and not just computer scientists
- Sketches and prototyping

Some guidelines

1. Do one thing, and do it really well
   - Low quality is not allowed
   - Take into account your target(s)
2. Find that "special element"
   - But don't take ages to develop it
   - Always ask: "can I repurpose/re-use this?"
3. Do minimalistic and avoid useless complexity

Mobile Interface Design [http://www.creativebloq.com/mobile/10-principles-mobile-interface-design-4122910]

1. Mobile mindset
   - Focused, unique, charming, user-centered
2. Different classes of users
   - Clearly identify your target(s): bored, busy, lost
3. First impression is key
   - Limited/No help text
   - Characteristic and intriguing look and feel
   - Just a few seconds and the app...

- Be bold
  - Users are captured by unique design
  - Users get tired of seeing the same old thing
  - Do not use Android/Apple-supplied UI elements as a always-good solution
    - ★ They are starting to look dated

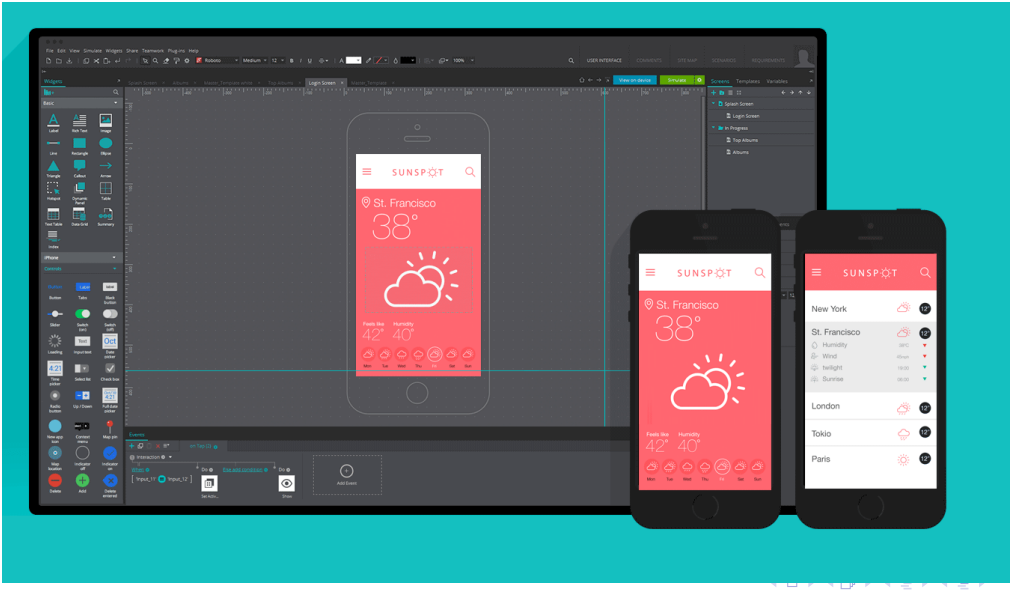
Mobile Interface Design

QUESTION: What are the basic tools for mobile inteface design?

- ....

Sketching with paper, pencil and rubber

Sketching with a tool (like, e.g., [http://http://www.justinmind.com/])



About features

- Users do not spend time discovering features
- Users do not complain about “advanced” features
  - More features imply more apps
- Users complain about features that do not work
- Do users get tired of seeing the same old thing?

Android design principles [http://developer.android.com/design/get-started/principles.html]

- Enchant me
  - Delight me in surprising ways
  - Real objects are more fun than buttons and menus
  - Let me make it mine
- Simply my life
  - Keep it brief
  - Pictures are faster than words
  - Decide for me but let me have the final say
  - I should always know where I am

iOS design suggestions [http://www.fastcodesign.com/3030652/8-tips-from-apples-official-guide-to-app-design]

- Use Layout to Communicate
- Avoid asking people to supply setup information
  - Focus on the needs of 80% of your users
- Launch in the device's current orientation
- When your app restarts, restore its state so users can continue where they left off
- An iOS app never displays a Close or Quit option
  - Never quit an iOS app programmatically

Mobile Interface Design (continues)

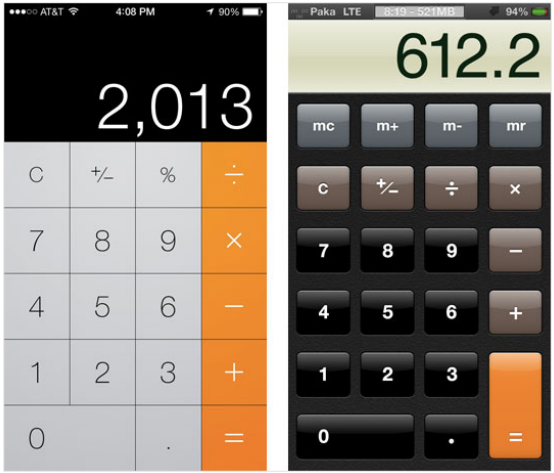
[http://www.creativebloq.com/mobile/10-principles-mobile-interface-design-4122910]

4. Single and appropriate navigation model
5. Minimal user inputs (through the proper means)
   - Auto-correct can be so frustrating
6. Gestures are not really standardized
   - They are nice to have, but not mandatory
7. Support orientations
   - Be consistent and exploit orientation locks
8. Communications
   - Provide polite feedback, modal alerts, confirmations

- Postpone sign up

Flat design [http://www.creativebloq.com/graphic-design/what-flat-design-3132112]

If your app looks outdated, users will note that:



Flat design:

- Not boring
- Ornamental elements are viewed as unnecessary clutter
- Bright, contrasting colors make illustrations and buttons pop from backgrounds
- Minimalistic nature

Flat design doesn't depend on bright colour

Adopt consistent layout

- Can be very “expensive”
- Extremely important
- Design libraries exist to help decide which layout is the best for a particular problem

Avoid anti-patterns

QUESTION: What is an anti-pattern?

Avoid anti-patterns

QUESTION: What is an anti-pattern?

> *An anti-pattern (or antipattern) is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.*[1]

1: Koenig, Andrew. "Patterns and Antipatterns". Journal of Object-Oriented Programming 8 (1): 46–48. March–April 1995.

Avoid anti-patterns

An anti-pattern (or antipattern) is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.[1]

QUESTION: Examples?

1: Koenig, Andrew. "Patterns and Antipatterns". Journal of Object-Oriented Programming 8 (1): 46–48. March–April 1995.

Avoid anti-patterns

An anti-pattern (or antipattern) is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.[1]

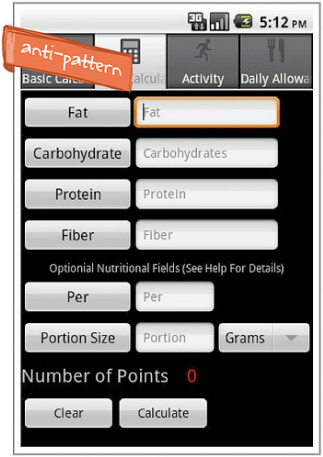
QUESTION: Examples?

Mobile design anti-pattens:

- Metaphor mismatch
  - Control, icon, or mental model mismatch
- Idiot boxes
- Too many chart elements
- Oceans of buttons

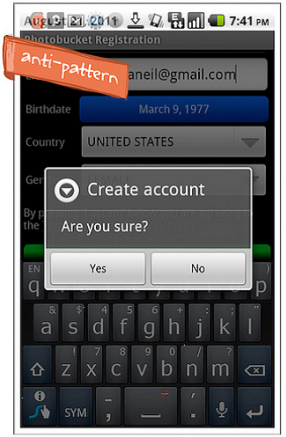
1: Koenig, Andrew. "Patterns and Antipatterns". Journal of Object-Oriented Programming 8 (1): 46–48. March–April 1995.

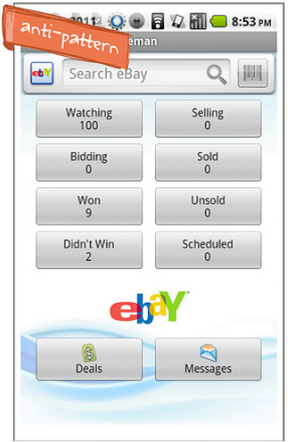
Avoid PCisms (source [http://www.slideshare.net/danhermes/mobile-design-patterns-36205894])



Images courtesy of Mobile Design Pattern Gallery by Theresa Neil
[http://cdn.oreillystatic.com/oreilly/booksamplers/9781449363635_sampler.pdf]

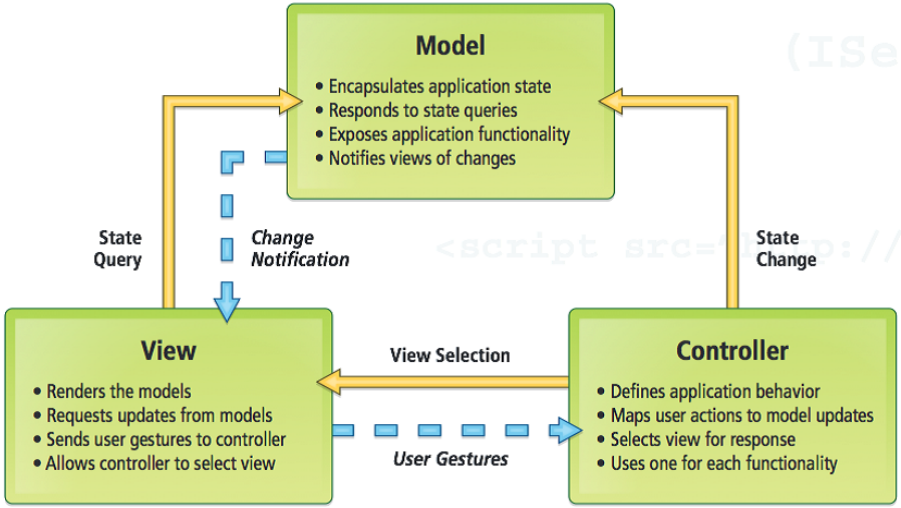
Examples: Idiot boxes, oceans of buttons





Outline

Model-View-Controller



Web-based solutions [1 / 2]

Pros

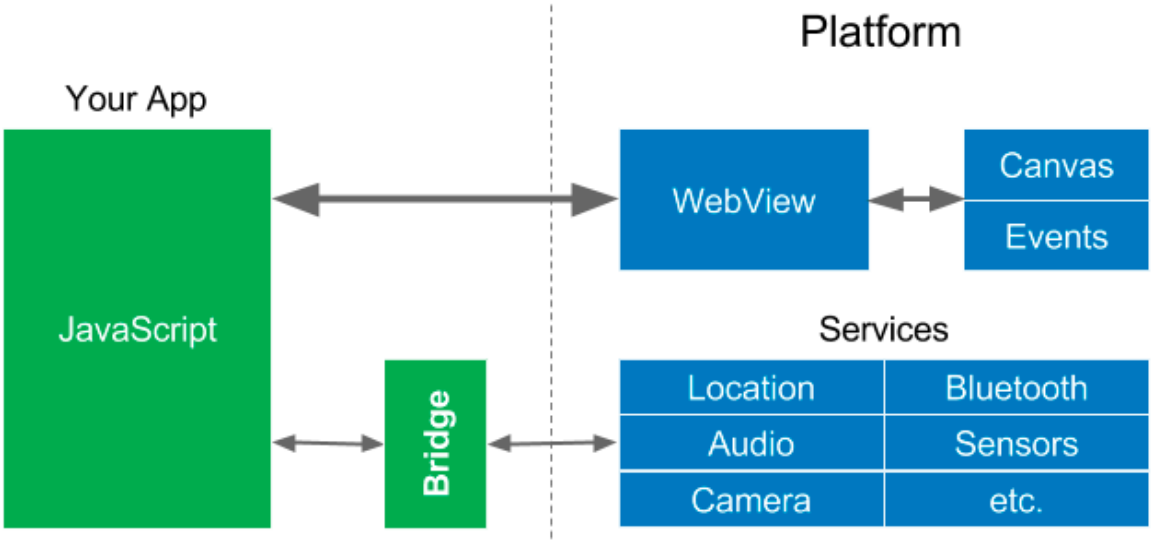
- It is not installed on the device
- Being server-based, it can easily be updated
- The same user experience can be reused on different platforms

Cons

- Being internet-based, performance can be an issue
- The interactions with local software and hardware components is limited
- It is not distributed through a marketplace

Examples: AppsBuilder, iBuildApp, Bootstrap

Web-based solutions [2 / 2]



Hybrid solution [1 / 2]

Hybrid apps are essentially written as a web application (using technology like HTML, JavaScript, and CSS) that's embedded within a "native wrapper" allowing it to run on any device while bypassing the restrictions of a browser-only app functionality (i.e., they can access a device's hardware). Hybrid app frameworks bridge the gap between device and a web app. The developer can use a JavaScript-API to read additional data from the device and trigger actions such as vibration in a standardized way across the different native platforms.

Pros

- The user experience can be based on native elements and be reused
- It can (partially) interact with the hardware components of the device
- It can be distributed through a marketplace

Cons

- Performance can be an issue given the need for an interpreter
- JavaScript might be interpreted differently on different devices
- The user experience is only close to the native one

Examples: Apache Cordova (based on PhoneGap [http://nitobi.createsend.com/t/ViewEmailArchive/y/E306897CE185ABDE/C67FD2F38AC4859C/]).

Hybrid solution [2 / 2]

Interpreted solution [1/ 2]

Interpreted apps use platform-specific native UI elements to interact with the user whereas the application logic is captured in a platform-independent way

Pros

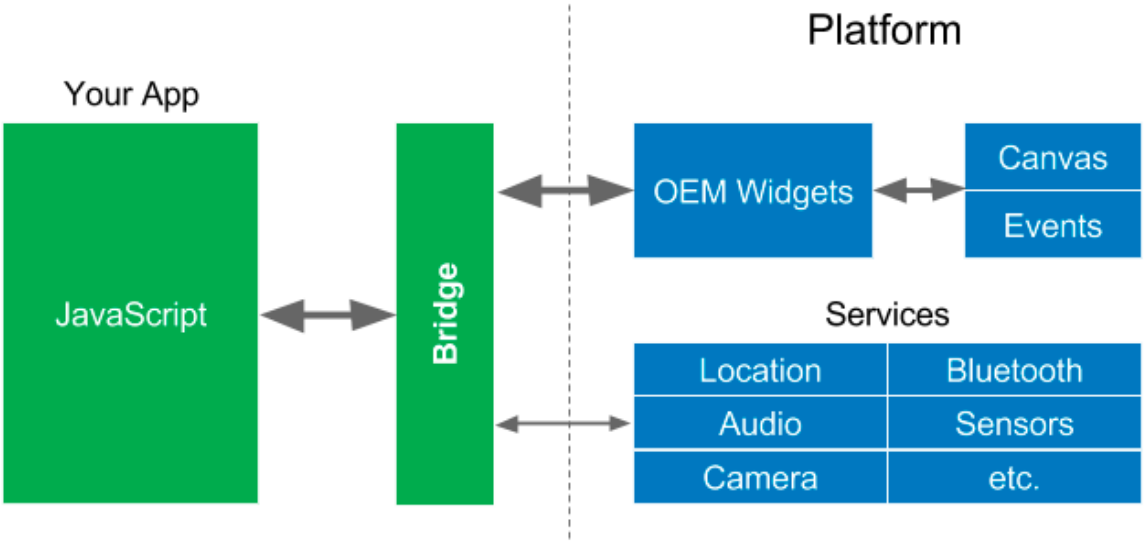
- The user experience corresponds to the (basic) native one
- The business logic can be reused
- It can be distributed through a marketplace

Cons

- Performance can be an issue
- The reuse of the user experience depends on the abstraction level of the framework
- The actual development depends on the specific framework

Examples: Appcelerator Titanium (JavaScript), React Native (JavaScript and React), Native Script (Angular, TypeScript, or JavaScript)

Solutions based on reactive views [1 / 2]



Interpreted Solutions

React Native

- Native mobile apps using JavaScript and React
- Uses the same fundamental UI building blocks as regular iOS and Android apps
- UI building blocks put together using JavaScript and React
- Pushed by Facebook

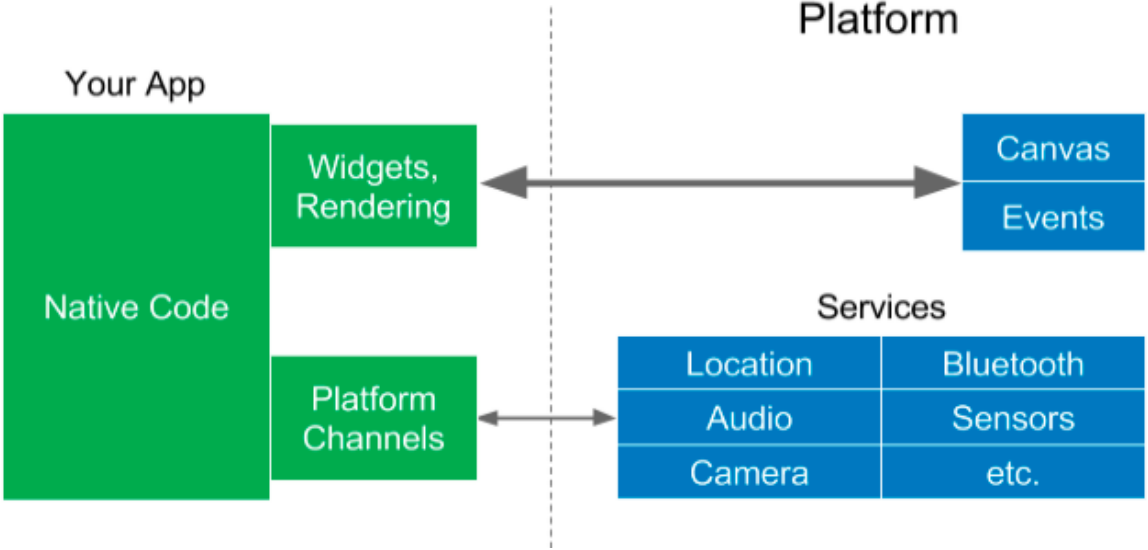
NativeScript

- Native mobile apps with Angular, TypeScript or JavaScript
- Use web skills, like Angular and CSS
- Native UI and performance on iOS and Android
- Hundreds of NativeScript plugins

Flutter [https://flutter.io/] [1 / 2]

- Flutter is a mobile app SDK, complete with framework, engine, widgets, and tools
- Gives developers easy and productive way to build and deploy beautiful apps
- Also used for Fuchsia
- Dart (Flutter's language) can be used to build web and server applications as well
  - Learn Dart once, develop for five platforms

Flutter [https://flutter.io/] [2 / 2]



Cross-compiled solutions [1 / 2]

Native apps that are created by cross-platform development tools. Cross-platform software compiles a single app source code into native code that will run on different operating systems. It's a more native feel than a hybrid app, but you're still only working with one source code.

Pros

- It can offer all the characteristics of a native solution
- Hardware and software components can be exploited
- Performance is usually good

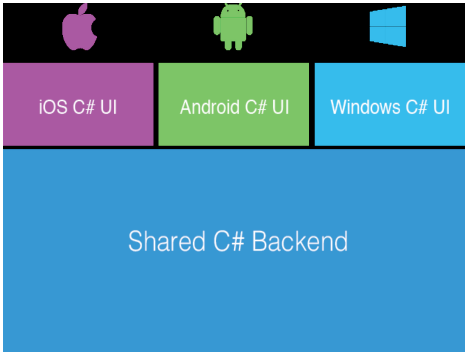
Cons

- The user experience usually cannot be reused
- There could be some limitations in the way hardware components can be used
- The result is usually not too sophisticated

Examples: Xamarin (C#), Corona SDK (Lua)

Cross-compiled solution [2 / 2]

Xamarin



Have also a look at the (recently updated) post at

`http://appindex.com/blog/ten-best-cross-platform-development-mobile-enterprises/`

Native solution [1 / 2]

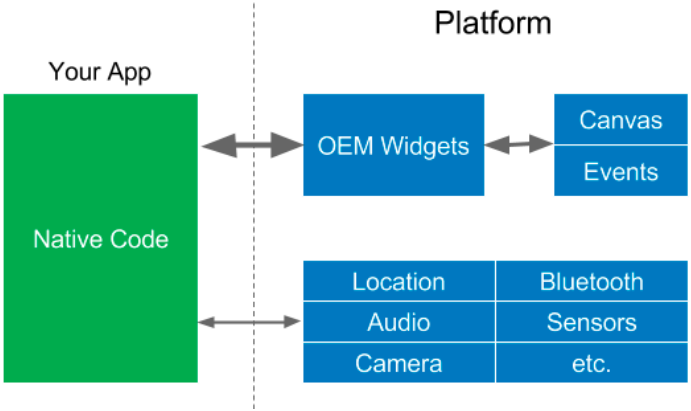
Pros

- It can be efficient and special-purpose
- It can fully exploit any single characteristic
- It can (easily) provide a completely native user experience

Cons

- Development costs tend to become high
- One development for each platform
- Almost no reuse

Native solutions [2 / 2]



Connection types

- **Always connected:** This app requires network availability.

- **Never connected:** This app executes irrespective of network availability and can access device peripherals such as the camera, contacts, email, calendar, GPS, and storage via exposed APIs (front calls). For database needs, this app can only connect to a local database.
- **Partially connected:** This app must be able to run when no network connection is available. This app first operates without a network connection, and must be able to run without a network connection. Once network connectivity is restored, the app can perform network-dependent tasks such as synchronizing with a remote database, make a web service call, or use a web component.

QUESTION: Examples?

Connection types

- **Always connected:** This app requires network availability.
  E.g.: SSH client, Microsoft Remote Desktop,...
- **Never connected:** This app executes irrespective of network availability and can access device peripherals such as the camera, contacts, email, calendar, GPS, and storage via exposed APIs (front calls). For database needs, this app can only connect to a local database. E.g.: ???
- **Partially connected:** This app must be able to run when no network connection is available. This app first operates without a network connection, and must be able to run without a network connection. Once network connectivity is restored, the app can perform network-dependent tasks such as synchronizing with a remote database, make a web service call, or use a web component.
  E.g.: Karpathos guide [http://karpathos.in/], App GTT Mobile, K-9 Mail,...