React Native with ClojureScript

Michael Langford
@mj_langford

A Bit About Me:

- Embedded Programming Background (firmware/robotics/ embedded Linux)
- iOS Developer since 2008, some Android off and on
- Clojure in backends/tools off and on since then 2014, looking at using it for Android/Multi Platform devices

Overview of the Platform Story

Principles of Mobile

- Old Versions of code can exist for long after you want them to
- Cryptographic Signing is an annoying and ever present component
- App Stores set standards for the apps you *must* adhere to for public apps

Principles of Mobile (cont)

- Smaller surface spaces than computers, so consistency with UI norms is essential
- Fingers are far far fatter than mouse pointers: use different controls than web
- Prepare for maintenance cycle related to new major OS version release

Principles of Mobile (cont)

- Authentication not always needed...local data doesn't need auth
- An app is far more available and personal than a website, ever available, many sensors
- Simulator/Emulators are not the same thing as running on a device

You can just write apps on a simulator...but where's the use of that?

To use it on your phone, you need to do a few things (platform specific)

- Create a developer account
- Provision your phone

To put it on a Store

- Android: Make a signing key, never ever lose it
- Apple: Make a signing key, preferably in a separate user and import it into your main account
- Either Platform, sign it, upload it, and fill in screen shots
- For Apple, you wait a week or so, and hope you met all the design and functionality guidelines

Speaking of Guidelines

Documents to comply with on iOS

Human Interface Guide https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

App Review Guidelines: https://developer.apple.com/app-store/review/guidelines/

Guidelines for Android

Material Guide for Android: http://developer.android.com/design/material/index.html

Google Play Store Guidelines: https://play.google.com/about/developer-content-policy.html

On To The Specifics

React Native: By Facebook!

- RN is a Facebook project
- *= Android 4.1 (API 16) and >= iOS 7.0*
- Allows native modules (Objective-C or Java and some 3rd parties have done Swift) if RN doesn't have a piece of functionality you need
- Requires you to deal with "NPM and JS quality" toolchains

What does RN do different from Native?

- React-style composable UIs
- Declarative UI instead of stateful imperative
- No GUI UI builders
- Flexbox instead of AutoLayout(iOS)/Layouts(Android)

How is it different than using a WebView (Cordova, etc)?

- Uses JavaScriptCore to run the apps, no WebView required
- Native components, the same components as native code does
- Multiple threads
- Far better performance

Tools

- Natal -> iOS Om or Om.Next easily, others with effort (https://github.com/dmotz/natal)
- Re-Natal -> iOS/Android Reagent+re-frame (https://github.com/drapanjanas/re-natal)
- Standard ClojureScript libraries for networking/non-view features

Still requires the full toolchain for the platform

- iOS requires Xcode, Mac OS X
- Android requires Android Studio still

Preparing for from the command line (OS X version)

brew install ant

brew install maven

brew install gradle

brew install android-sdk

brew install android-ndk

(https://gist.github.com/patrickhammond/4ddbe49a67e5eb1b9c03)

Preparing for iOS from the command line

xcode-select --install

Push some buttons

Preparing the node tools to get the ClojureScript environment

- brew install node
- npm install -g natal

or

- npm install -g re-natal

Create the app (Pick One!, this can be slow)

- Om.Now: natal init WhateverYouCallIt
- Reagent: re-natal init WhateverYouCallIt
- Om.Next: natal init --interface om-next WhateverYouCallIt
- Some of these auto-launch the iOS simulator when done

Open up the core.cljs file and start editing

Time to examine one of them. I'm going to go for Om, as the principle carries through: it's the web framework with different functions to generate elements

Om (aka, Om.Now)

om/root function takes the top level view, the app state atop, and I honestly don't know what the target map is for in the context of React Native.

(om/root main-view app-state {:target 1})

om/IRender protocol and the render function

with-error-view: This is a wrapper class that catches errors and displays them in the simulator

view: This specifies a View, the basic building block of items that display on the screen. The first param is a map of attributes, and the rest of the params are other controls

The other "controls" specified

text: This specifies basic static text. Same prototype, first param map of attributes, mostly formatting, but other notables, onLayout for animations/becoming viewable, testID for calling this item up in tests.

image: This specifies a image, and can summon things from the web instead of local (not suggested for most things, iOS apps often work offline).

The touch "control" specified

touchable-highlight: This specifies a button. I don't know why they don't say "button", but whatever. You can also pass :onPress in the map param

They're just ClojureScript functions though...so you can compose

```
(image
 {:source
   {:uri "https://raw.githubusercontent.com/cljsinfo/logo.cljs/
master/cljs.png"}
 :style {:width 80 :height 80
 :marginBottom 0}})

 ;could be just

 (image-from-web)
```

Layouts are accomplished with the :flexDirection key

```
(view {:style {:flexDirection "column"}} (another-view) (another-view))

(view {:style {:flexDirection "row"}} (another-view) (another-view))
```

Layouts are also accomplished also with the :justifyContent key

```
(view {:style {:justifyContent "flex-start"}} (another-view) (another-view))

(view {:style {:justifyContent "space-around"}} (another-view) (another-view))

(view {:style {:justifyContent "center"}} (another-view) (another-view))
```

Layouts can also be absolute

```
(view {:style {:position "absolute" :top 10 :left
10 :backgroundColor "#ff0000" :height 20 :width 300}})

;This puts a box at 10,10 on a screen
```

There is a repl

- Run natal repl
- In another terminal, run natal launch (and wait)
- Import your app namespace
- Change the app state at will

There is live reload too

- Part of the launch sequence
- Exiting the simulator wrong can mess up the live reload

Go Forth and Fill Your Pockets with Apps

Suggested Resources

Clojurians.slack.com #cljsrn

Github pages of the tools (inline)

Mike Fikes Blog: http://blog.fikesfarm.com

Boot for React Native: https://github.com/mjmeintjes/boot-react-native

React Native website: https://facebook.github.io/react-native/docs/getting-started.html

Demo Time

Thanks

@mj_langford

michael.langford@rowdylabs.com