An iOS Developer’s take on React Native

Harry Tormey

Background


α7

How does React Native Work?

React Native lets you build mobile apps using only Javascript

Is React Native a hybrid mobile app framework?

No

What is a hybrid mobile app frameworks?

Hybrid mobile app frameworks

It’s a native container that uses a web view to display locally hosted HTML


BAD
REPUTATION

Why do hybrid mobile apps have a bad reputation?

They use web views

Web development used to suck :(

Web development has improved a lot :)

Why has web development improved?

- Rise in popularity of functional programming
- Transpilers (Babel)
- Javascript language improvements

What is React?

UI = f*(data)

*No side effects

Count: 0

Add + 1

UI = f(count)

```
render() {
  let count = this.state.count
  return (
    <div>
      <h2>Count: {count}</h2>
      <button onClick={() => ??? }>
        Add + 1
      </button>
    </div>
  )
}
```

This is not HTML. This is the VirtualDOM, the description of our UI.

How do you update state?

Android:

TextView counter = (TextView) findViewByID(R.layout.counter)
counter.setText(’10’)

Objective C:

_counter.text = @“10”

Javascript:

document.getElementByID(‘counter’).children[1].innerHTML = ’10’;
$(‘#counter b’).html(’10’)

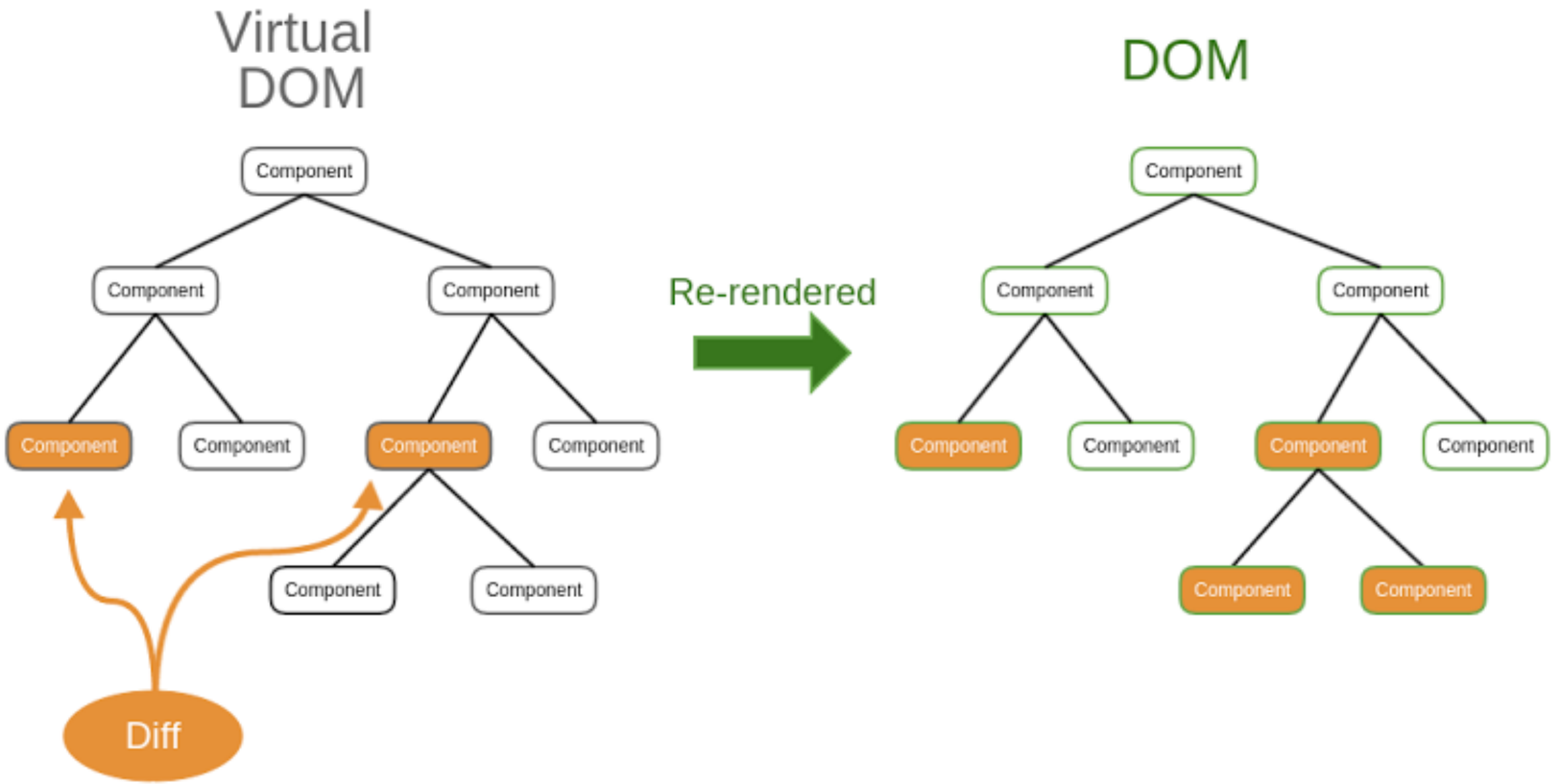
How do you update state with React?

setState

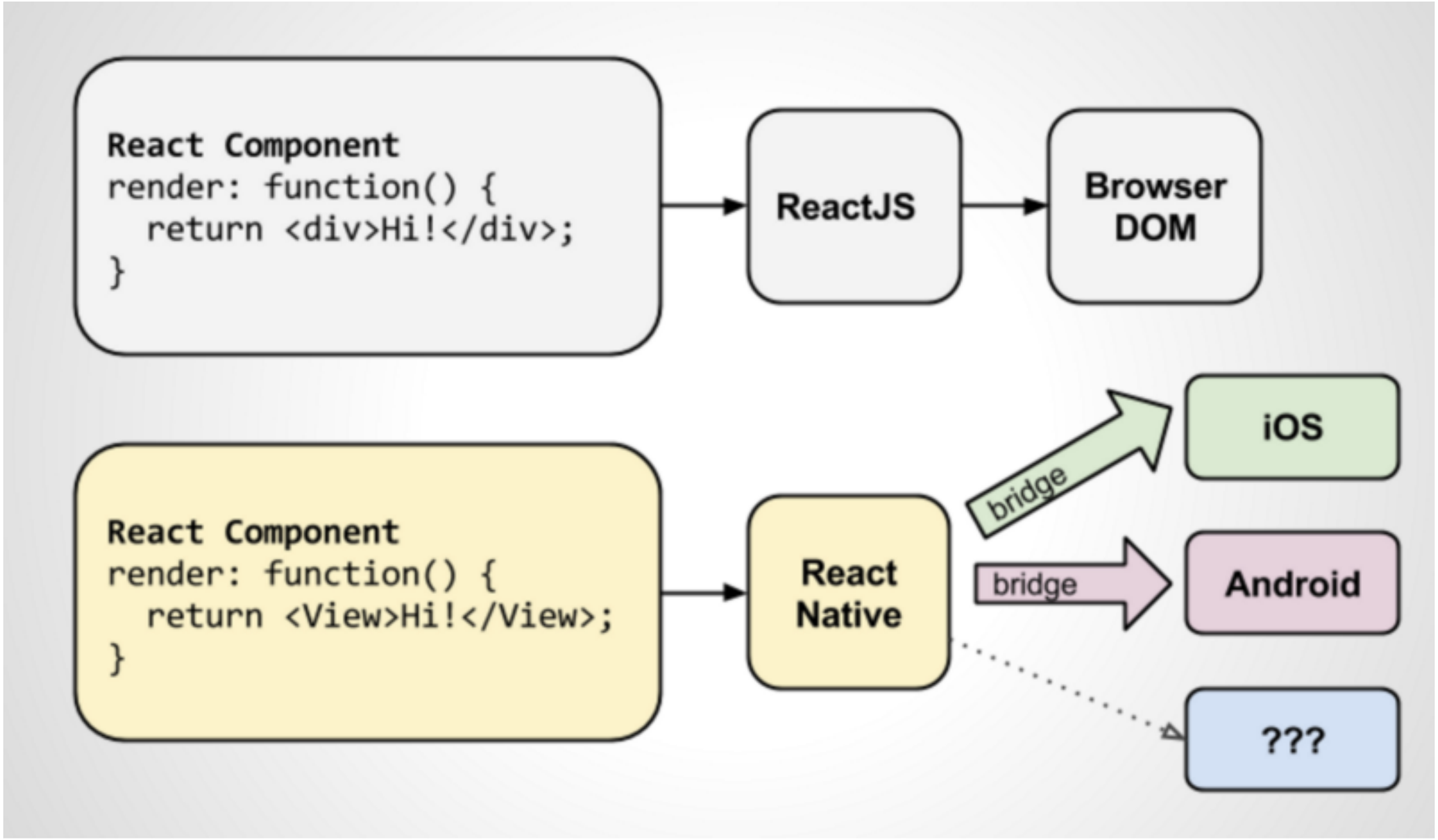
```
class CounterApp extends Component {
  constructor (props) {
    super(props)
    this.state = {
      count: 0
    }
  }
  render() {
    let count = this.state.count
    return (
      <div>
        <h2>Count: {count}</h2>
        <button onClick={() => { this.setState({count:count+1}) } }>
          Add + 1
        </button>
      </div>
    )
  }
}

export default CounterApp
```

What just happened?

- setState increments count (data)
- Results in new VirtualDOM being rendered
- React compares old and new representation and only updates what has changed

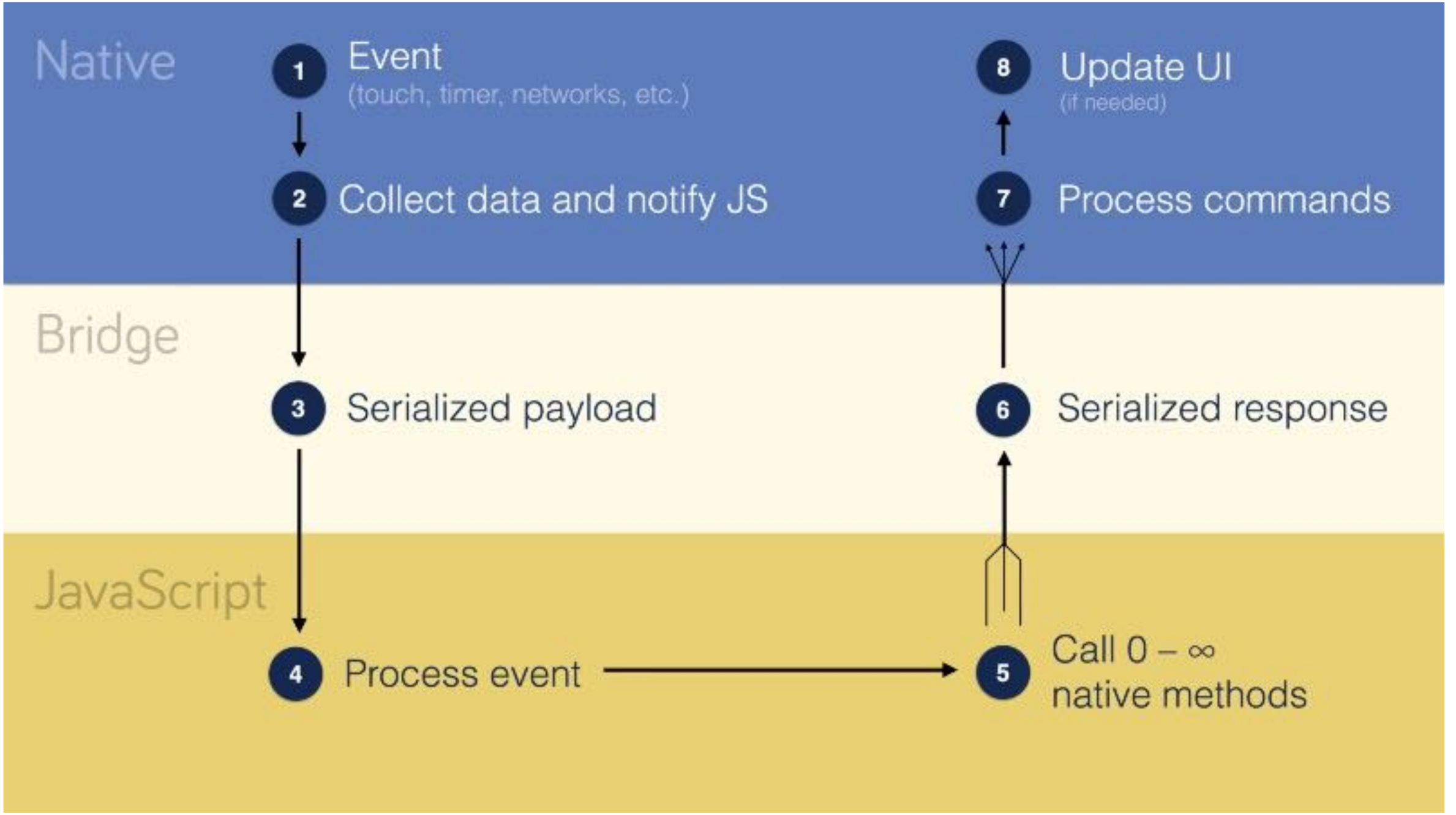

Virtual DOM
Component
Component
Component
Component
Component
Component
Component
Component
Component
Diff
Re-rendered
DOM
Component
Component
Component
Component
Component
Component
Component
Component
Component

How does React Native work?


React Component
render: function() {
return Hi! ;
}
ReactJS
Browser DOM
React Component
render: function() {
return <View>Hi!</View>;
}
React Native
bridge
iOS
bridge
Android
???

```
class CounterApp extends Component {
  constructor (props) {
    super(props)
    this.state = {
      count: 0
    }
  }
  render() {
    let count = this.state.count
    return (
      <div>
        <h2>Count: {count}</h2>
        <button onClick={() => { this.setState({count:count+1}) } }>
          Add + 1
        </button>
      </div>
    )
  }
}

export default CounterApp
```

```
class CounterApp extends Component {
  constructor (props) {
    super(props)
    this.state = {
      count: 0
    }
  }

  render() {
    let count = this.state.count
    return (
      <View style={style.container}>
        <Text>Count: {count}</Text>
        <Button
          onPress={() => { this.setState({count: count + 1}) }}
          title='Add + 1'/>
      </View>
    )
  }
}

export default CounterApp
```


Native
1
Event
(touch, timer, networks, etc.)
2
Collect data and notify JS
8
Update UI
(if needed)
7
Process commands
Bridge
3
Serialized payload
6
Serialized response
JavaScript
4
Process event
5
Call 0 – ∞
native methods

Alexander Kotliarskyi - React Native: Under the Hood

- https://www.youtube.com/watch?v=hDviGU-57lU

The case for React Native

Performance

Even though most of our app's code is written in Javascript, the UI of our app itself is completely native.

- UI is manipulated exclusively on the main thread, but there can be others for background computation. React Native does most of the heavy lifting in this realm for us.

- The JS realm — Javascript is executed in its own separate thread by a Javascript-engine. Our business logic, including which Views to display and how to style them, is usually implemented here.

Where do performance bottlenecks occur?

All React Native apps have code running in two realms (Native/Javascript).Variables defined in one realm cannot be directly accessed in the other.

- This means that all communication between the two realms must be done explicitly over a bridge.
- Each realm by itself is blazingly fast. The performance bottleneck often occurs when we move from one realm to the other.
- In order to architect performant React Native apps, we must keep passes over the bridge to a minimum.

Why is this not a problem?

- Doesn’t use web views
- Easy to wrap native components
- Optimizing the bridge in React Native is relatively easy
- Overdraw issues can be avoided by carefully choosing third party libraries

Wrapping Native Code

Objective C:

- Macros to wrap functions
- Supports emitting events
- Supports callbacks
- Supports Promises
- Allows you to specify thread your native code runs on

Swift:

- Can do everything that Objective C can but uses modifiers instead of macros

```objc
#import <React/RCTBridgeModule.h>

@interface CalendarManager : NSObject <RCTBridgeModule>
@end
```

```objc
#import "CalendarManager.h"
#import <React/RCTLog.h>

@implementation CalendarManager

RCT_EXPORT_MODULE();

RCT_EXPORT_METHOD(addEvent:(NSString *)name location:(NSString *)location)
{
  RCTLogInfo(@"Pretending to create an event %@ at %@", name, location);
}
```

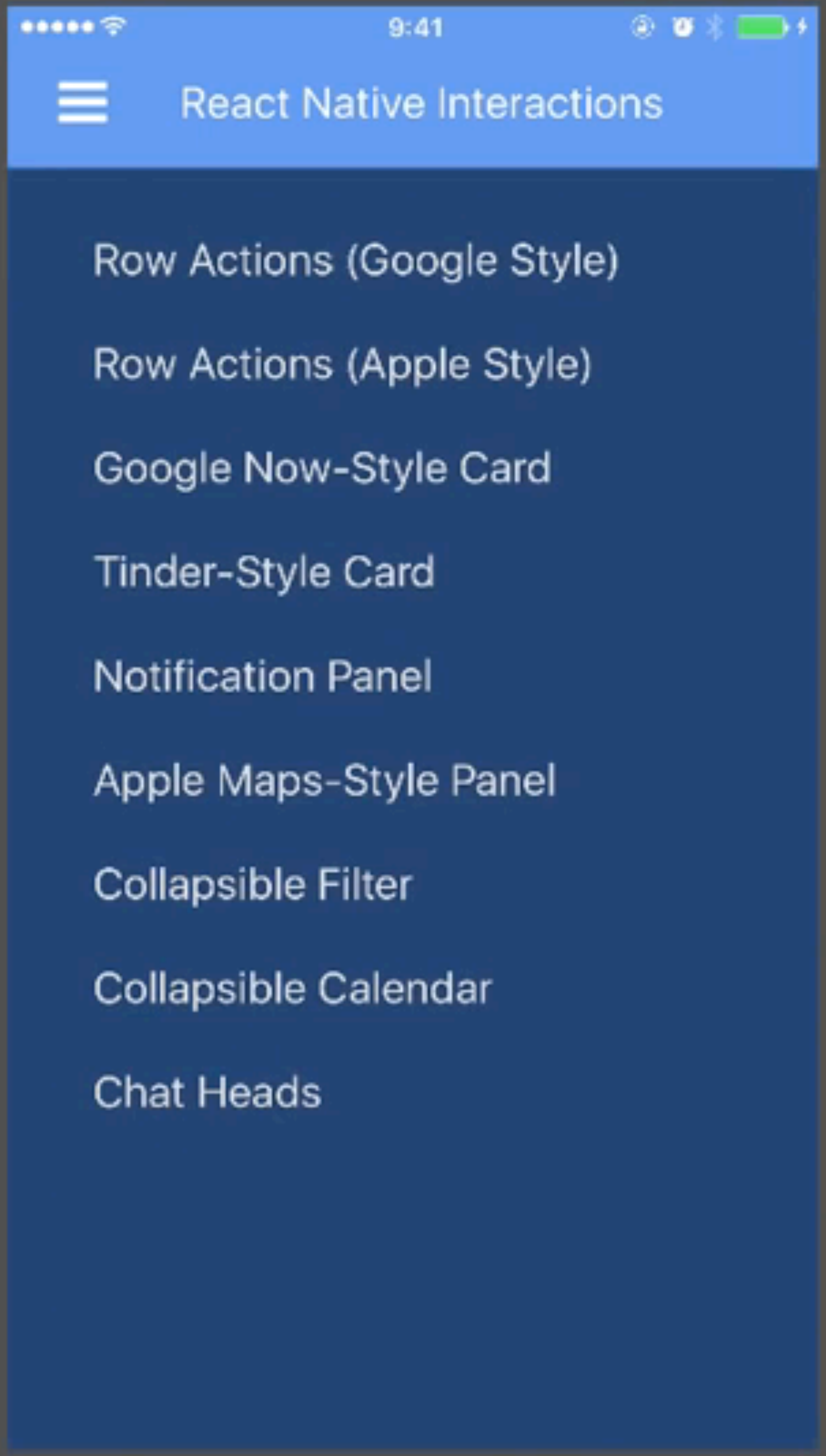
I made React Native fast, you can too

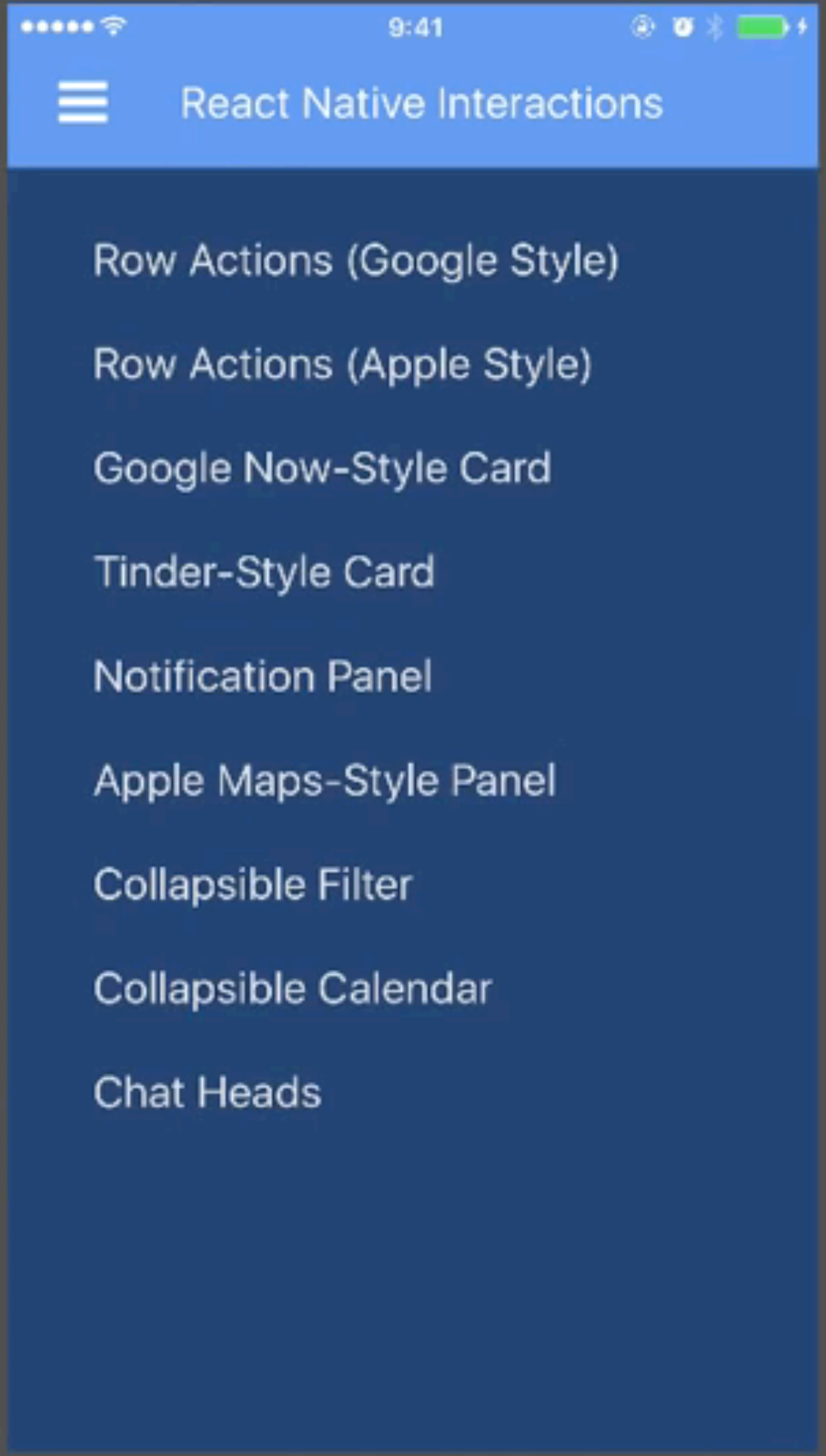
https://launchdrawer.com

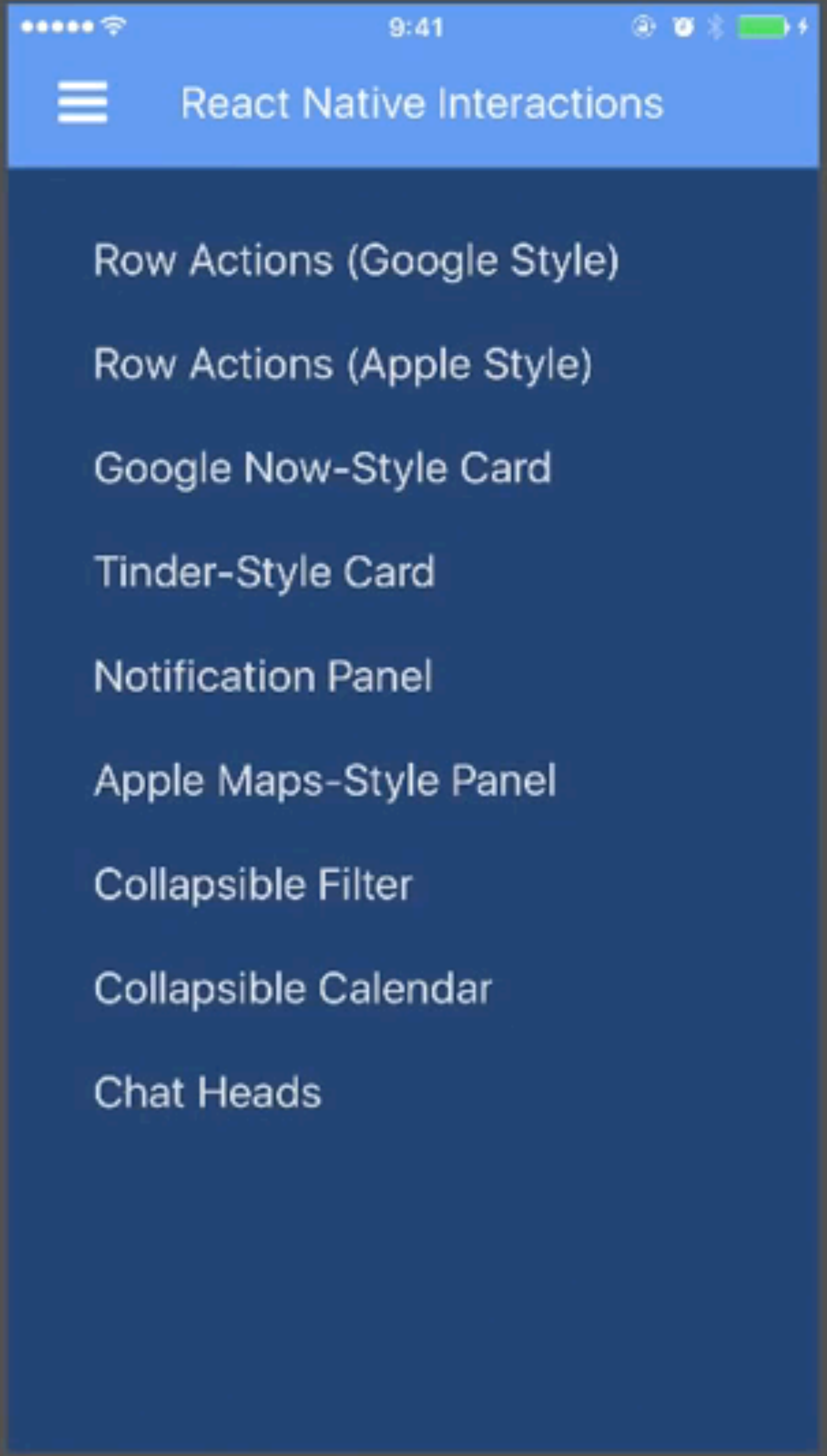
Articles on performance

- Recycling Rows For High Performance React Native List Views: https://medium.com/@talkol/recycling-rows-for-high-performance-react-native-list-views-628fd0363861
- Moving Beyond Animations to User Interactions at 60 FPS in React Native: https://hackernoon.com/moving-beyond-animations-to-user-interactions-at-60-fps-in-react-native-b6b1fa0ba525
- React Native Performance https://hackernoon.com/react-native-performance-an-updated-example-6516bfde9c5c

Demo app implemented in React Native

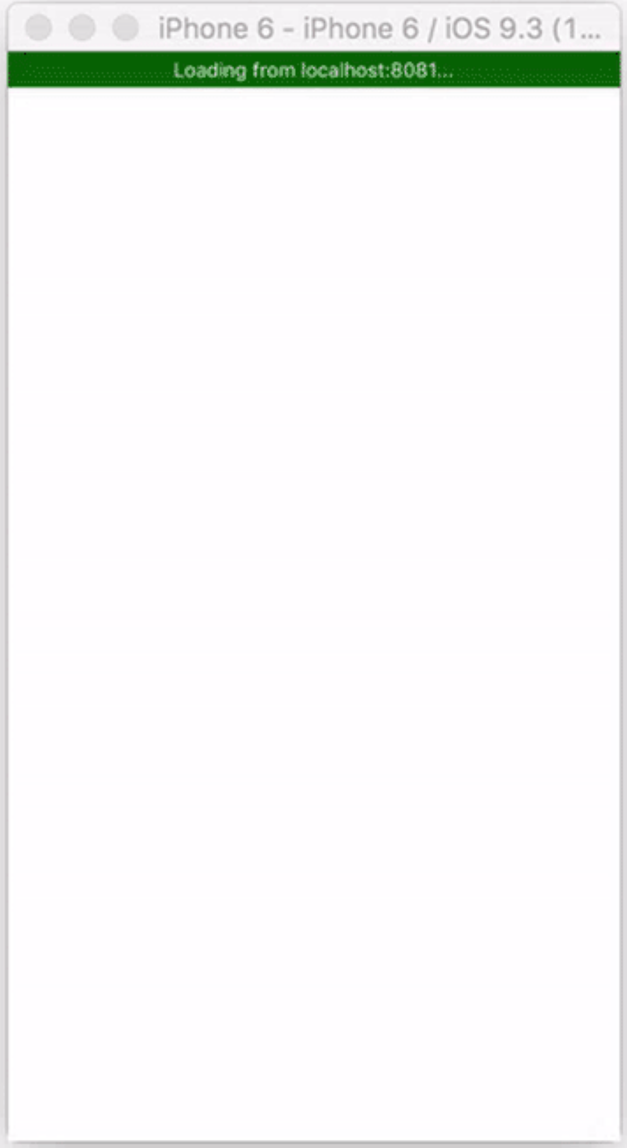


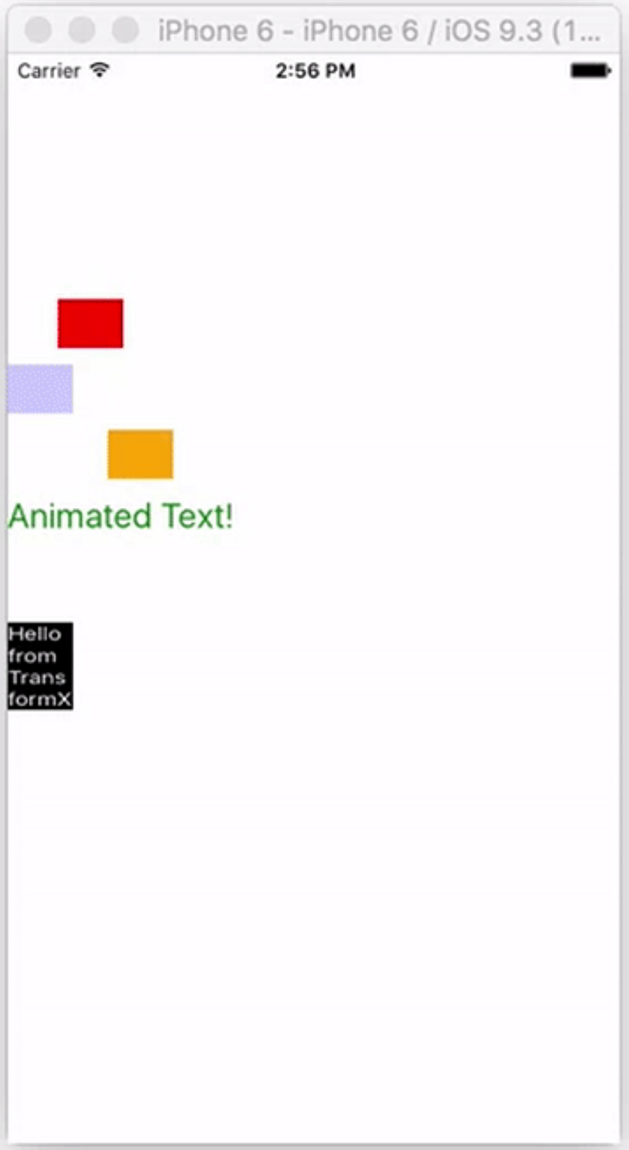




Search app stores for

"React Native Interactions"


iPhone 6 - iPhone 6 / iOS 9.3 (1...
Loading from localhost:8081...


iPhone 6 - iPhone 6 / iOS 9.3 (1...
Carrier
2:56 PM
Animated Text!
Hello from TransformX

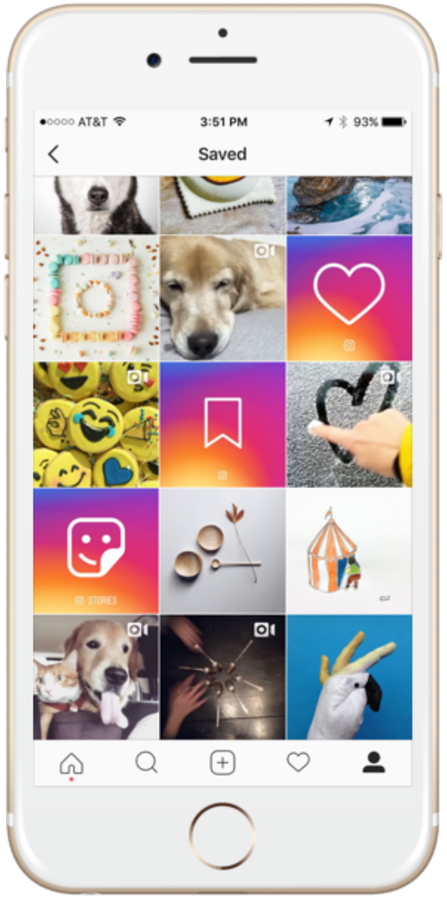
```
componentDidMount () {
  this.animate()
}
animate () {
  this.animatedValue.setValue(0)
  Animated.timing(
    this.animatedValue,
    {
      toValue: 1,
      duration: 2000,
      easing: Easing.linear
    }
  ).start(() => this.animate())
}
```

```
render () {
  const marginLeft = this.animatedValue.interpolate({
    inputRange: [0, 1],
    outputRange: [0, 300]
  })
  const opacity = this.animatedValue.interpolate({
    inputRange: [0, 0.5, 1],
    outputRange: [0, 1, 0]
  })
  const movingMargin = this.animatedValue.interpolate({
    inputRange: [0, 0.5, 1],
    outputRange: [0, 300, 0]
  })
  const textSize = this.animatedValue.interpolate({
    inputRange: [0, 0.5, 1],
    outputRange: [18, 32, 18]
  })
  const rotateX = this.animatedValue.interpolate({
    inputRange: [0, 0.5, 1],
    outputRange: ['0deg', '180deg', '0deg']
  })
...
}
```

React Native Animations

- Examples: https://github.com/react-native-training/react-native-animations
- Article by Nader Dabit https://medium.com/react-native-training/react-native-animations-using-the-animated-api-ebe8e0669fae

Instagram

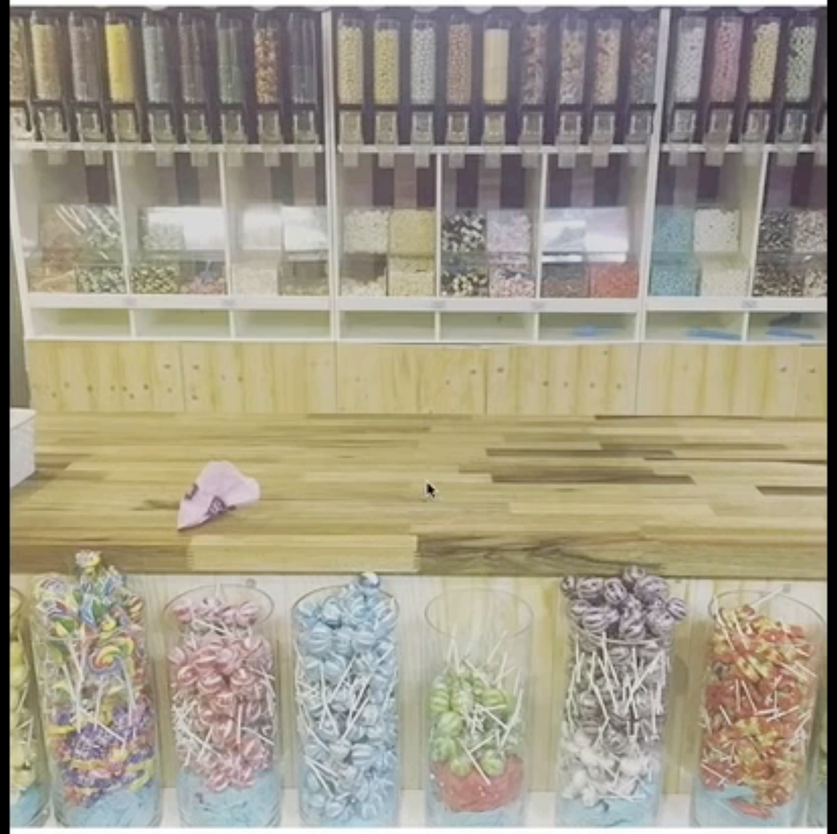


- https://engineering.instagram.com/react-native-at-instagram-dd828a9a90c7

Carrier 7:53 PM

Photo

martinbigio
Facebook London >

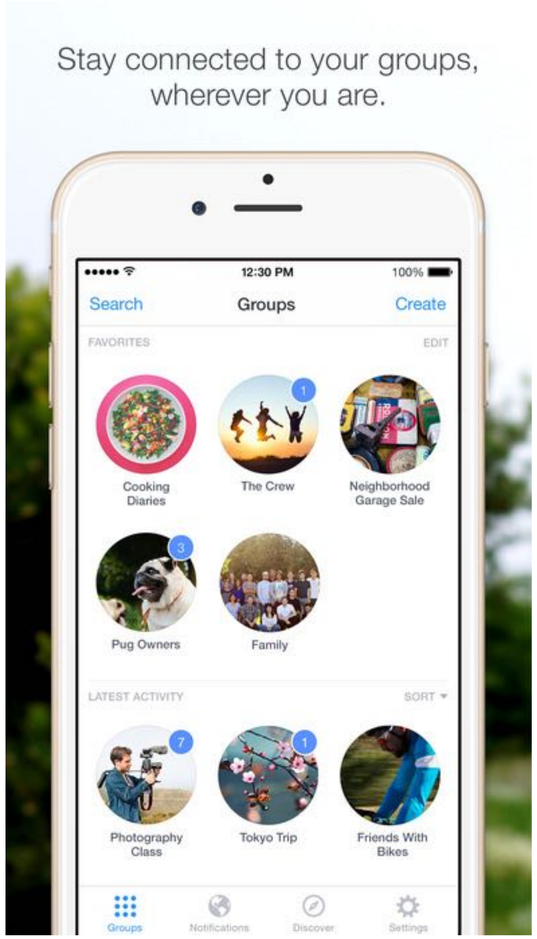


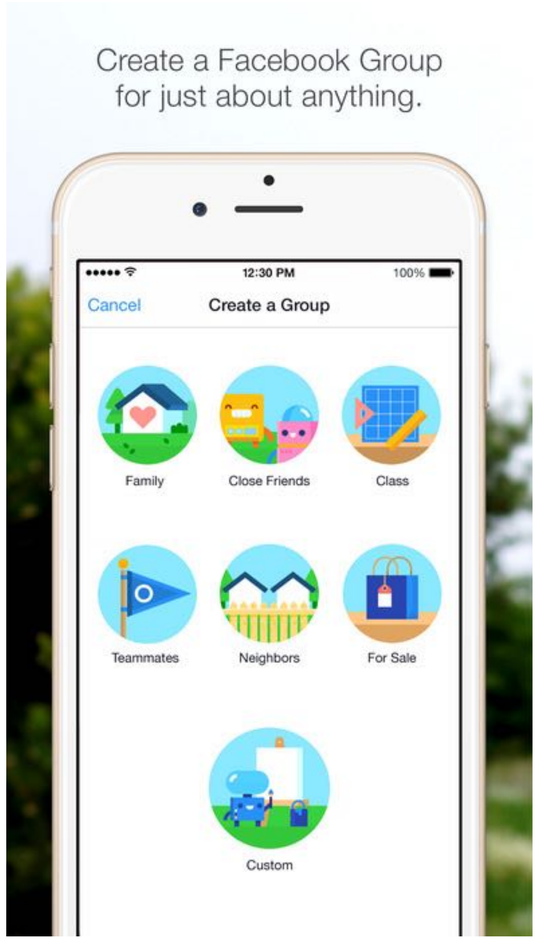
View Insights Promote

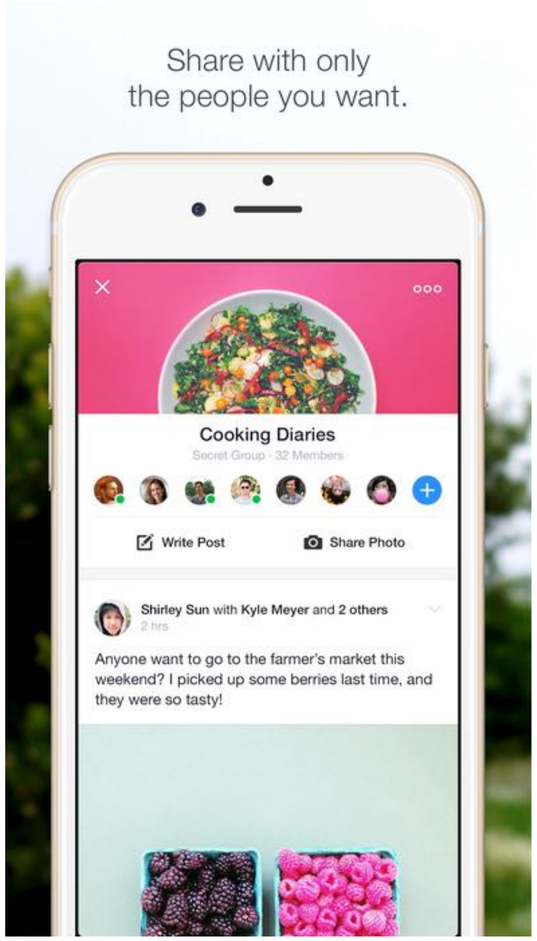
Liked by **dafregenhardt**, **kangax** and **21 others**

View all 4 comments

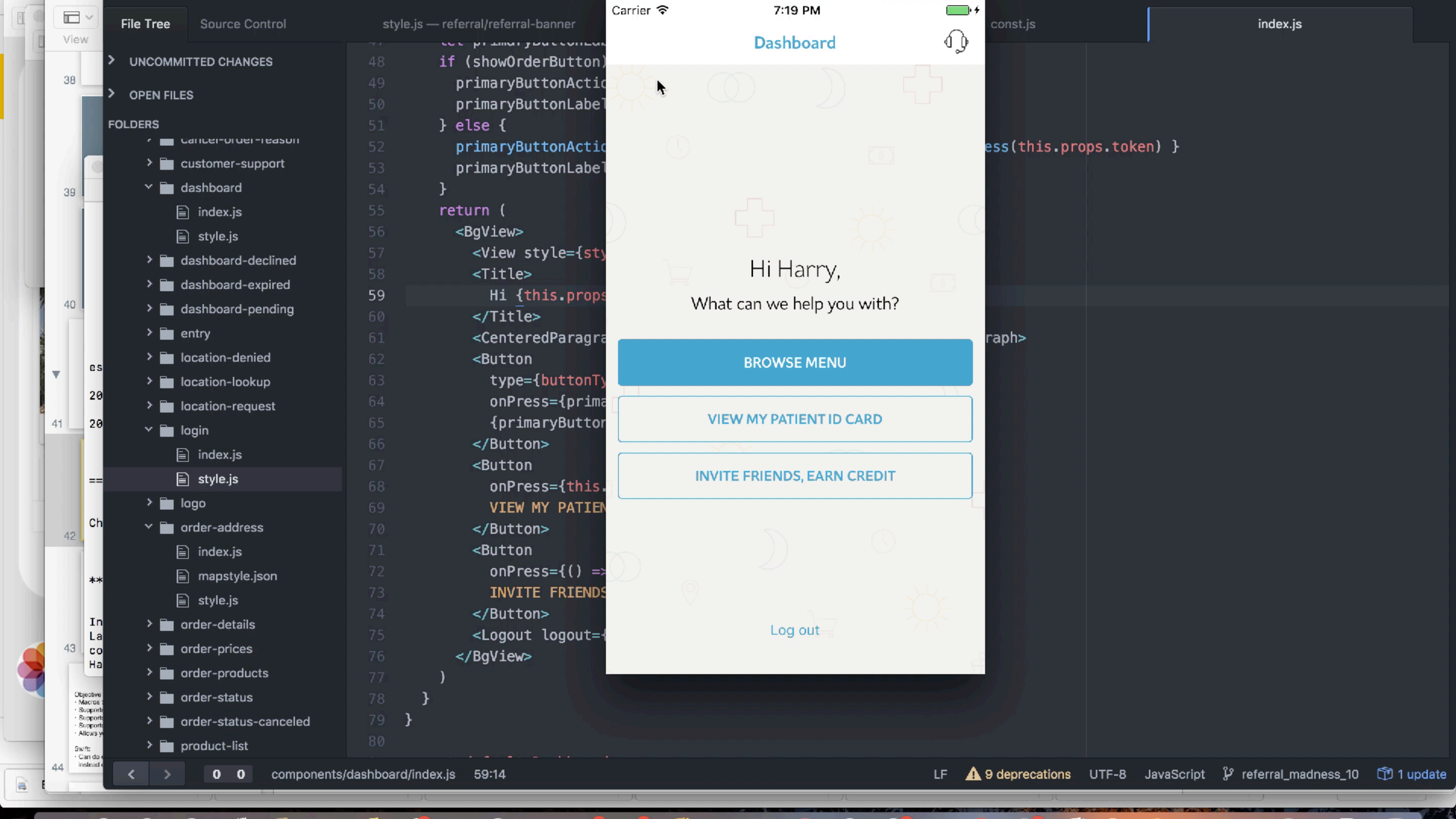
Facebook Groups







Hot reloading


File Tree
Source Control
style.js — referral/referral-banner
const.js
index.js
UNCOMMITTED CHANGES
OPEN FILES
FOLDERS
customer-support
dashboard
index.js
style.js
dashboard-declined
dashboard-expired
dashboard-pending
entry
location-denied
location-lookup
location-request
login
index.js
style.js
logo
order-address
index.js
mapstyle.json
style.js
order-details
order-prices
order-products
order-status
order-status-canceled
product-list
if (showOrderButton)
primaryButtonActio
primaryButtonLabel
} else {
primaryButtonActio
primaryButtonLabel
}
return (
<BgView>
<View style={sty
<Title>
Hi {this.props
</Title>
<CenteredParagra
<Button
type={buttonTy
onPress={prima
{primaryButtor
</Button>
<Button
onPress={this.
VIEW MY PATIEN
</Button>
<Button
onPress={() =>
INVITE FRIENDS
</Button>
<Logout logout={
</BgView>
)
}
}
ess(this.props.token) }
raph>
Carrier
7:19 PM
Dashboard
Hi Harry,
What can we help you with?
BROWSE MENU
VIEW MY PATIENT ID CARD
INVITE FRIENDS, EARN CREDIT
Log out
components/dashboard/index.js 59:14
LF
9 deprecations
UTF-8
JavaScript
referral_madness_10
1 update

Cross platform apps

Cross platform apps

- Web: reuse 100% API code, some business logic, no UI without third party libraries
- Android: 90% of our code base is shared consumer
- Android: 70% of our code base is shared driver
- Easy to special case by platform (i.e isAndroid ? do foo : do bar)

Over the air updates

Over the air updates

- Javascript push allowed by App Store TOS
- Can’t push native changes
- Easy to tie rollouts to binary version
- Easy to rollback releases or do staged rollouts

Team productivity

The case against React Native

Ramp up time

Ramp up time

- Even experienced native engineers will take 1-3 months to be writing “good” RN code
- Third party ecosystem is massive and you need to learn about it to write even moderately complex apps
- Web developers ramp up in under a week but wont be able to help much with native/ perf issues.

Third party libraries

Third party libraries

- Too many choices: i.e 5 different nav stacks!!
- Massive variation in quality
- Redux is awesome but learning it and all its extensions can be intimidating

Android is buggy

Android is buggy

- Not as mature as iOS
- Certain navigation stacks break core things like accessibility
- More likely to run into performance issues/ crashes due to wide spectrum of devices/ OS versions

Javascript