

# Native Mobile Apps with Clojure(Script)

Chris Vermilion  
[chris@figlyinc.com](mailto:chris@figlyinc.com)

Boston Clojure Meetup 9/10/15

# Why Clojure for mobile?

- Maybe you like Clojure!
- Maybe you have a smartphone and want to use it!
- Something something functional something immutability
- *Code as data is an interesting model for a cloud-updatable app!*

# Possible approaches

Pure Clojure/  
ClojureScript  
=  
mobile web

ClojureScript  
+  
JS-native bridge  
=  
hybrid app

Clojure  
+  
(some kind of dark magic)  
=  
native app

# Possible approaches

Decreasing obviousness



Pure Clojure/  
ClojureScript  
=  
mobile web

ClojureScript  
+  
JS-native bridge  
=  
hybrid app

Clojure  
+  
(some kind of dark magic)  
=  
native app

Increasing difficulty



# Possible approaches

Pure Clojure/  
ClojureScript  
=  
mobile web

ClojureScript  
+  
JS-native bridge  
=  
hybrid app

Clojure  
+  
(some kind of dark magic)  
=  
native app

Not going to cover this one

# Possible approaches

Pure Clojure/  
ClojureScript  
=  
mobile web

ClojureScript  
+  
X  
=  
hybrid app

Clojure  
+  
(some kind of dark magic)  
=  
native app

Lots of choices for “X”:

PhoneGap, Cordova, etc.

New hotness: React Native

# Possible approaches

Pure Clojure/  
ClojureScript  
=  
mobile web

ClojureScript  
+  
X  
=  
hybrid app

Clojure  
+  
RoboVM (iOS)/  
various (Android)  
=  
native app

Android's already Java. On  
iOS, compile Java bytecode to  
clang somehow?

# More on Clojure+Android

<http://clojure-android.info/>

Things I know about Clojure on Android:  
1) It can be done.



# RoboVM



“Create truly native iOS apps in Java”

Two things make this possible:

- Java bytecode to x86/ARM compiler
- Objective-C -> Java bindings for Cocoa libraries

## Cool story, Bro.

```
import org.robvm.rt.bro.*;
import org.robvm.rt.bro.annotation.*;

@Library("c") // [1]
public class Abs {
    static {
        Bro.bind(); // [3]
    }
    @Bridge private static native int abs(int i); // [2]
    public static void main(String[] args) {
        System.out.println(abs(-100));
    }
}
```

Bro: Java bindings to C/Objective-C code

<http://docs.robvm.com/advanced-topics/bro.html>

# Gets hairy quickly...

```
/*</javadoc>*/
/*<annotations>*/@Library("Foundation") @NativeClass/*</annotations>*/
/*<visibility>*/public/*</visibility>*/ class /*<name>*/NSArray/*</name>*/ <T extends NSObject>
    extends /*<extends>*/NSObject/*</extends>*/
    /*<implements>*/implements NSFastEnumeration, NSPropertyList, List<T>/*</implements>*/ {

    . . .

    /*<properties>*/
    @Property(selector = "count")
    protected native @MachineSizedUInt long getCount();
    /**
     * @since Available in iOS 4.0 and later.
     */
    @Property(selector = "firstObject")
    public native T first();
    @Property(selector = "lastObject")
    public native T last();
    @Property(selector = "sortedArrayHint")
    public native NSData getSortedArrayHint();
    /*</properties>*/
    /*<members>*//*</members>*/
}
```

Whole NSArray wrapper is  
~400 LOC (and only  
covers the basic methods)

# RoboPods should make this easier...

— just drop a dependency in Gradle/Maven and you're good to go

## List of available RoboPods for iOS

Name	Version	Dependency
<a href="#">Bolts</a>	1.1.5	org.robovm:robopods-bolts-ios:1.6.0
<a href="#">Chartboost</a>	5.5.0	org.robovm:robopods-chartboost-ios:1.6.0
<a href="#">Facebook</a>	4.3.0	org.robovm:robopods-facebook-ios:1.6.0
<a href="#">Flurry</a>	6.6.0	org.robovm:robopods-flurry-ios:1.6.0
<a href="#">Google Analytics</a>	3.12	org.robovm:robopods-google-analytics-ios:1.6.0
<a href="#">Google APIs</a>		org.robovm:robopods-google-apis-ios:1.6.0
<a href="#">Google Mobile Ads</a>	7.3.1	org.robovm:robopods-google-mobile-ads-ios:1.6.0
<a href="#">Google Play Games</a>	1.4.1	org.robovm:robopods-google-play-games-ios:1.6.0
<a href="#">Parse</a>	1.7.4	org.robovm:robopods-parse-ios:1.6.0

So what?

Bottom line: RoboVM lets you treat Android and iOS as two (different!) JVM platforms.

This is a double-edged sword!

## Great power:

- At the end of the day, you have a compiled app.
- Real native access to anything you want.
- 100% (in principle) native feature coverage, pretty good JVM (and Clojure) feature coverage.

## Great responsibility:

- Bindings are ugly and you might have to write some.
- *UI code is all platform specific!*

```

(ns radiator-ios.textrender
  (:require [radiator-ios.ios-text :as text])
  (:import [org.robovm.apple.uikit UIColor UILabel UIView]))

(defn rgba->UIColor [rgba-vector]
  "Get iOS specific color where style is [r g b a].
  r, g, b are between 0-255. a is between 0 and 1.0 (defaults to 1.0)"
  (let [[red green blue] (->> (subvec rgba-vector 0 3)
                                (map #(/ (or % 0) 255)))
        alpha (last rgba-vector)]
    (UIColor/fromRGBA (double red) (double green) (double blue) (double alpha))))

(defprotocol UIViewRenderer
  (make-uiview ^UIView [this uitree state event-channel] "Creates a UIView"))


(defrecord TextRenderer []
  UIViewRenderer
  (make-uiview ^UILabel [_this uitree _state _event-channel]
    (let [label (UILabel.)]
      (.setText label (or (:content uitree) ""))
      (when-let [color (get-in uitree [:slot :style :color])]
        (.setTextColor label (rgba->UIColor color)))
      (when-let [alignment (get-in uitree [:slot :style :text-align])]
        (.setTextAlignment label (text/style->alignment alignment)))
      (when-let [style (get-in uitree [:slot :style])]
        (.setFont label (text/style->UIFont style)))
      label)))

```

# Parting thoughts

1. RoboVM is a really interesting way to get totally native access with a JVM language.
2. It's more about *language* portability than app portability. RoboVM gives you Java access to iOS libraries, *not* an abstraction on top of iOS/Android.
3. Dev cycle of edit/compile/test/simulate is pretty slow and painful. May improve?
4. RoboVM as a product is young, immature. Maybe growing?



A photograph of a man swimming in a pool of water on a road. The road is paved and has double yellow lines. The water is splashing around the man, who is shirtless and has dark hair. The background shows a road curving away into the distance with some greenery on the sides.

There is always  
another way!

“Hybrid”: web-like portability, native bindings when you need them



PhoneGap



APACHE  
CORDOVA™



titanium™



**CROSSWALK**  
PROJECT



ionic

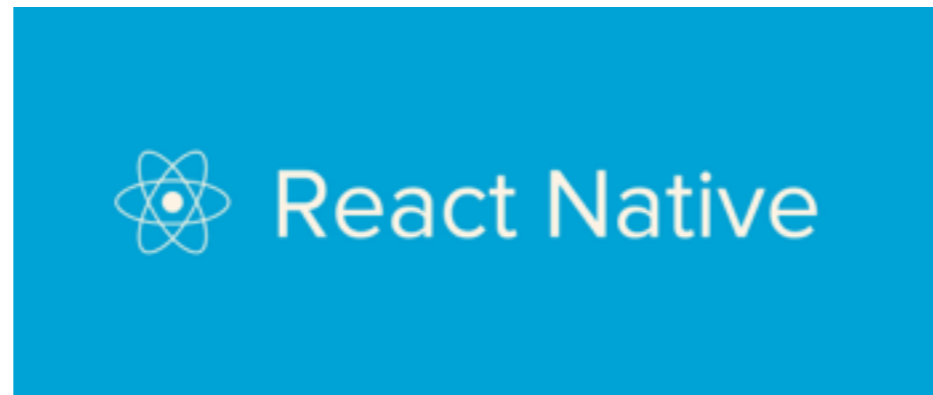
“Hybrid”: web-like portability, native bindings when you need them



PhoneGap



APACHE  
CORDOVA™



titanium™



CROSSWALK  
PROJECT



ionic

Why React? Why React Native?

Why React? Why React Native?

Honest answer: This is my talk and that's what I've used!

# Why React? Why React Native?

- Virtual DOM, performance, Facebook backing, etc.
- Good Clojure uptake (Om/Reagent/Quiescent) [Why? *Immutability* maps well to React model!]
- React Native then gives access to native bindings when needed
- May or may not be a good tradeoff, depending on what you want native for!

# Why hybrid instead of compiled?

- Maybe you just like JavaScript!
  - Threads are overrated anyway.
- Use JS, webby language for describing UI
  - => UI code is actually portable
  - => Flipside: UI code doesn't nec. "feel native"
- Only need to go native when you need/want to
  - => BUT "lowest common denominator" UI abstractions
- *Way* faster dev cycle: simple autorefresh, etc.

OK, shut up and show me something.

<https://github.com/dmotz/natal>



# Final thoughts

The right tool for native development depends on *why you want native in the first place*.

Reasons you might want compiled/Clojure

- Lots of complicated native functionality, in lots of places in your code
- Non-UI code dominates, want it to be performant (games, eg)
- Native controls/look-and-feel is a priority

Reasons you might want hybrid/ClojureScript

- UI is the most important/frequently changing code, want portability
- Web-like UI is what you want (maybe also you want a web client!)
- Dev speed/tooling is a priority
- Vendor-dependency makes you nervous

# Resources

RoboVM:

<http://robovm.com/>

Clojure+Android links:

<http://clojure-android.info/>

React Native:

<https://facebook.github.io/react-native/>

ClojureScript+React Native bootstrapper:

<https://github.com/dmotz/natal>

ClojureScript + iOS REPL:

<https://github.com/omcljs/ambly>

ClojureScript + RN + Reagent example project:

<https://github.com/mfikes/reagent-react-native>