10 Lessons I Learned Building React Native Apps

Who Am I?

Adhithi Ravichandran

http://adhithiravichandran.com/

Why I love React Native?

Lesson 1: Learn React

React

JSX

React

Lifecycle Methods

JSX

React

Lifecycle Methods

JSX

React

Props

State

Lifecycle Methods

JSX

Props

React

State

Composition

Lifecycle Methods

JSX

Props

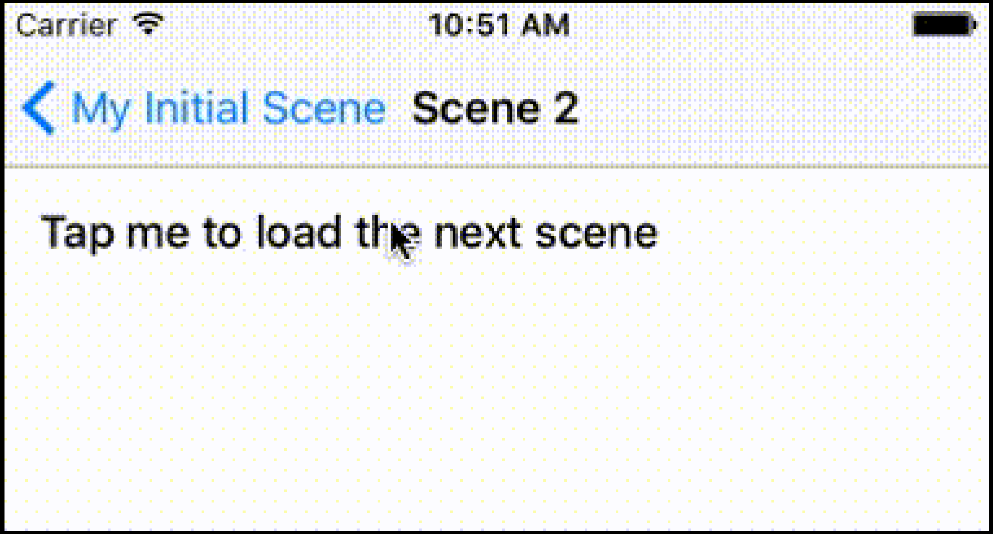
React

Think React

State

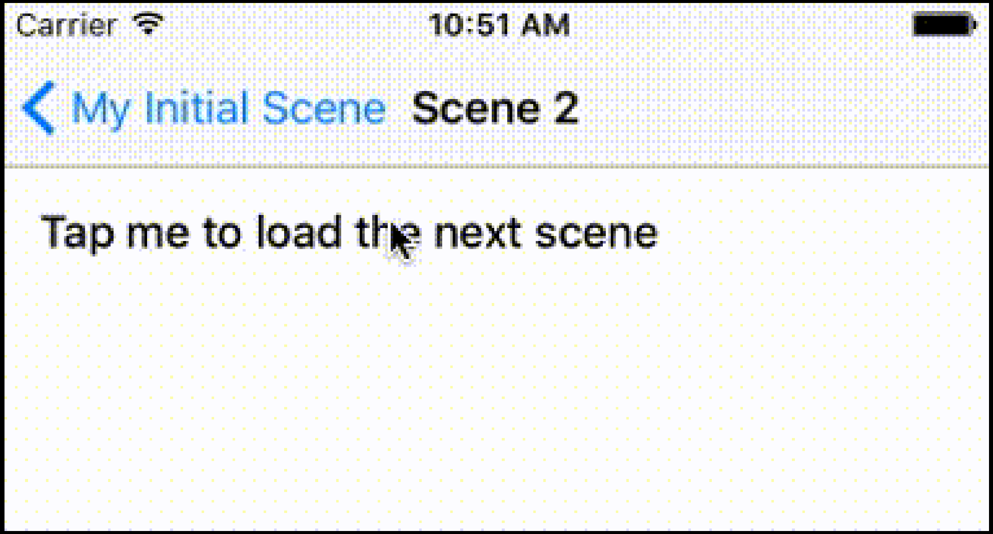
Composition

Lesson 2: Navigation is still evolving

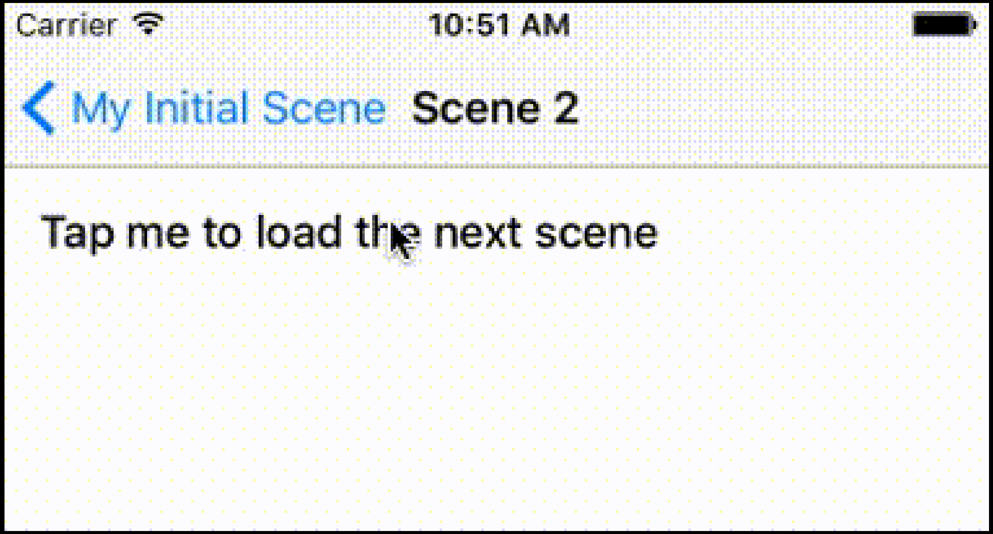


Tap me to load the next scene

NavigationExperimental

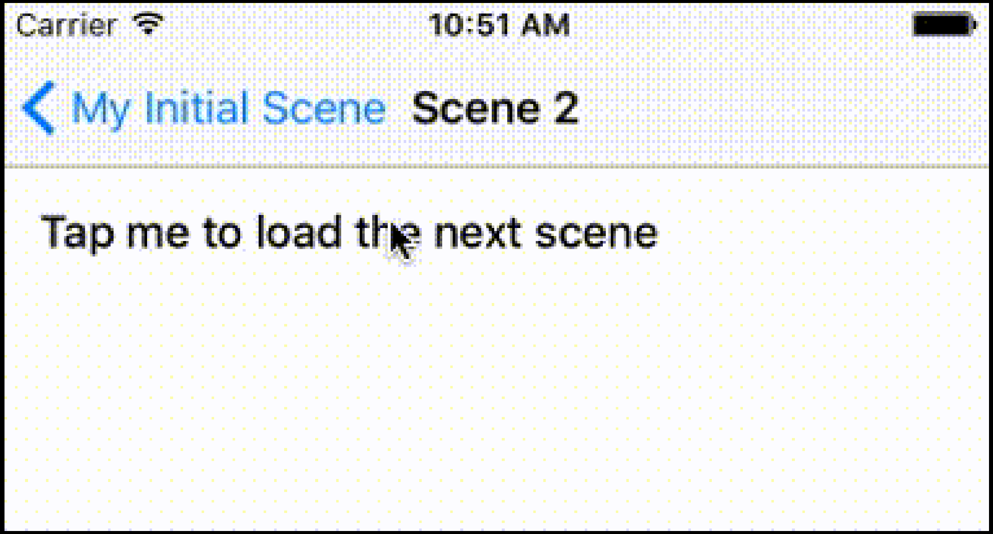


Navigator NavigationExperimental



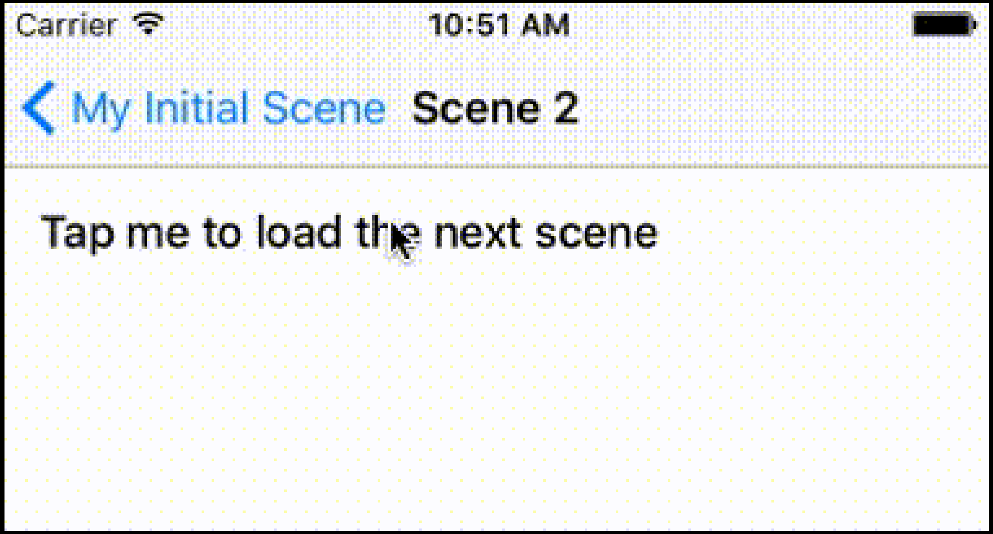
Tap me to load the next scene

NavigationIOS Navigator NavigationExperimental



NavigationIOS Navigator NavigationExperimental

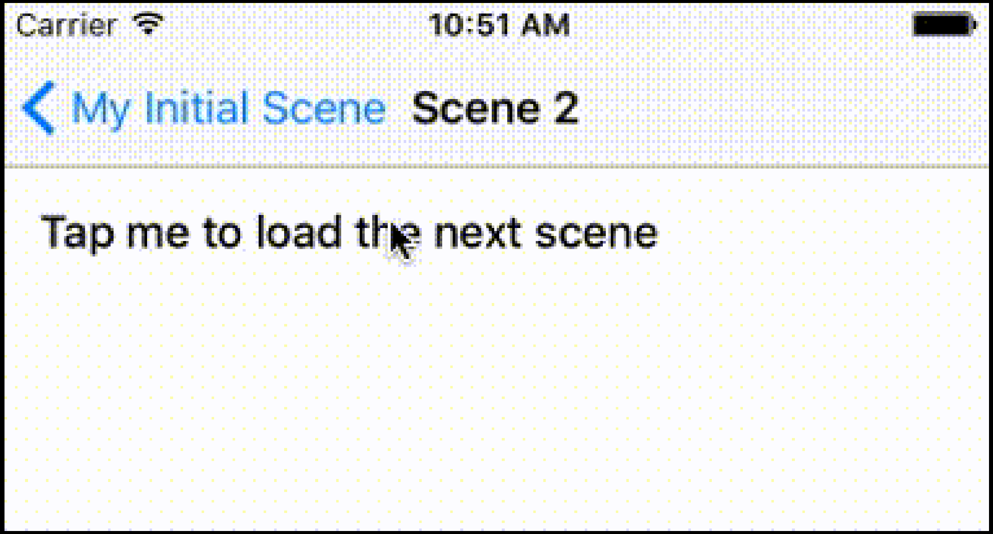
ative Navigation



Tap me to load the next scene

NavigationIOS Navigator NavigationExperimental

ative Navigation

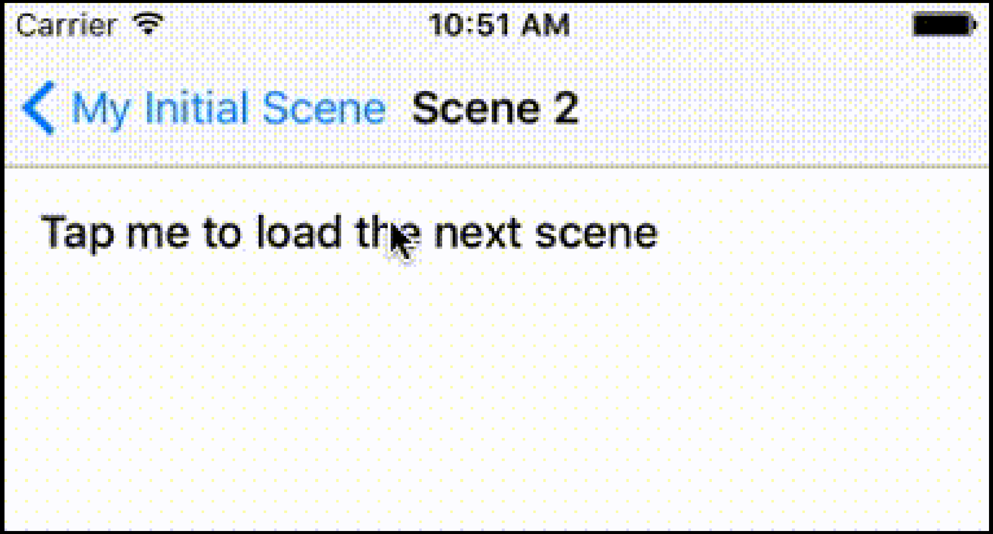

Carrier 10:51 AM

Tap me to load the next scene

React Native Router

NavigationIOS Navigator NavigationExperimental

ative Navigation



Tap me to load the next scene

React Native Router React Native Navigation

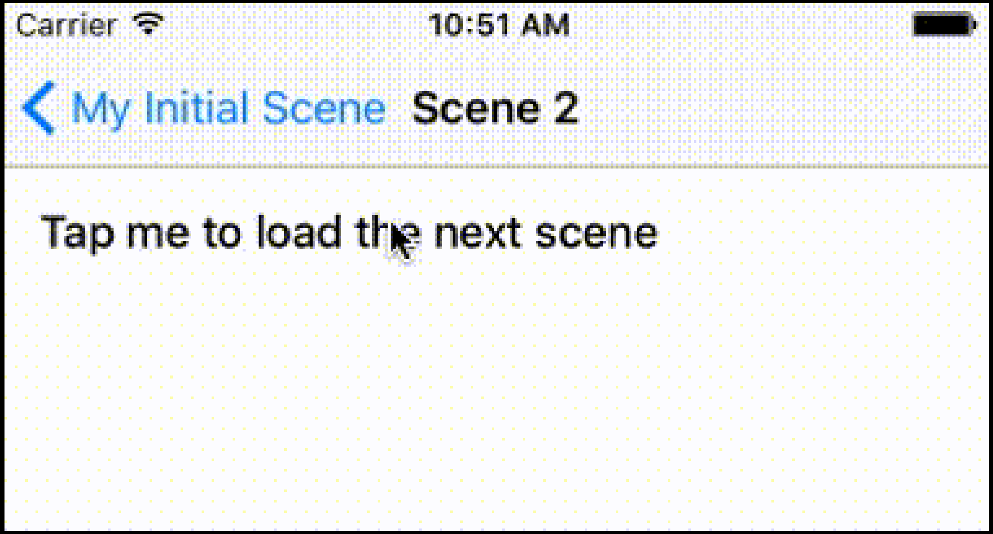
NavigationIOS

Navigator

NavigationExperimental

ative Navigation

React Navigatio



React Native Router

React Native Navigation

BIG MOUTH
NETFLIX

Safe bet if you are starting today

React Navigation

Lesson 3: Use State Management if necessary (Redux)

Redux

State management gets hard to maintain in large apps.

Redux

When to use Redux?

When to use Redux?

- Props passed through hierarchy of components

When to use Redux?

- Props passed through hierarchy of components
- Global components (notifications)

When to use Redux?

- Props passed through hierarchy of components
- Global components (notifications)
- Share application state to multiple container components (user info, session state)

When to use Redux?

- Props passed through hierarchy of components
- Global components (notifications)
- Share application state to multiple container components (user info, session state)
- Caching the page state (text boxes, search fields etc.)

Lesson 4: Don't use a Boilerplate

Don't boilerplates save tons of time?

Don't boilerplates save tons of time?

Yes and No

Don't boilerplates save tons of time?

Yes and No

Great for beginners to start learning the framework and get a quick start.

Why say NO to a boilerplate

Why say NO to a boilerplate

- Loaded

Why say NO to a boilerplate

- Loaded
- Restrictive (Push Notifications)

Why say NO to a boilerplate

- Loaded
- Restrictive (Push Notifications)
- Native Modules and API

Why say NO to a boilerplate

- Loaded
- Restrictive (Push Notifications)
- Native Modules and API
- Detachment

What are some good boilerplates anyway?

IGNITE
react native

Expo

Lesson 5: Don't postpone Android testing

Test Android Devices along with iOS

Android vs iOS

Android vs iOS

- About 90% of the code is shared.

Android vs iOS

- About 90% of the code is shared.
- Small tweaks are needed to support cross-platform.

Android vs iOS

- About 90% of the code is shared.
- Small tweaks are needed to support cross-platform.
- Android "back" button.

Android vs iOS

- About 90% of the code is shared.
- Small tweaks are needed to support cross-platform.
- Android "back" button.
- Some third-party libraries may behave differently on Android vs iOS (Date picker).

Android vs iOS

- About 90% of the code is shared.
- Small tweaks are needed to support cross-platform.
- Android "back" button.
- Some third-party libraries may behave differently on Android vs iOS (Date picker).
- Gradle vs xCode.

TIPS

TIPS

- Automate both Android and iOS releases.

TIPS

- Automate both Android and iOS releases.
- Maximize Devices.

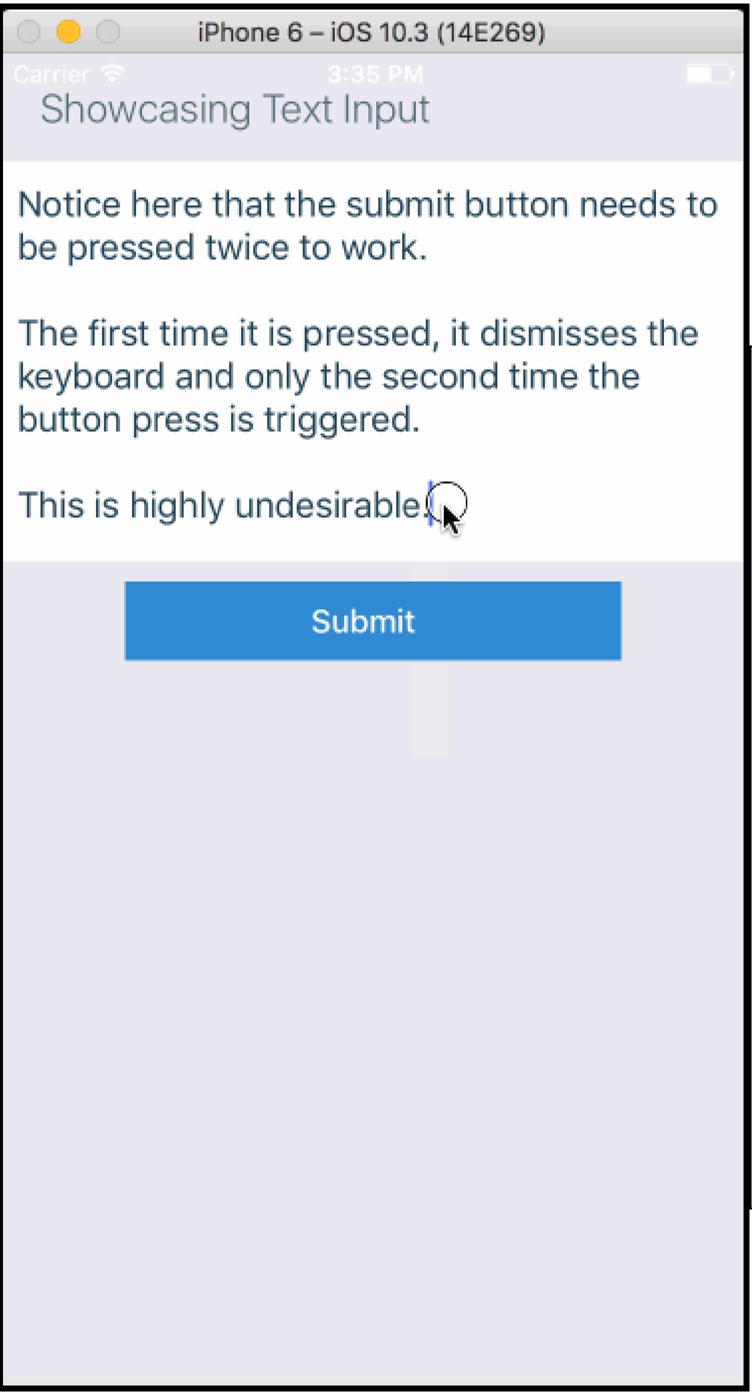
TIPS

- Automate both Android and iOS releases.
- Maximize Devices.
- Test early and often.

TIPS

- Automate both Android and iOS releases.
- Maximize Devices.
- Test early and often.
- Bottom-line: Treat Android and iOS as equals.

Lesson 6: Handling keyboard on TextInput


iPhone 6 – iOS 10.3 (14E269)

Carrier 3:35 PM

Showcasing Text Input

Notice here that the submit button needs to be pressed twice to work.

The first time it is pressed, it dismisses the keyboard and only the second time the button press is triggered.

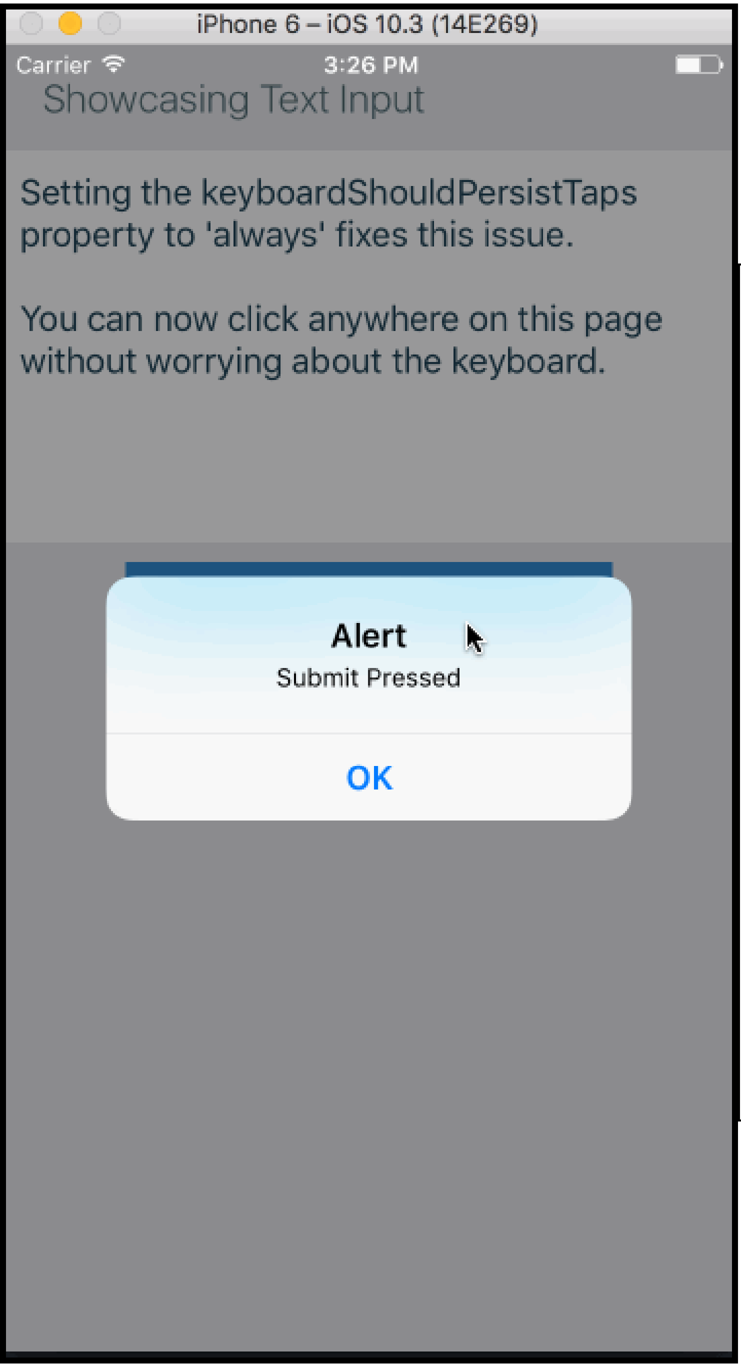
This is highly undesirable.

Buttons need to be pressed twice

```
<ScrollView>

  <TextInput
     style={styles.textContainer}
     placeholder={'Add Your Text Here'}
     textAlignVertical={'top'}
     multiline={true}
     onChangeText={(text) =>
     this.setState({addNoteText: text})}
     value={this.state.addNoteText}
     autoCorrect={true} />

</ScrollView>
```


iPhone 6 – iOS 10.3 (14E269)

Carrier 3:26 PM

Showcasing Text Input

Setting the keyboardShouldPersistTaps property to 'always' fixes this issue.

You can now click anywhere on this page without worrying about the keyboard.

Alert

Submit Pressed

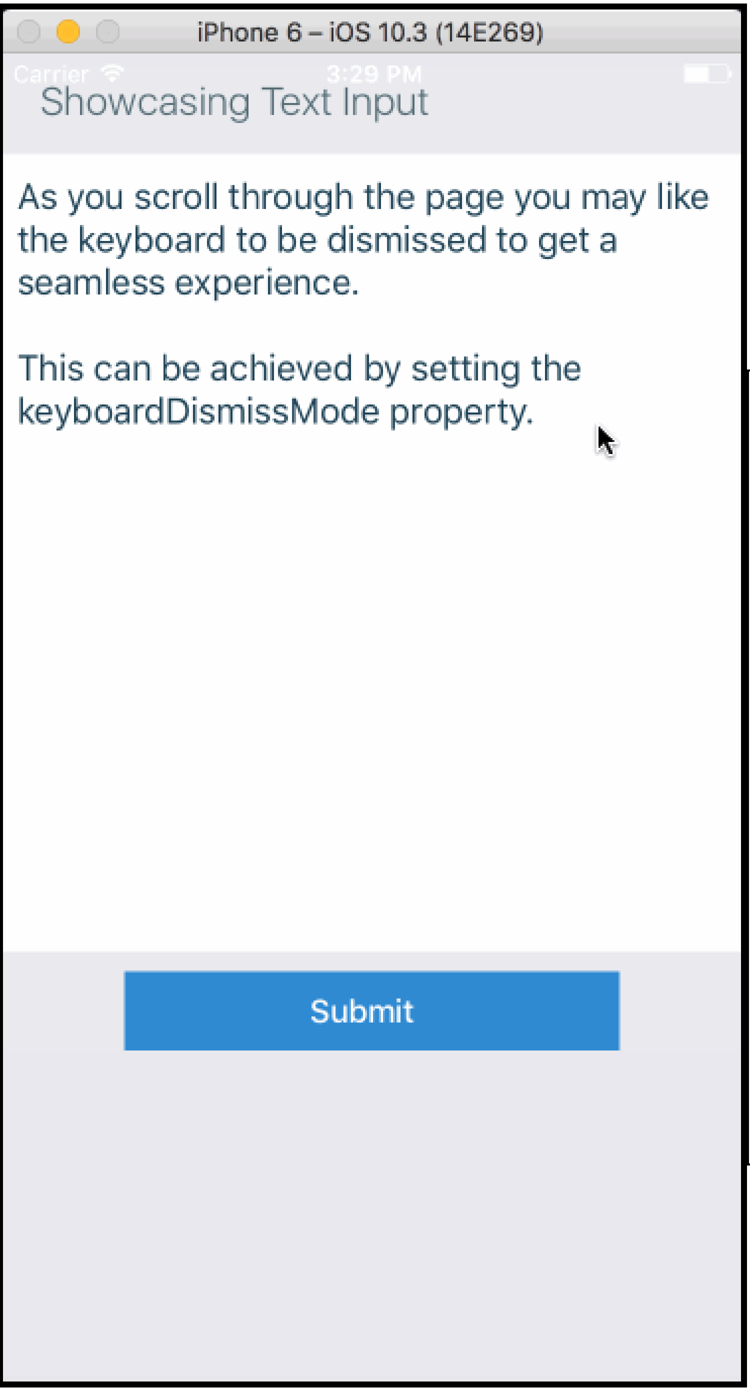
OK

Here is the fix

```
<ScrollView keyboardShouldPersistTaps='a

  <TextInput
     style={styles.textContainer}
     placeholder={'Add Your Text Here'}
     textAlignVertical={'top'}
     multiline={true}
     onChangeText={(text) =>
     this.setState({addNoteText: text})
     value={this.state.addNoteText}
     autoCorrect={true} />

</ScrollView>
```



As you scroll through the page you may like the keyboard to be dismissed to get a seamless experience.

This can be achieved by setting the keyboardDismissMode property.

Dismiss keyboard while scrolling

```
<ScrollView keyboardDismissMode='on-dra

  <TextInput
     style={styles.textContainer}
     placeholder={'Add Your Text Here'}
     textAlignVertical={'top'}
     multiline={true}
     onChangeText={(text) =>
     this.setState({addNoteText: text})
     value={this.state.addNoteText}
     autoCorrect={true} />

</ScrollView>
```

Lesson 7: Use Fastlane for Continuous Delivery

fastlane

By https://docs.fastlane.tools

Distributes beta builds

fastlane

By https://docs.fastlane.tools

Distributes beta builds

Testing

fastlane

By https://docs.fastlane.tools

Distributes beta builds

Testing

fastlane

By https://docs.fastlane.tools

Capture screenshots

Distributes beta builds

Testing

fastlane

By https://docs.fastlane.tools

Capture screenshots

Code signing

Distributes beta builds

Testing

fastlane

By https://docs.fastlane.tools

Capture screenshots

App store deployment

Code signing

Distributes beta builds

Testing

fastlane

iOS/Android

By https://docs.fastlane.tools

Capture screenshots

App store deployment

Code signing

Distributes beta builds

Tons of hours saved

Testing

fastlane

iOS/Android

By https://docs.fastlane.tools

Capture screenshots

App store deployment

Code signing

Beta Deployment with Fastlane

Beta Deployment with Fastlane

Testflight

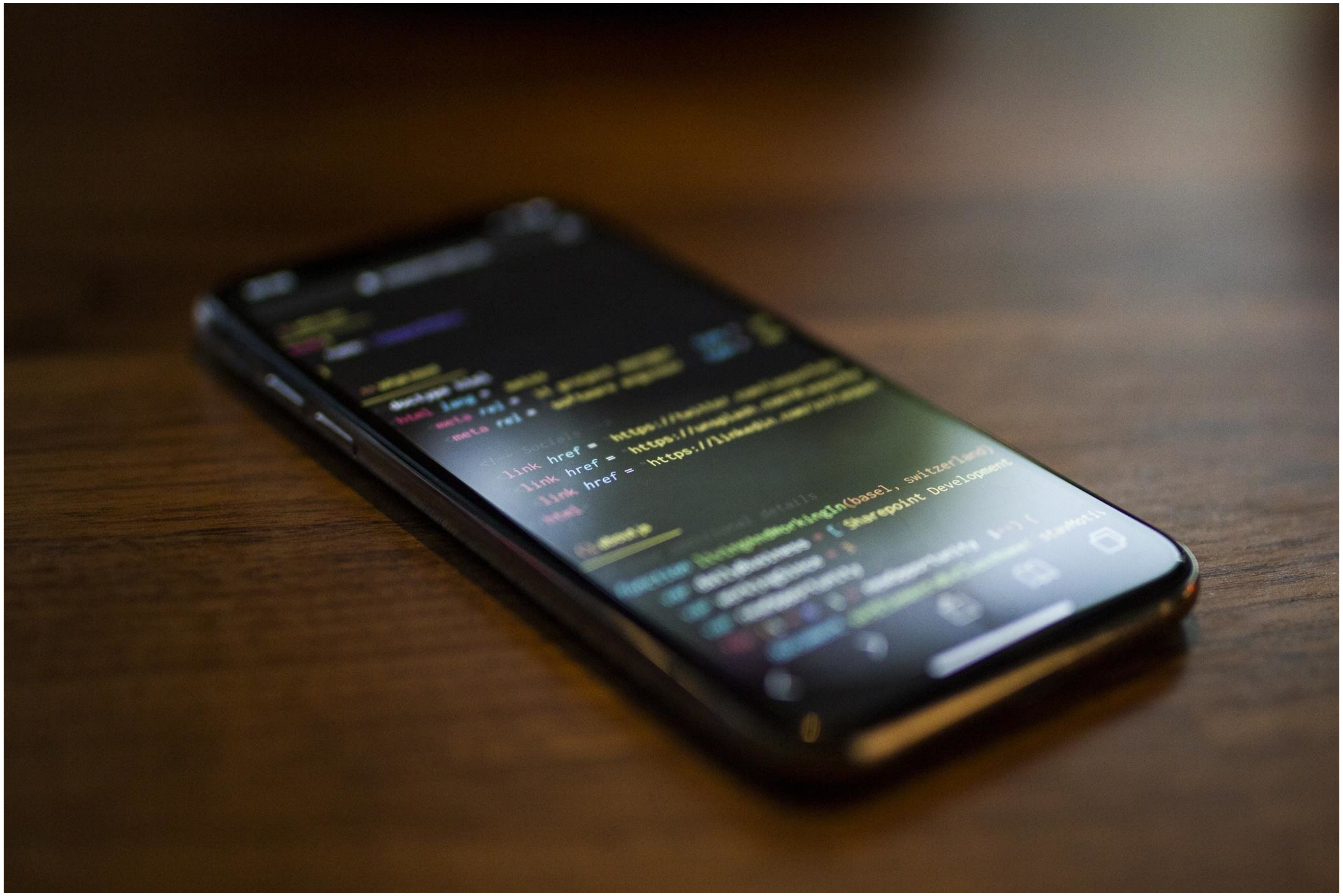
Beta Deployment with Fastlane

Testflight

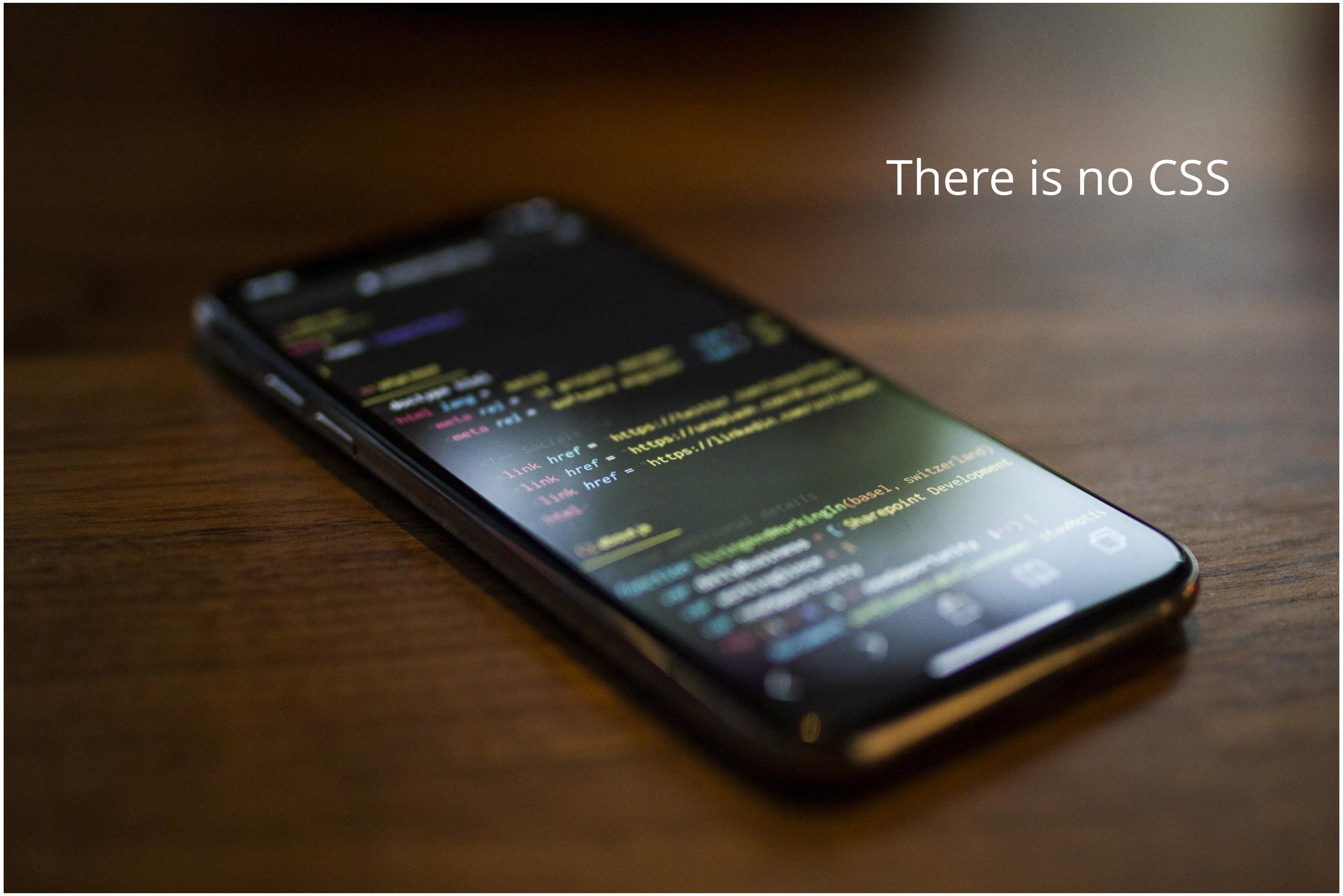
Google Play or Crashlytics

Lesson 8: Styling in React Native

Btw styling works differently in React Native



Btw styling works differently in React Native


There is no CSS

Stylesheets are scoped to Components

Stylesheets are scoped to Components

```
const listOneStyles = StyleSheet.create({
  listContainer: {
    backgroundColor: 'red'
  }
});

const listTwoStyles = StyleSheet.create({
  listContainer: {
    backgroundColor: 'yellow'
  }
});

<ListOne style={listOneStyles.listContainer}/>
<ListTwo style={listTwoStyles.listContainer}/>
```

Stylesheets are scoped to Components

```
const listOneStyles = StyleSheet.create({
  listContainer: {
    backgroundColor: 'red'
  }
});

const listTwoStyles = StyleSheet.create({
  listContainer: {
    backgroundColor: 'yellow'
  }
});

<ListOne style={listOneStyles.listContainer}/>
<ListTwo style={listTwoStyles.listContainer}/>
```

- Components and Styles are tightly coupled

Stylesheets are scoped to Components

```
const listOneStyles = StyleSheet.create({
  listContainer: {
    backgroundColor: 'red'
  }
});

const listTwoStyles = StyleSheet.create({
  listContainer: {
    backgroundColor: 'yellow'
  }
});

<ListOne style={listOneStyles.listContainer}/>
<ListTwo style={listTwoStyles.listContainer}/>
```

- Components and Styles are tightly coupled
- Styles are NOT inherited by default

Sharing Styles - Code Custom Components

Sharing Styles - Code Custom Components

```
import React from 'react'
import PropTypes from 'prop-types'
import { View, Text } from 'react-native'
import styles from './Styles/HeadingTextStyle'

export default class HeadingText extends React.Component {
    render () {
        return (
            <View style={styles.container}>
                <Text style={styles.titleText}>{this.props.titleText}</Text>
            </View>
            )
        }
    }
}

// Prop type warnings
ViewTitle.propTypes = {
    titleText: PropTypes.string
}
```

Sharing Styles - Code Custom Components

Using the custom component

```
/* MyComponent.js */

import HeadingText from 'HeadingText';

class MyComponent extends React.Component {
  render() {
    return (
      <View>
        <HeadingText titleText={'My Component heading'}></HeadingText>
        <Text>Body ...</Text>
      </View>
    );
  }
}
```

Create Base Style Sets

Create Base Style Sets

```
/* baseStyles.js */

export default StyleSheet.create({
  textInput: {
    padding: 16,
    borderColor: "#eee",
    borderWidth: 20
  },
});
```

Create Base Style Sets

```
/* baseStyles.js */

export default StyleSheet.create({
  textInput: {
    padding: 16,
    borderColor: "#eee",
    borderWidth: 20
  },
});
```

```
/* MyComponent.js */

import baseStyles from "baseStyles";

class MyComponent extends React.Component {
  render() {
    return (
      <TextInput style={baseStyles.textInput} />
    );
  }
}
```

Leverage Code in your styles

Leverage Code in your styles

```
import { StyleSheet, Dimensions } from 'react-native'

let deviceWidth = (Dimensions.get('window').width)
let iconSize = 24;

export default StyleSheet.create({
icon: {
    height: iconSize,
    width: iconSize,
    borderRadius: iconSize / 2 //make this a circle
},

container: {
    width: deviceWidth - 30,
    maxWidth: 470,
    paddingLeft: 15,
    paddingRight: 15,
    marginBottom: 15
  },
});
```

Lesson 9: Upgrades are not easy

Remember: Before you Upgrade

Remember: Before you Upgrade

- Planning

Remember: Before you Upgrade

- Planning
- Developer's upgrade timelines

Remember: Before you Upgrade

- Planning
- Developer's upgrade timelines
- It could last anywhere between a few hours to a couple of weeks!!!

Steps for upgrade

- Create a new branch off your master.

Steps for upgrade

- Create a new branch off your master.
  - Unlink all packages. (*react-native unlink pkgName*)

Steps for upgrade

- Create a new branch off your master.
  - Unlink all packages. (*react-native unlink pkgName*)
  - Run upgrade command. Upgrade to the next version.(*react-native-git-upgrade <version>)* instructions

Steps for upgrade

- Create a new branch off your master.
  - Unlink all packages. (*react-native unlink pkgName*)
  - Run upgrade command. Upgrade to the next version.(*react-native-git-upgrade <version>)* instructions
  - Resolve Merge conflicts and merge.

Steps for upgrade

- Create a new branch off your master.
  - Unlink all packages. (*react-native unlink pkgName*)
  - Run upgrade command. Upgrade to the next version.(*react-native-git-upgrade <version>)* instructions
  - Resolve Merge conflicts and merge.
  - Re-link your packages.

Steps for upgrade

- Create a new branch off your master.
  - Unlink all packages. (*react-native unlink pkgName*)
  - Run upgrade command. Upgrade to the next version.(*react-native-git-upgrade <version>)* instructions
  - Resolve Merge conflicts and merge.
  - Re-link your packages.
  - Test your app on the emulator.

Steps for upgrade

- Create a new branch off your master.
  - Unlink all packages. (*react-native unlink pkgName*)
  - Run upgrade command. Upgrade to the next version.(*react-native-git-upgrade <version>)* instructions
  - Resolve Merge conflicts and merge.
  - Re-link your packages.
  - Test your app on the emulator.
  - Repeat the above steps and upgrade to the next version.

Steps for upgrade

- Create a new branch off your master.
  - Unlink all packages. (*react-native unlink pkgName*)
  - Run upgrade command. Upgrade to the next version.(*react-native-git-upgrade <version>)* instructions
  - Resolve Merge conflicts and merge.
  - Re-link your packages.
  - Test your app on the emulator.
  - Repeat the above steps and upgrade to the next version.
- Final: Merge to master when upgrade is complete.

Lesson 10: Focus on Performance

Tips to help speed your app

Use FlatList or SectionList

Use FlatList or SectionList

ListView

Using key attribute on list items

Using key attribute on list items

```
class MyComponent extends React.Component {
  render() {
    return this.props.data.map((item, i) => {
      return <Text key={item.id}>{item.title}</Text>
    });
  }
}
```

Using key attribute on list items

```
class MyComponent extends React.Component {
  render() {
    return this.props.data.map((item, i) => {
      return <Text key={item.id}>{item.title}</Text>
    });
  }
}
```

Without a unique key for every list item, React will re-render every time you add/remove an item.

Bind Early, Render Simple

Bind Early, Render Simple

```
class MyComponent extends React.Component {

  constructor(props) {
    super(props);
    this.something = this.something.bind(this);
  }
  something() {
    // Execute function
  }

  render() {
    return <Text onPress={this.something}>Execute something</Text>
  }
}
```

Bind Early, Render Simple

✓

```
class MyComponent extends React.Component {

  constructor(props) {
    super(props);
    this.something = this.something.bind(this);
  }
  something() {
    // Execute function
  }

  render() {
    return <Text onPress={this.something}>Execute something</Text>
  }
}
```

✗

```
render() {
 return {
  <Text onPress={()=>this.something()}>Execute something</Text>
  <Text onPress={this.something.bind(this)}>Execute something</Text>
  }
}
```

10 Lessons I Learned Recap

10 Lessons I Learned Recap

1. Learn React.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).
4. Don't use a Boilerplate.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).
4. Don't use a Boilerplate.
5. Don't postpone Android testing.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).
4. Don't use a Boilerplate.
5. Don't postpone Android testing.
6. Handling keyboard on TextInput.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).
4. Don't use a Boilerplate.
5. Don't postpone Android testing.
6. Handling keyboard on TextInput.
7. Using fastlane for Continuous Delivery.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).
4. Don't use a Boilerplate.
5. Don't postpone Android testing.
6. Handling keyboard on TextInput.
7. Using fastlane for Continuous Delivery.
8. Styling is different in React Native.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).
4. Don't use a Boilerplate.
5. Don't postpone Android testing.
6. Handling keyboard on TextInput.
7. Using fastlane for Continuous Delivery.
8. Styling is different in React Native.
9. Upgrades are not easy.

10 Lessons I Learned Recap

1. Learn React.
2. Navigation is still evolving.
3. Use state management if necessary (Redux).
4. Don't use a Boilerplate.
5. Don't postpone Android testing.
6. Handling keyboard on TextInput.
7. Using fastlane for Continuous Delivery.
8. Styling is different in React Native.
9. Upgrades are not easy.
10. Focus on Performance.

Get on board and create your React Native app!

Questions???