Mobile apps with React Native

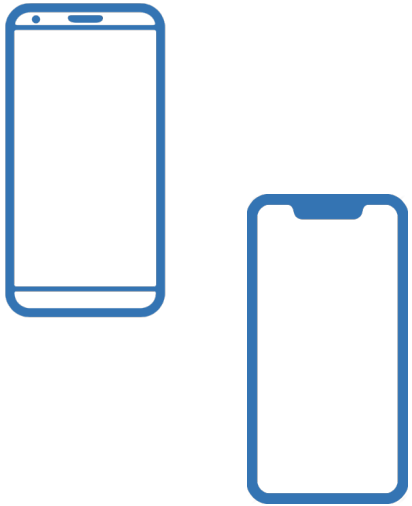
UNITO - Programmazione Mobile
Prof. Ferruccio Damiani

Marco De Nittis
marco.denittis@synesthesia.it

synesthesia

Agenda

- What is mobile dev
- What is react native
- How it works
- Conclusions

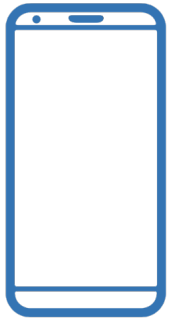


What is mobile development

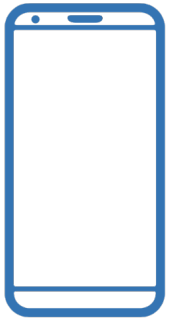
- App on mobile devices
- Mainstream platforms
  - iOS - Apple
  - Android - Google
- Flavours
  - phones
  - tablets

Why is different

- Different interaction from desktop
- Ad hoc UI/UX Paradigms
- Limited resources
  - CPU
  - Bandwidth
  - Battery
  - Limited screens (related to touch)



- Limited user focus
  - **few seconds** attention
  - on the go experience
  - 2 taps rule



Traditional / native development

- Each platform has **specific paradigms**
  - Dev environment
  - Language
  - Technological stack
  - Idioms
  - Libraries
  - Architecture
  - Publishing rules and workflows
- Nevertheless similar paradigm features and limitations

Technological stacks

- Apple
  - Xcode
  - Swift / Objective C
  - iOS Simulator
  - iPhone / iPad
  - App Store
- Android
  - Android Studio
  - Java / Kotlin
  - Android Emulator
  - Android devices
  - Play Store

App distribution

- Distribute apps to real users
- No easy alternatives (especially iOS)
  - Security constraints on devices (digital signatures)
  - In-house distribution

- Approval workflow before publishing
  - Approval on app updates

- **No control** over customer updates

Cross platform development

- Develop in single codebase for both platform
- Single development language
- Manage platform impedance mismatch
- Adapt single dev paradigm to 2 worlds
- Integrate with native functionalities
  - Camera
  - Position / GPS
  - Notifications

Cross platform vs Hybrid

- Cross platform
  - **app oriented paradigm**
- Hybrid apps
  - **HTML based**
  - Adapted web development

Cross/Hybrid today

- Cordova / phonegap / ionic
  - web applications on web views
  - html + js
- Appcelerator
  - js based
  - fullstack platform
  - paid
- Xamarin
  - .NET based
- React Native
  - js / react based
  - native UI
- NativeScript
  - js / angular vue based
- Flutter
  - Dart based
  - Native compiled
- Progressive Web Apps (not really apps)

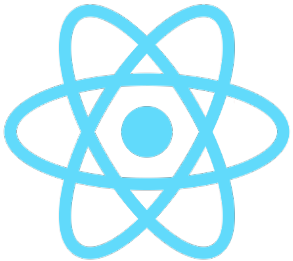
Pros

- Faster development
- Single codebase (almost)
- Broader language knowledge

Cons

- Slower apps
- Integration with native features
- Duplicate part of code
- Workaround for different paradigms

- JS Based / React Based mobile dev framework
  - Made with ❤️ by facebook
- Evolution/Adaptation from React JS Technology
  - From web to mobile
- Recycle web skill to develop mobile apps



Not html / js solution => native UI

Adopted by several (big) players

- Facebook
- Uber
- Tesla
- AirBnb (not anymore)
- Wallmart
- Mulino Bianco
- Synesthesia =)

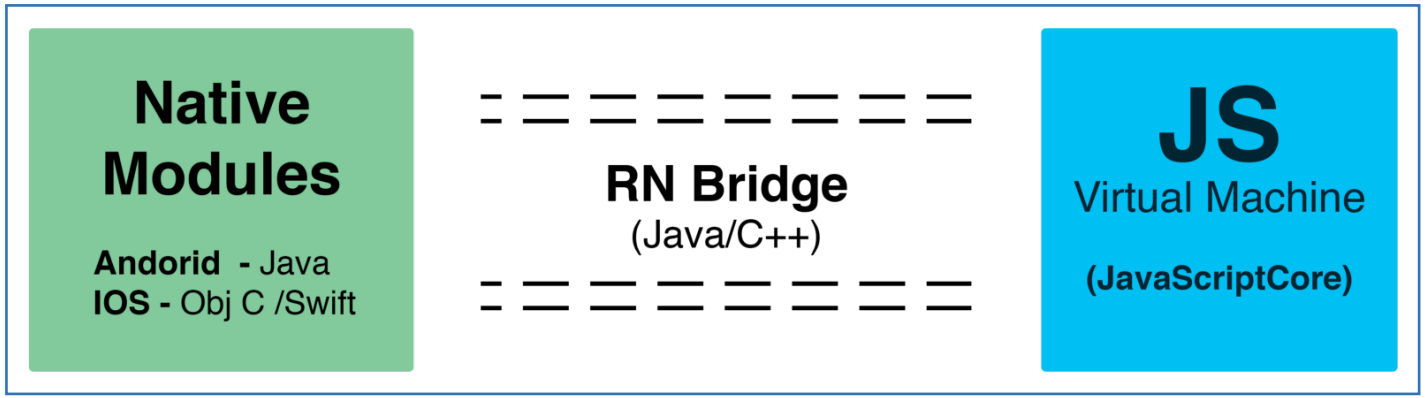
Technoloy stack

- React
  - JS 2015+
  - JSX (JavaScript Xml)
- NPM - external libraries
- XCode
- Android Studio

- JS toolchain
  - node.js
  - BabelJS
  - Metro bundler
  - Chrome debugger

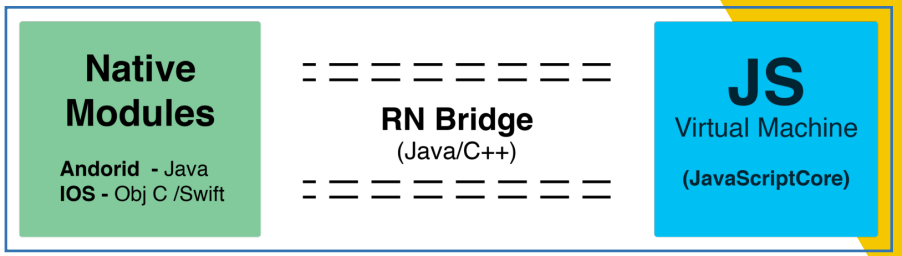
- JS for Business Logic
- Native for UI
  - Speed boost
- React Native Bridge (The Bridge)
  - Message bus between JS and Native realms
  - Bidirectional

The Bridge


Native Modules
Andorid - Java
IOS - Obj C /Swift
RN Bridge
(Java/C++)
JS
Virtual Machine
(JavaScriptCore)

The Bridge

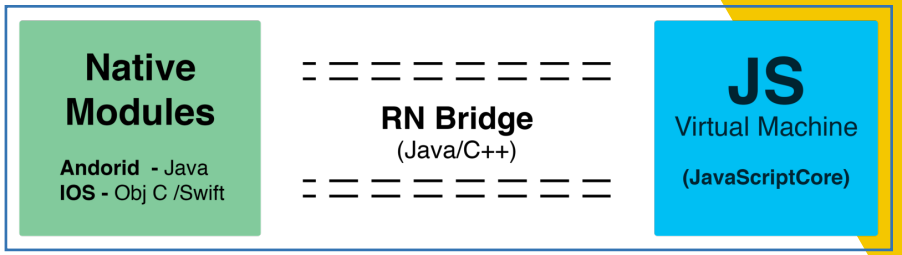
- Keeps in sync the two Realms
  - React (JS) - App Logic
  - Native Code - UI



- Separate Threads
  - Delay of several ms
  - Not suitable for realtime responsiveness (es: arcade games)

The Bridge

- Messages with layout change
  - JS => Native



- Messages with UI events
  - Native => JS

- Based on JavaScript Core Engine
- Host React app
  - App Logic
  - Api call
  - Whatever
- NPM packages
  - External libraries

Native Realm

- Run on native (ios / android)
- UI Rendering
  - flexbox-like layout system
- Native Widgets
- Native Modules
- Custom native code
- HW/SDK features
  - geolocalization
  - notifications
  - camera
  - ...

- Constraints based on Native SDKs
  - Android: MacOs / Windows / Linux
  - iOS: MacOs only
    - Constraints on Apple toolchain

Development tools

- node.js
- create-react-native-app
  - Scaffolds projects
- IDE for JS
  - VS Code
  - JetBrain WebStorm
  - Atom / Whatever
- Android Studio / Android SDK / Xcode
- React Native Bundler (Metro)
- React Native Debugger
- Device simulator / emulator
- Real devices

DEMO

Mere prattle without practice.

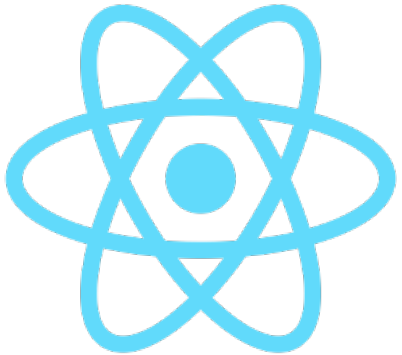
W. Shakespeare

- Root for react project
- package.json - dependencies
  - free JS structure
- platform specific folders with native projects
  - iOS
  - android

- JS 2015 + / ES6+
  - feature rich and structured language
  - module management
  - transpiled to support new features
- JSX: JavaScript XML
  - XML like component declaration inside JS

- Transpiled to effective JS
  - BabelJS
- Modern and structured language
  - class
  - Module system
  - Effective constructs

- View Management Framework
- Strongly Component based
  - Composition
  - Reusability
- From Web Development
  - Agnostic to effective medium
  - Virtual DOM
- Expressed via JSX



Simple Component (WEB)

```
1  import React from 'react';
2
3  const SayHello = () => {
4    return (
5      <div>
6        <div>Hello!</div>
7        <span>funny right?</span>
8      </div>
9    );
10 }
11
12 export default SayHello
```

- Focused on UI
- No app logic - state management architecture
  - Several de facto standard
- redux
  - Predictable
  - Time machine debugging

Simple Component (React Native)

```
1  import React from 'react';
2  import { View, Text } from 'react-native';
3
4  const SayHello = () => {
5    return (
6      <View>
7        <Text>Hello!</Text>
8        <Text>funny right?</Text>
9      </View>
10     );
11 };
12
13 export default SayHello;
```

React Native Components

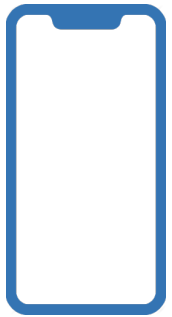
- Same code as web
  - UI generic *tags*
    - View, Image, Text ...
- Recycle logic code (eventually)

```
1  import React from 'react';
2  import { View, Text } from 'react-native';
3
4  const SayHello = () => {
5    return (
6      <View>
7        <Text>Hello!</Text>
8        <Text>funny right?</Text>
9      </View>
10     );
11 };
12
13 export default SayHello;
```

- Native components
  - Platform agnostic
  - Device features
- UI disposition with css-like
  - flex-box layout
- Easy integration with native code
  - Platform & custom API

Developer workflow

- Run app in dev mode
  - On simulator / device
- Write code
- Realtime (almost) refresh code in app
  - Metro bundler compiles delta
  - Push to code dev device
- Check the changes
- Debug on Chrome debugger
  - React Native debugger
  - Integrated RN tools and React JS tools



DEMO

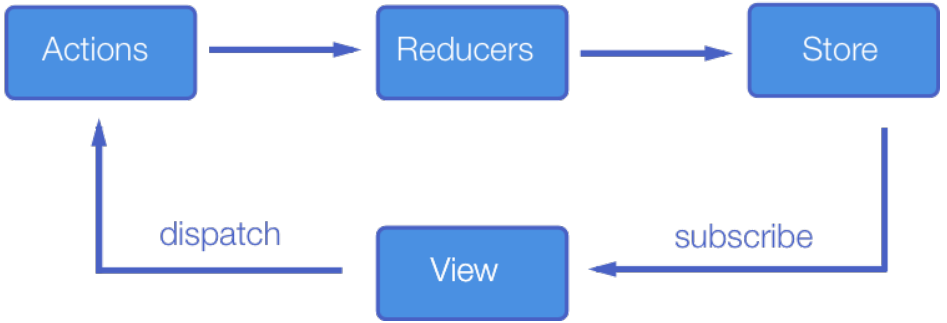
Talk is cheap. Show me the code.

L. Torvalds

- Aka Data & State of the app
- React has simple state model
- External libraries
  - Redux: de facto standard
  - mobx
  - ...

- Single global state of the app
- State immutability
- Predictable state management
  - Predictable app behaviour
  - Time machine debug
- Simple model
  - Not easy at the beginning

- unidirectional data flow
- actions
  - messages of events
- reducers
  - change the state
  - pure functions



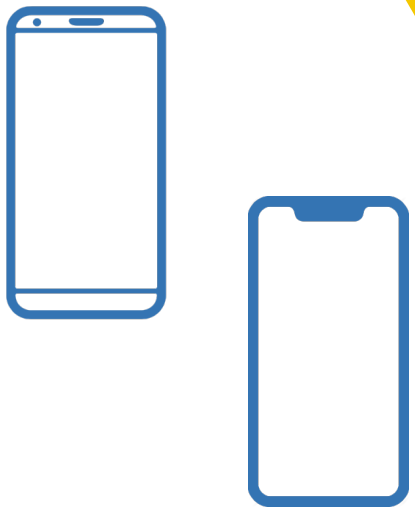
DEMO

It is lightning that strikes, not thunder.

M. Dhliwayo

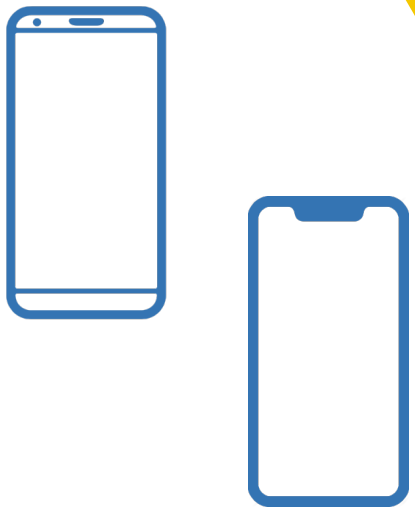
Mobile specific behaviours

- Navigation
  - Mobile navigation != web navigation
  - Platform specific behaviour
    - Android HW back button
  - External libraries: react navigation



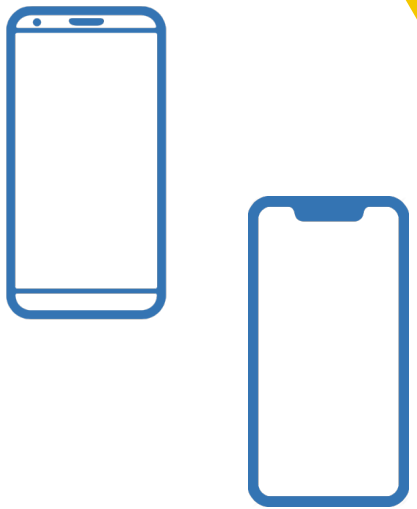
Mobile specific behaviours

- Permissions
  - Manage user permission to access device features
  - Notifications
  - Position
  - Camera
  - ...
- Platform and UX concerns



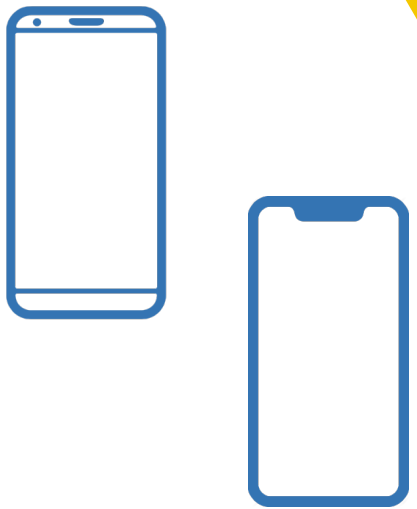
Mobile specific behaviors

- Notification
  - Manage permissions
  - In app notifications
  - Push notifications
  - Data payload
- Platform agnostic services (Firebase)



Mobile specific behaviors

- UX/UI paradigms
  - UX guidelines
    - nav tab behaviours
    - pickers
    - maps
    - ...



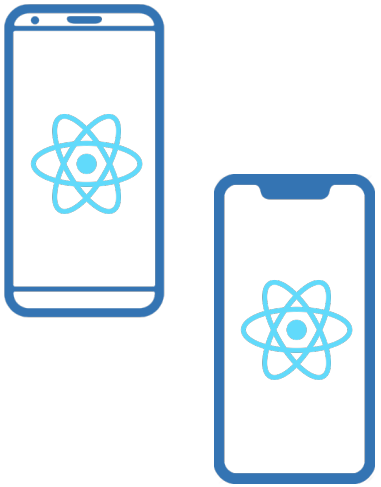
Pros

- Ultra fast development flow
  - Fantastic dev tools
  - Hot Reload
- Javascript easy to know
- React is stable
- Single codebase (almost)
- Incredible amount of components
- Easy integration with native API / components

Cons

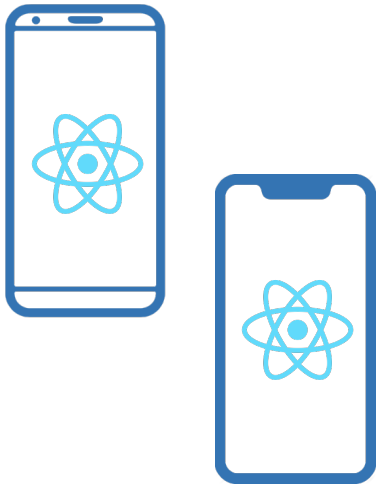
- Steep learning curve
- React + redux
- Require a lot of different skills
  - JS, iOS, (Android), React, node...
- Dependencies upgrade hell
- Several moving parts
- UI specific platform customisations
- Performance issues on computing intensive tasks
- Not easy to balance JS vs native

- It depends:
  - personal/team skills
  - App type:
  - Realtime / CPU intensive?
  - Platform support
  - Single or both iOS + Android
- As 100 effort for single native platform
  - => with RN: 120 - 180 for iOS + Android



Worth learning?

- JS is one of the most growing language
- React is very actual and live technology
- Mobile cross technologies are trending
- Several Job offers
  - More for native platforms
  - Market is evolving swiftly



React Native

Thank you

synesthesia

Marco De Nittis

marco.denittis@synesthesia.it

mdn