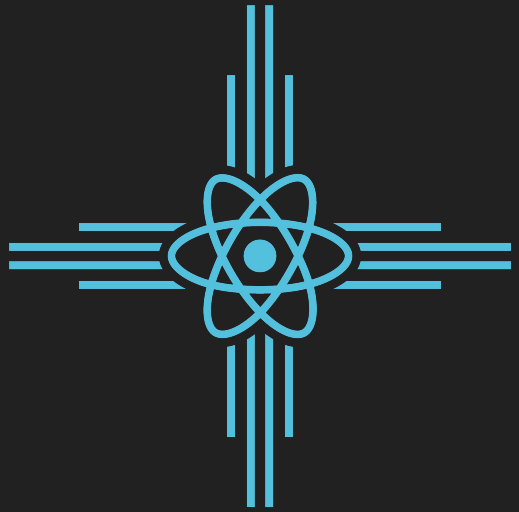


# 505 React

Meetup 6

# Before we begin

Install Node.js for your operating system

https://nodejs.org/en/

# What is React Native?

- React Native is like React, but it uses native components instead of web components as building blocks.

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

export default class HelloWorldApp extends Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: "center", alignItems: "center" }
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

No Errors

Show Details

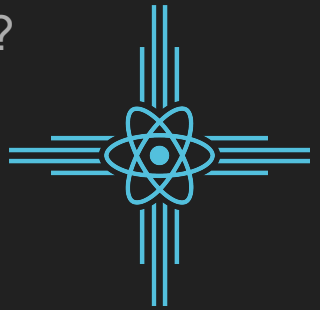
# Deciding how to build React Native

• Will you need access to Native Modules that are handled by traditional native languages such as Swift, Objective-C, Java or Kotlin? (ie. Bluetooth)

• Do you need access to packages or libraries for React Native that do not have Expo support yet? ( You can tell easily on the package's NPM or GitHub docs by looking if it requires React Native Link in the command line to install)

•Will your app grow to need any of these things in the near future?

• Are you comfortable with Xcode and/or Android Studio

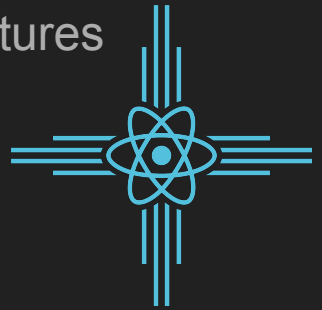


# What is Expo?

- A set of tools, libraries and services you can use to build native iOS and Android apps without having to open Xcode or Android Studio.
- The Expo SDK is a set of libraries written natively for each platform which provides access to the device's system functionality (things like the camera, push notifications, contacts, local storage, and other hardware and operating system APIs) from JavaScript.
- Expo also provides UI components to handle a variety of use-cases that almost all apps will cover but are not built into React Native core, e.g. icons, blur views, and more.

# Expo Pros

- Project setup is fast (No XCode or Android Studio)
- QA is easy to share among several devs/QA team
- No traditional build is necessary to run the app
- Continuous integration is possible
- You have access to basic debugger tools
- You can easily distribute to the Google Play and App Store
- You have access to SOME basic native libraries
- You can eject at anytime and continue using SOME of those features



# Expo Cons

- You cannot add native modules
- You can’t use libraries that require native code (Swift, Java, etc.)
- The app size is larger than normal
- Ejecting can be painful
- Most React Native NPM libraries require native modules

# Getting started

1. You need to have Node installed
2. Install XCode and the iOS simulator
3. Run the following commands in your terminal to start a new React Native project with Expo client:

```
$ npm install -g expo-cli
$ expo init my-project
$ cd my-project
$ npm start
```

You can use yarn instead of npm if you prefer

# Expo configuration

```
? Choose a template: blank
? Choose a workflow: managed
? Please enter a few initial configuration values.
 {
   "expo": {
     "name": "Songs App - Intro To React Native Meetup",
     "slug": "songs-app"
   }
 }
```

# Quick tour of our project + Expo dev tools

- Project files
- Running project in simulator or on a physical device
- Cmd + d to open developer menu (or ctrl + m for android)
- Viewing logs

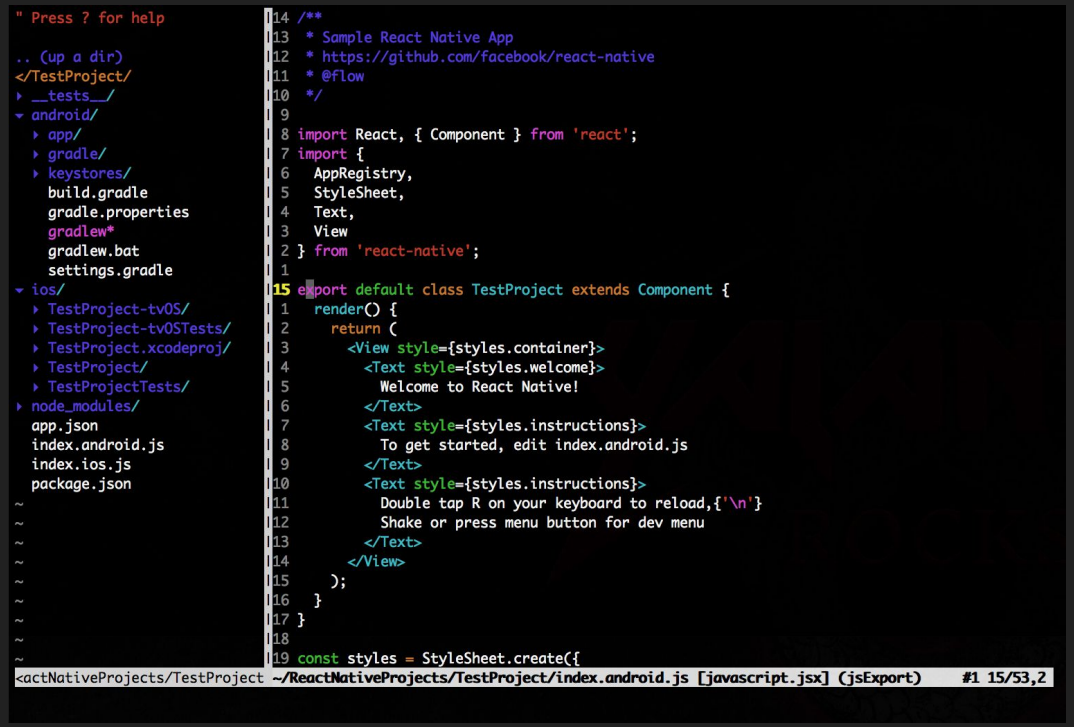
# React Native Components

- Similar to HTML based components
- All come with various default methods and props
- Translate into native components when the project is built

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

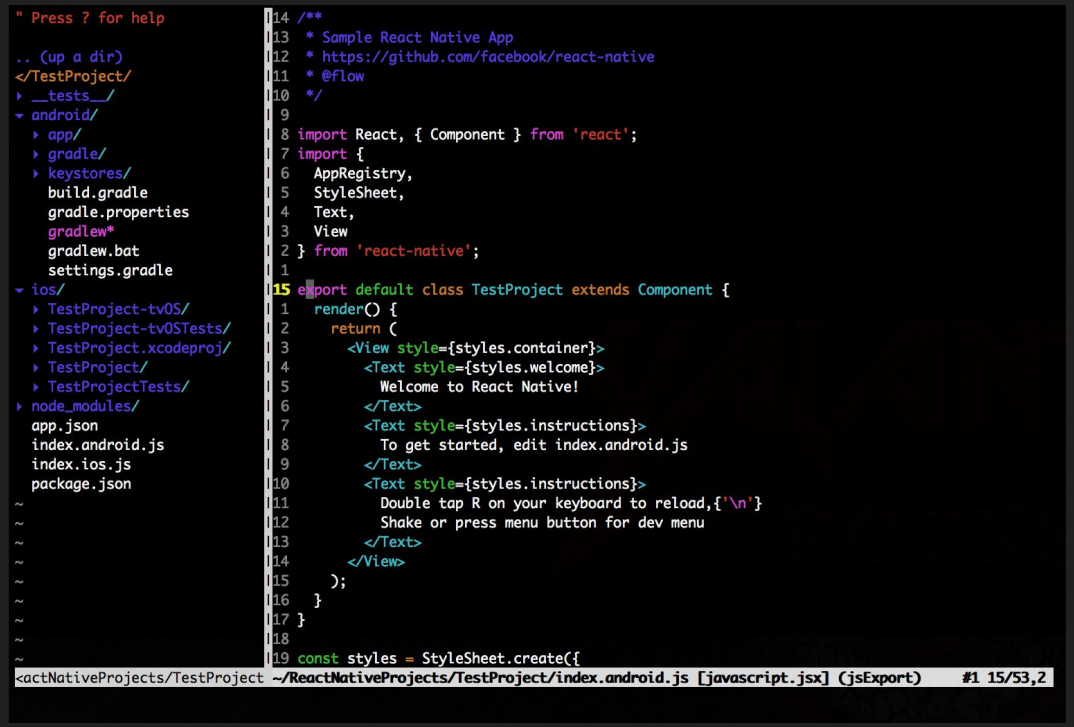
# View

- Similar to HTML 
- Used to hold one more RN components
- CANNOT render text or images, it is solely a container
- Can be styled but it can only control block/container level attributes. No text control



# Text

- Similar to HTML 
- Used to hold Text
- CANNOT have any children other than text
- Can be styled but it can only control text level attributes



# TextInput

- Similar to HTML <input>
- Allows the user to input text
- Is self closing (cannot have children)
- Comes with several methods to handle input using state
    - Most common are onChangeText( ) and onSubmitEditing( )

```
render() {
   return (
      <View style = {styles.container}>
         <TextInput style = {styles.input}
            underlineColorAndroid = "transparent"
            placeholder = "Email"
            placeholderTextColor = "#9a73ef"
            autoCapitalize = "none"
            onChangeText = {this.handleEmail}/>

         <TextInput style = {styles.input}
            underlineColorAndroid = "transparent"
            placeholder = "Password"
            placeholderTextColor = "#9a73ef"
            autoCapitalize = "none"
            onChangeText = {this.handlePassword}/>

         <TouchableOpacity
            style = {styles.submitButton}
            onPress = {
               () => this.login(this.state.email, this.state.password)
            }>
            <Text style = {styles.submitButtonText}> Submit </Text>
         </TouchableOpacity>
      </View>
   )
   }
}
```

# FlatList

- Maps over an array and renders each item as a component you define
- Requires a data prop and renderItem( ) method

```
<FlatList
  ItemSeparatorComponent={Platform.OS !== 'android' && ({highlighted}) => (
    <View style={[style.separator, highlighted && {marginLeft: 0}]} />
  )}
  data={[{title: 'Title Text', key: 'item1'}]}
  renderItem={({item, separators}) => (
    <TouchableHighlight
      onPress={() => this._onPress(item)}
      onShowUnderlay={separators.highlight}
      onHideUnderlay={separators.unhighlight}>
      <View style={{backgroundColor: 'white'}}>
        <Text>{item.title}</Text>
      </View>
    </TouchableHighlight>
  )}
/>
```

# ScrollView

- A scrollable version of RN View

```
return (
  <ScrollView contentContainerStyle={styles.contentContainer}>
  </ScrollView>
);
...
const styles = StyleSheet.create({
  contentContainer: {
    paddingVertical: 20
  }
});
```