Build WhatsApp in a day with

React Native & GraphQL

vnovick.com

Vladimir Novick

Software Architect & Consultant

Dev Advocate at Hasura.io

Web, Mobile, Virtual Reality, Augmented Reality, Mixed Reality & Internet of Things

@VladimirNovick

vnovick.com

Agenda

- Brief overview of React Native
- Creating your React Native app
- Developer tools and debugging
- React Native core components
- Styling React Native components
- FlexBox recap
- Images and SVG Icons in React Native
- Building navigation workflow
- Animations in React Native
- GraphQL Intro
- Creating your GraphQL API with Hasura
- Adding real time capabilities
- Ways of adding your business logic
- Best practices and tips and tricks

Brief Overview of React Native

Assumptions

- You are familiar with JavaScript and Web development
- You've at least heard about ReactJS library
- You are interested in learning React Native with GraphQL

Developer Experience

Experience developer is going through while developing

DX in modern Web

- Hot reload / live reload
- Flexbox layout styling
- Crome dev tools
- Instant visual feedback
- Remote code updates
- Inspecting app structure
- Inspecting network traffic

The ecosystem

- Npm (Node package manager) community packages
- Webpack (bundling, hot module reload, transpilation)
- Modern JavaScript (ES6/7/next)
- Flux architecture (Redux/MobX/MST)

The idea of React Native

“React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI from declarative components.”

Motivation

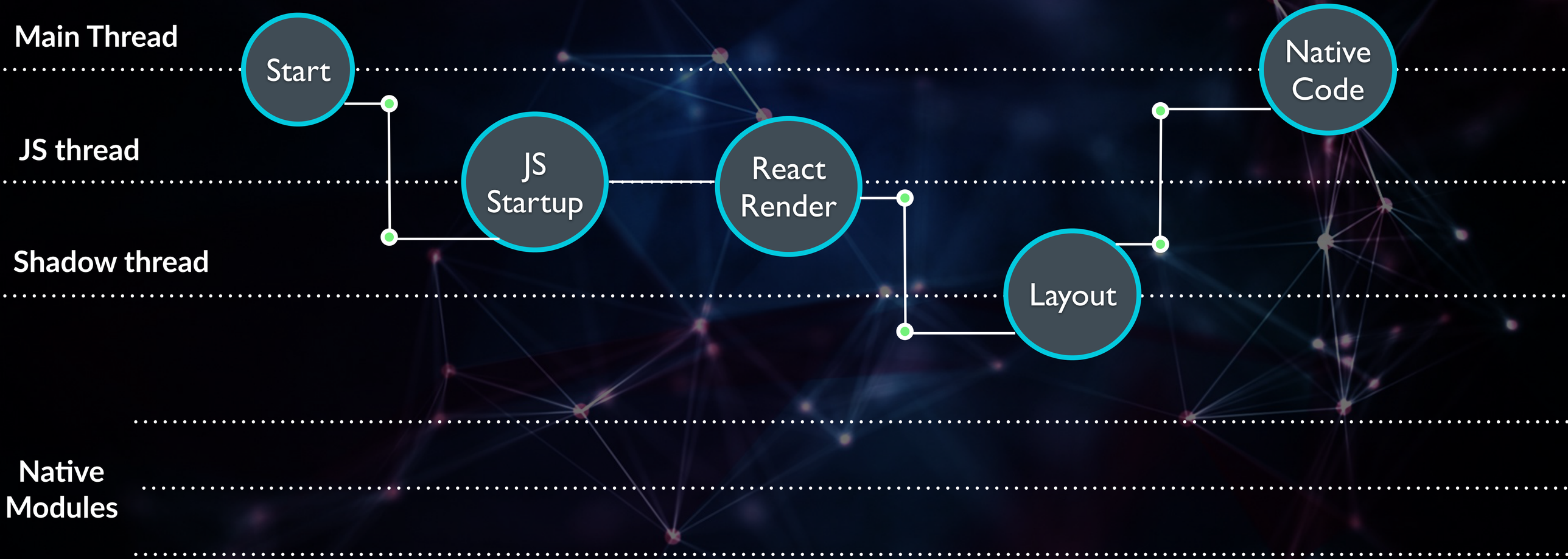
- Code reuse between various platforms Android/iOS/Web
- Smaller dev team is able to handle both Android and iOS
- Sticking to the same ecosystem as on web
- Declarative and more functional programming style
- Styling using modern CSS techniques in mobile world

Architecture

Threading model

- Main Thread - layout, measuring, drawing
- JS thread - event loop executing js code and sending batched updates before next frame renders (60fps)
- Shadow thread - calculates layout changes
- Native modules - platform apis

Execution flow



Getting started

Getting started

EXPO

Good for a start, no need for iOS, Android dependencies

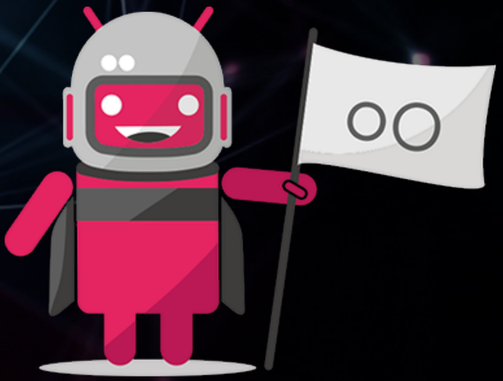
RN CLI

When you want to integrate with existing app, want to write Native code

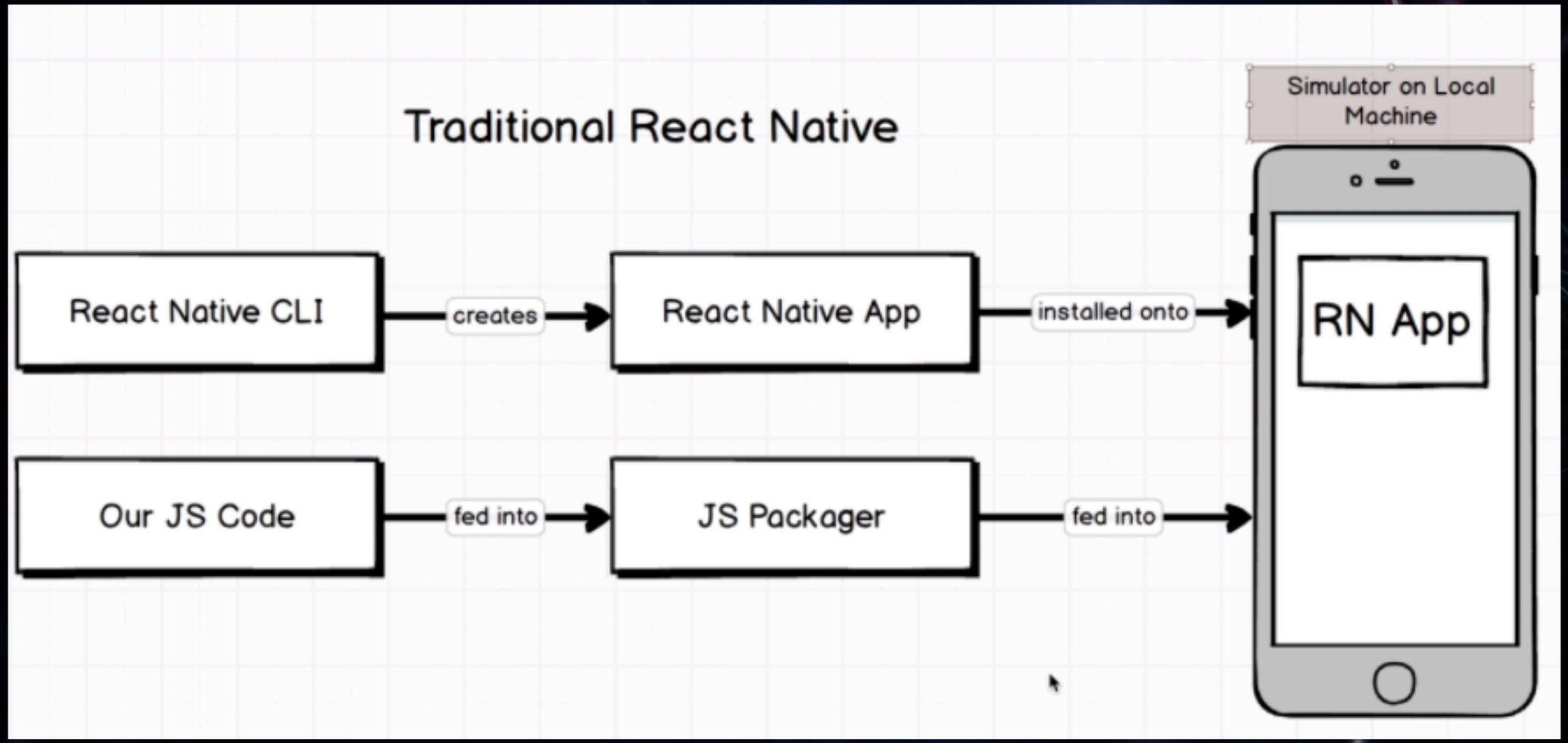
RN CLI

```
» npm install -g react-native-cli
```

Java™



Project lifecycle



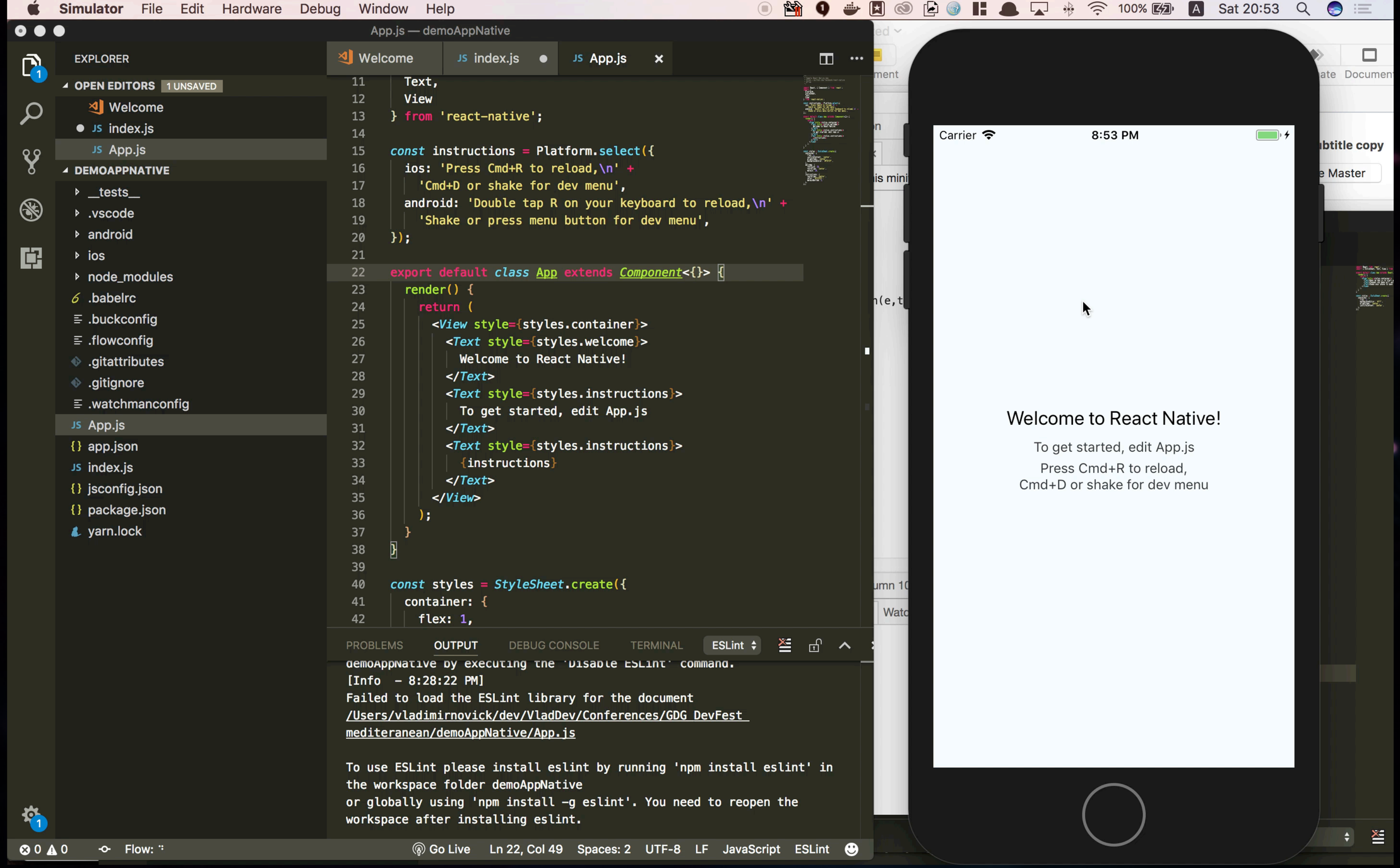
Running your app

Folder structure


__tests__
android
ios
node_modules
.buckconfig
.eslintrc.js
.flowconfig
.gitattributes
.gitignore
.prettierrc.js
.watchmanconfig
App.js
app.json
babel.config.js
index.js
metro.config.js
package.json
Readme.md
yarn.lock
react-native-cli

Code overview


React Native DX
vnovick.com


Simulator File Edit Hardware Debug Window Help
Sat 20:53
App.js — demoAppNative
EXPLORER
OPEN EDITORS 1 UNSAVED
Welcome
index.js
App.js
DEMOAPPNATIVE
__tests__
.vscode
android
ios
node_modules
.babelrc
.buckconfig
.flowconfig
.gitattributes
.gitignore
.watchmanconfig
App.js
app.json
index.js
jsconfig.json
package.json
yarn.lock
Welcome
index.js
App.js
Text,
View
} from 'react-native';
const instructions = Platform.select({
ios: 'Press Cmd+R to reload,\n' +
'Cmd+D or shake for dev menu',
android: 'Double tap R on your keyboard to reload,\n' +
'Shake or press menu button for dev menu',
});
export default class App extends Component<{}> {
render() {
return (
<View style={styles.container}>
<Text style={styles.welcome}>
Welcome to React Native!
</Text>
<Text style={styles.instructions}>
To get started, edit App.js
</Text>
<Text style={styles.instructions}>
{instructions}
</Text>
</View>
);
}
}
const styles = StyleSheet.create({
container: {
flex: 1,
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
ESLint
demoAppNative by executing the 'Disable ESLint' command.
[Info - 8:28:22 PM]
Failed to load the ESLint library for the document
/Users/vladimirnovick/dev/VladDev/Conferences/GDG DevFest mediteranean/demoAppNative/App.js
To use ESLint please install eslint by running 'npm install eslint' in the workspace folder demoAppNative
or globally using 'npm install -g eslint'. You need to reopen the workspace after installing eslint.
0 0
Flow:
Go Live
Ln 22, Col 49
Spaces: 2
UTF-8
LF
JavaScript
ESLint
Carrier
8:53 PM
Welcome to React Native!
To get started, edit App.js
Press Cmd+R to reload,
Cmd+D or shake for dev menu

Yoga

Yoga is a cross-platform layout engine enabling maximum collaboration within your team by implementing an API familiar to many designers and opening it up to developers across different platforms.

- Do you know Flexbox? Then you know Yoga
- Supports Java, C#, Objective-C, C
- Styling your app looks like CSS

```
const styles = StyleSheet.creaete({
  message: {
    width: '70%',
    margin: 10,
    padding: 10,
    backgroundColor: 'white',
    borderColor: '#979797',
    borderStyle: 'solid',
    borderWidth: 1,
    borderRadius: 10
  },
  incomingMessage: {
    alignSelf: 'flex-end',
    backgroundColor: '#E1FFC7'
  }
})

const Message = ({ item }) => (
  <View style={[
      styles.message, item.incoming &&
      styles.incomingMessage
    ]}>
    <Text>{item.message}</Text>
  </View>
)
```

Navigation

- reactnavigation.org - If you start from scratch. Considered as standard and is advised in React Native docs
- https://github.com/wix/react-native-navigation - If you integrate React Native in existing app

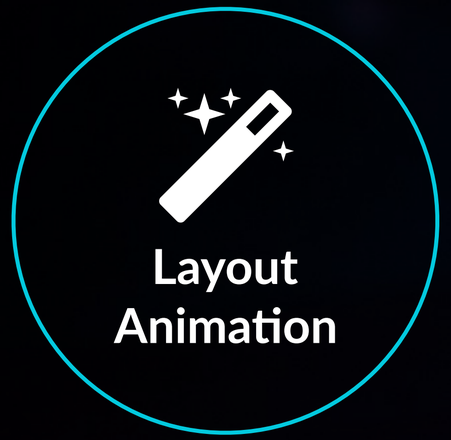
react-navigation

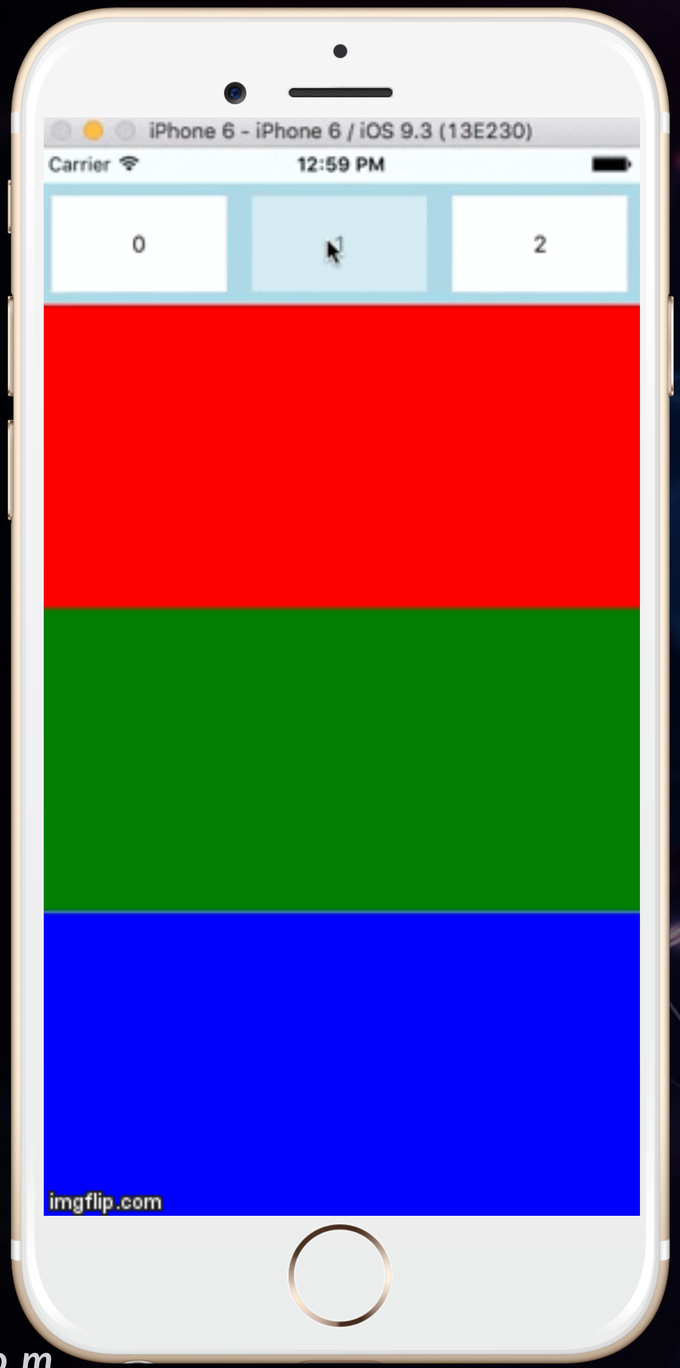
```
import {ChatViewScreen, ConversationsScreen} from './src/
screens';
import {createAppContainer} from 'react-navigation';
import {createStackNavigator} from 'react-navigation-stack';
```

```
const AppContainer = createAppContainer(AppNavigator

const App = () => {
  return (
    <>
      <StatusBar barStyle="light-content" />
      <AppContainer />
    </>
  );
};
```

```
'avigator = createStackNavigator(

sationsScreen: ConversationsScreen,
  chatView: ChatViewScreen,
},
{
  initialRouteName: 'conversationsScreen',
  defaultNavigationOptions: {
    headerStyle: {
      backgroundColor: '#006655',
    },
    headerTintColor: '#fff',
    headerTitleStyle: {
      fontWeight: 'bold',
    },
  },
```

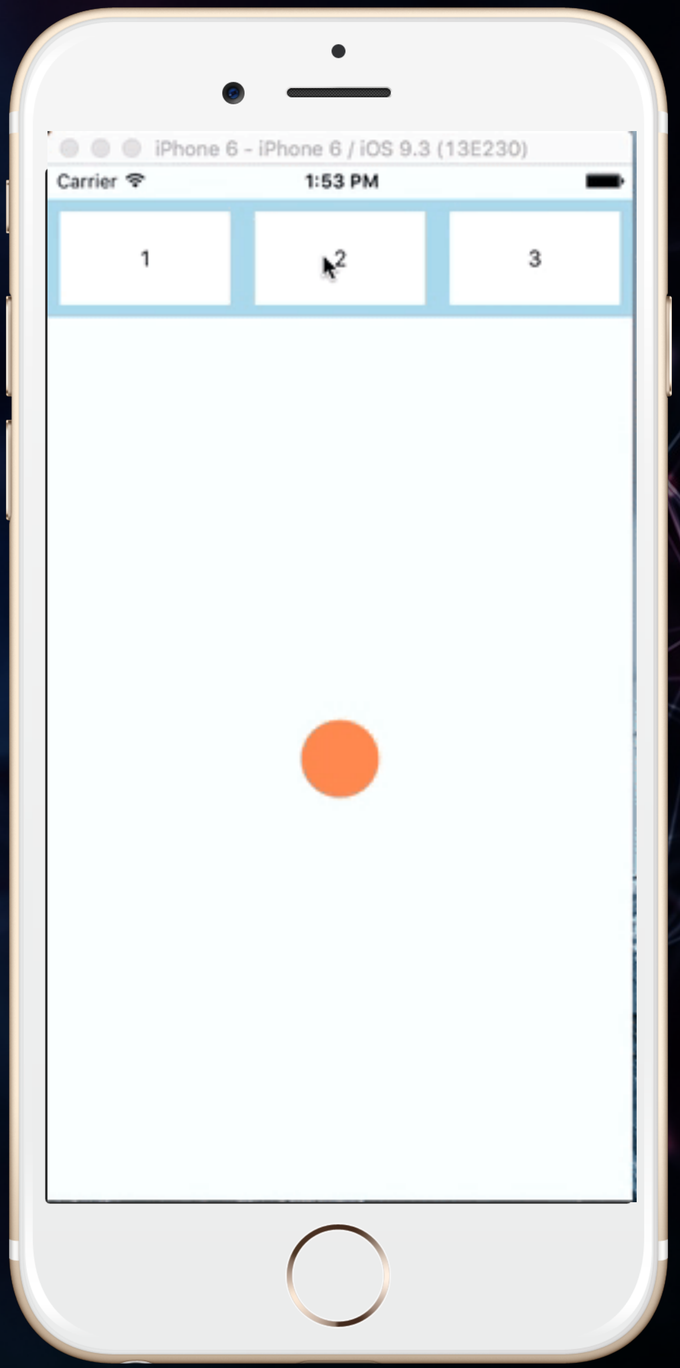

Animations

	Pros	Cons
 **Layout Animation**	✔ Applied to all style properties of component ✔ No need for specific values (heights/ widths)	✔ Less configurable ✔ Animates everything on next render
 **Animated**	✔ Extremely configurable ✔ Can target specific components	✔ Uses requestAnimationFrame by default ✔ For complex computation animations can stutter



Layout
Animatio

Animated



Networking

Dealing with native modules

linking packages with native modules

```
» react-native link
```

Writing your own native modules

Objective-C

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

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

```objc
// MyNativeModule.m
#import "MyNativeModule.h"
#import <React/RCTLog.h>
@implementation MyNativeModule

// To export a module named ExsportedModuleName
RCT_EXPORT_MODULE(ExsportedModuleName);

RCT_EXPORT_METHOD(logInfo : (NSString *)text) {
  RCTLogInfo(@"Logging from my native module %@", text);
}

@end
```

Java

- Create the module by creating class that extends *ReactContextBaseJavaModule*
- Expose methods to JS using @ReactMethod
- Create class that implements ReactPackage
- Override createNativeModules method

JS side

```
import {NativeModules} from 'react-native';
const logger = NativeModules.ExsportedModuleName;

logger.logInfo('Test message');
```

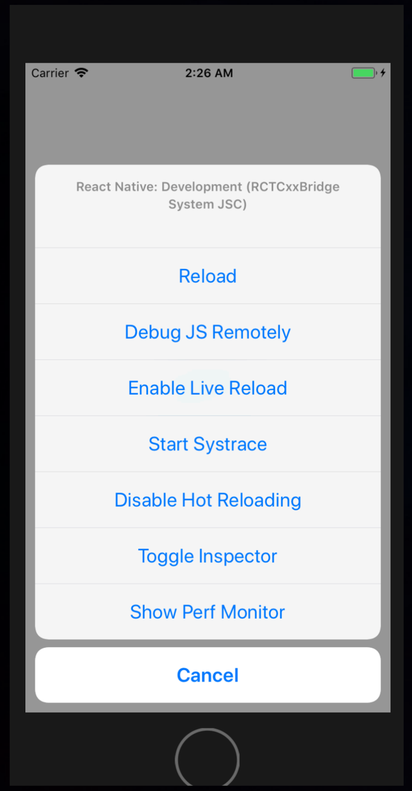
Create your React Native app

Follow the instructions

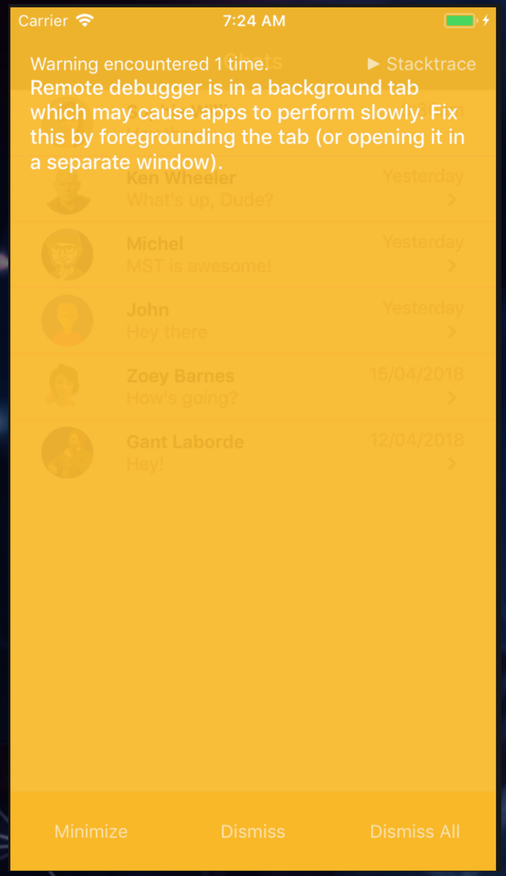
https://github.com/vnovick/whatsapp_workshop

Developer tools and debugging

Debug menu



YellowBoxes



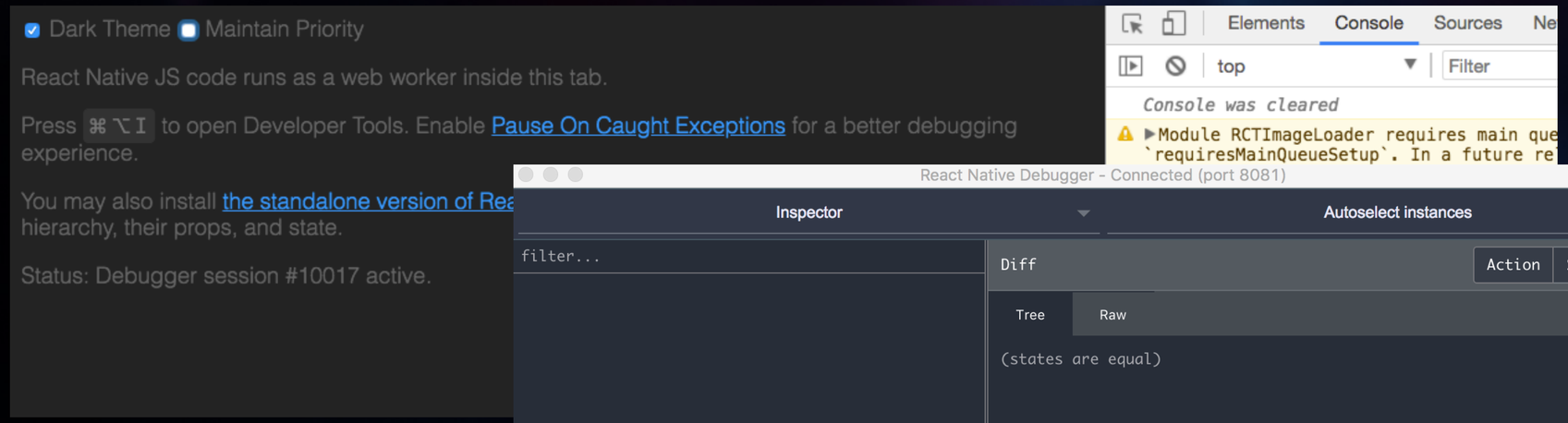
Redboxes


```
react-native log-ios
react-native log-android
```

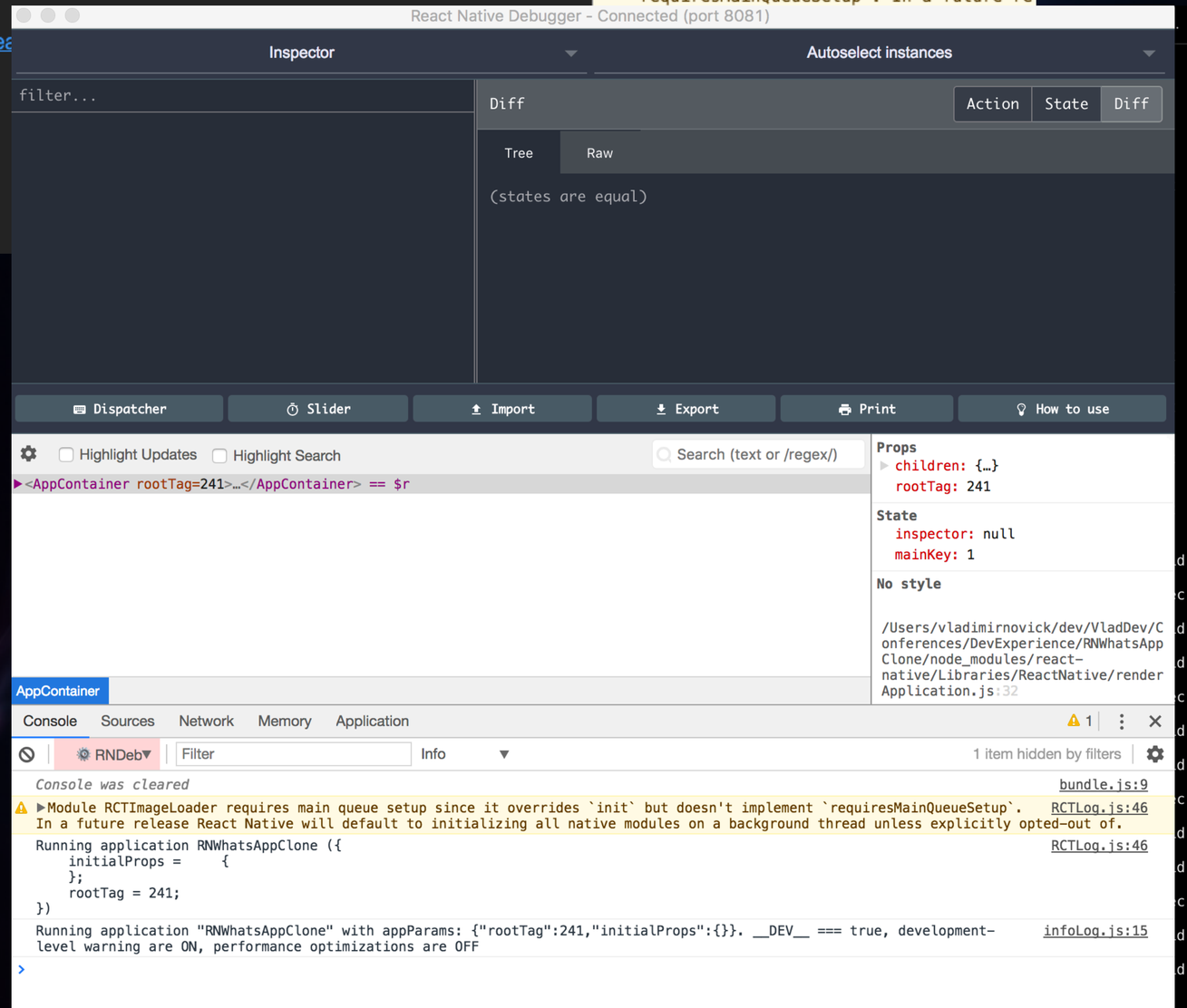
```
eosubscriptionsd) <Notice>: Service only ran for 0 seconds. Pushing respawn out by 10 seconds.
Apr 21 02:24:03 Vladimirs-MacBook-Pro-2 com.apple.CoreSimulator.SimDevice.B67103AA-B6D8-4E86-90E4-944601D21A70[26436] (com.apple.sec
urityuploadd) <Notice>: Service only ran for 6 seconds. Pushing respawn out by 4 seconds.
Apr 21 02:24:10 Vladimirs-MacBook-Pro-2 com.apple.CoreSimulator.SimDevice.B67103AA-B6D8-4E86-90E4-944601D21A70[26436] (com.apple.vid
eosubscriptionsd[96238]) <Warning>: Service exited with abnormal code: 1
Apr 21 02:24:10 Vladimirs-MacBook-Pro-2 com.apple.CoreSimulator.SimDevice.B67103AA-B6D8-4E86-90E4-944601D21A70[26436] (com.apple.vid
eosubscriptionsd) <Notice>: Service only ran for 0 seconds. Pushing respawn out by 10 seconds.
^C
~/dev/VladDev/Conferences/DevExperience/RNWhatsAppClone(step-1*) »                    vladimirnovick@Vladimirs-MacBook-Pro-2
```

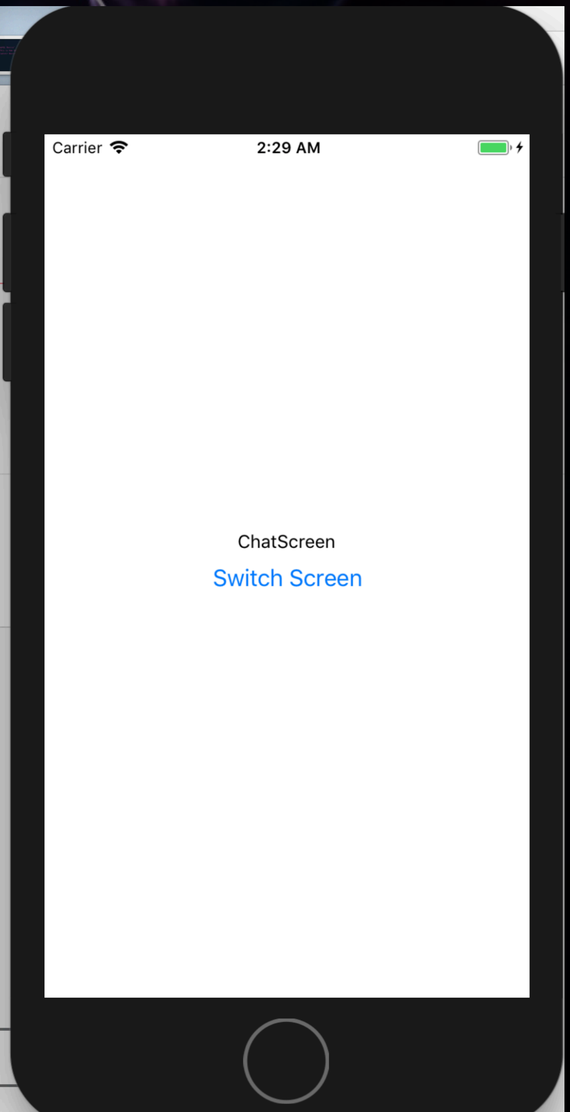
vnovick.com

Chrome dev tools



React Native debugger





Styling React Native components

The style names and values usually match how CSS works on the web, except names are written using camel casing, e.g `backgroundColor` rather than `background-color`.

There are no px/rem/em/vw units. Only numeric values and sometimes %

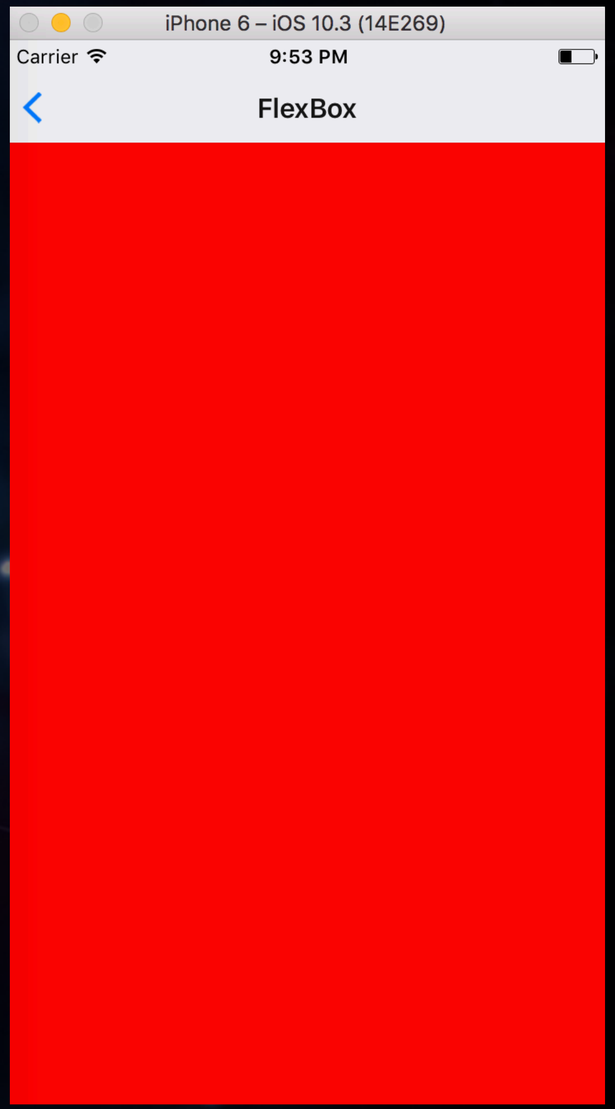
```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center'
  },
  welcome: {
    color: 'white',
    fontSize: 20,
    textAlign: 'center',
    margin: 10,
    width: '80%'
  },
  instructions: {
    color: 'white',
    textAlign: 'center',
    marginBottom: 5,
  },
  workshopInstructions: {
    alignItems: 'flex-start'
  }
});
```

```
<View style={styles.container}>
  { this.renderScreen() }
  <View>
    <Button title="Switch Screen" onPress={() => {
        this.setState({
          isChatsScreen: !this.state.isChatsScreen
        })
      }}
    />
  </View>
</View>
```

we can pass conditional statements and functions to style prop too as soon as they return an object

FlexBox

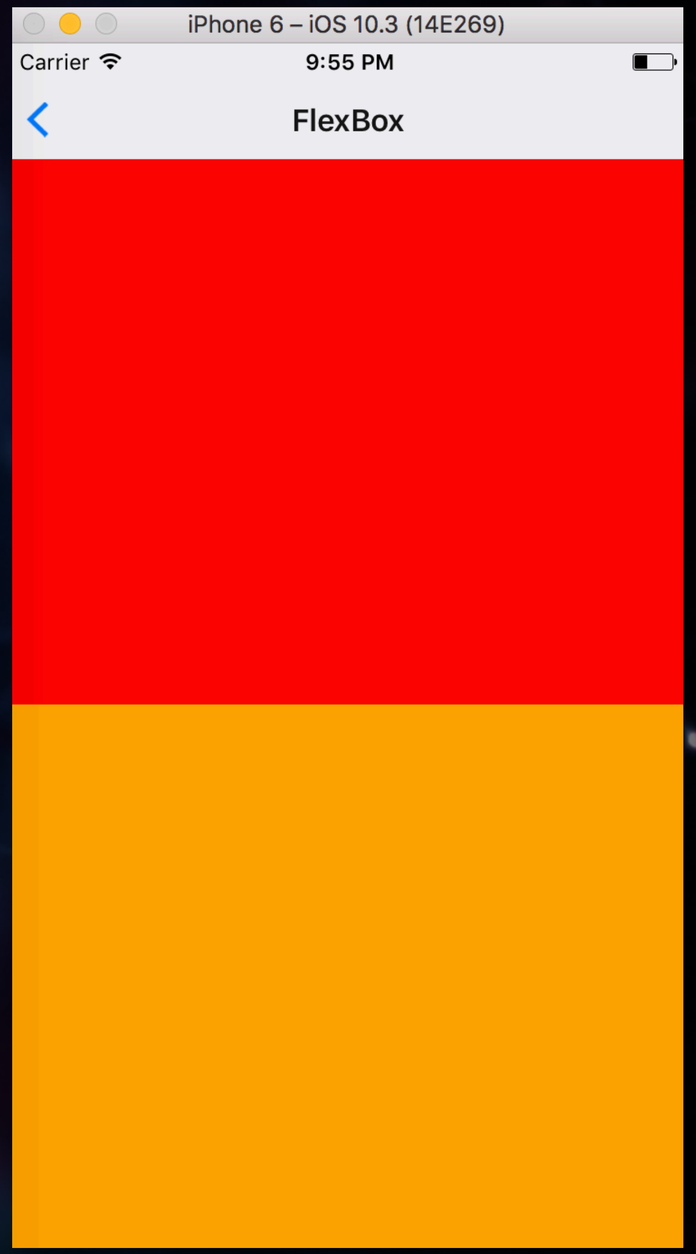
```
container: {
  flex: 1,
},
box1: {
  flex: 1,
  backgroundColor: 'red',
},

<View style={styles.container}>
  <View style={styles.box1} />
</View>
```



https://flexboxfroggy.com/

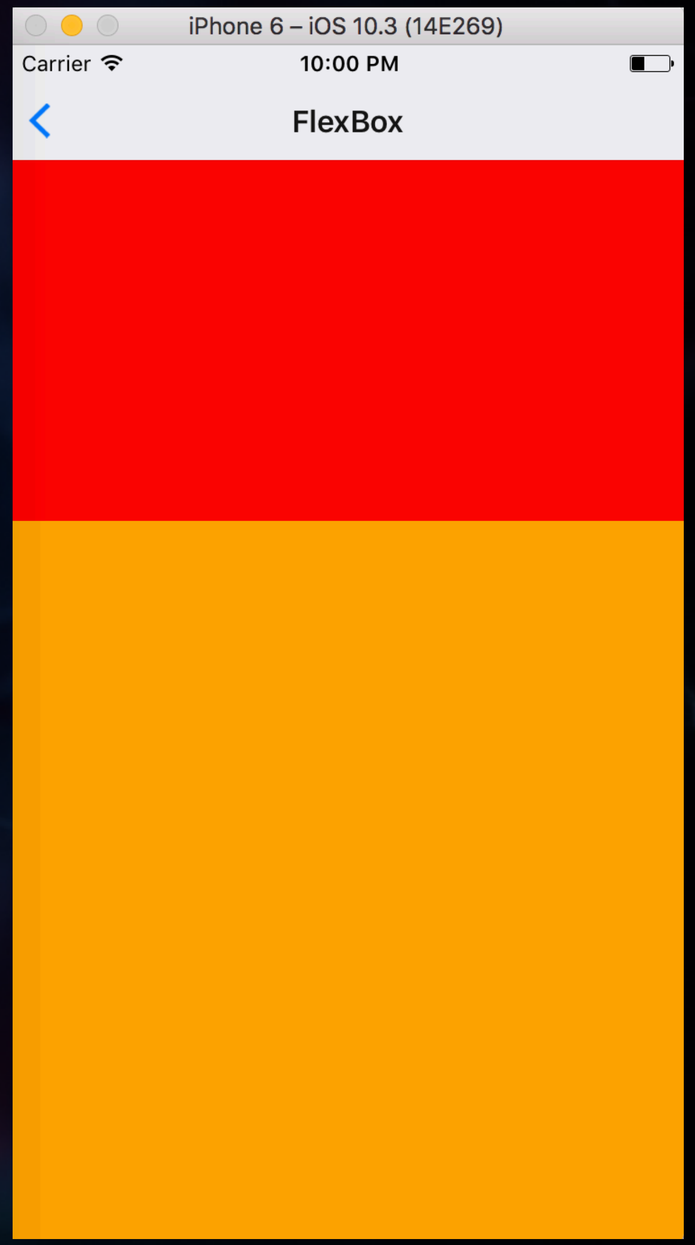
FlexBox

```
container: {
  flex: 1,
},
box1: {
  flex: 1,
  backgroundColor: 'red',
},
box2: {
  flex: 1,
  backgroundColor: 'orange',
},

<View style={styles.container}>
  <View style={styles.box1} />
  <View style={styles.box2} />
</View>
```

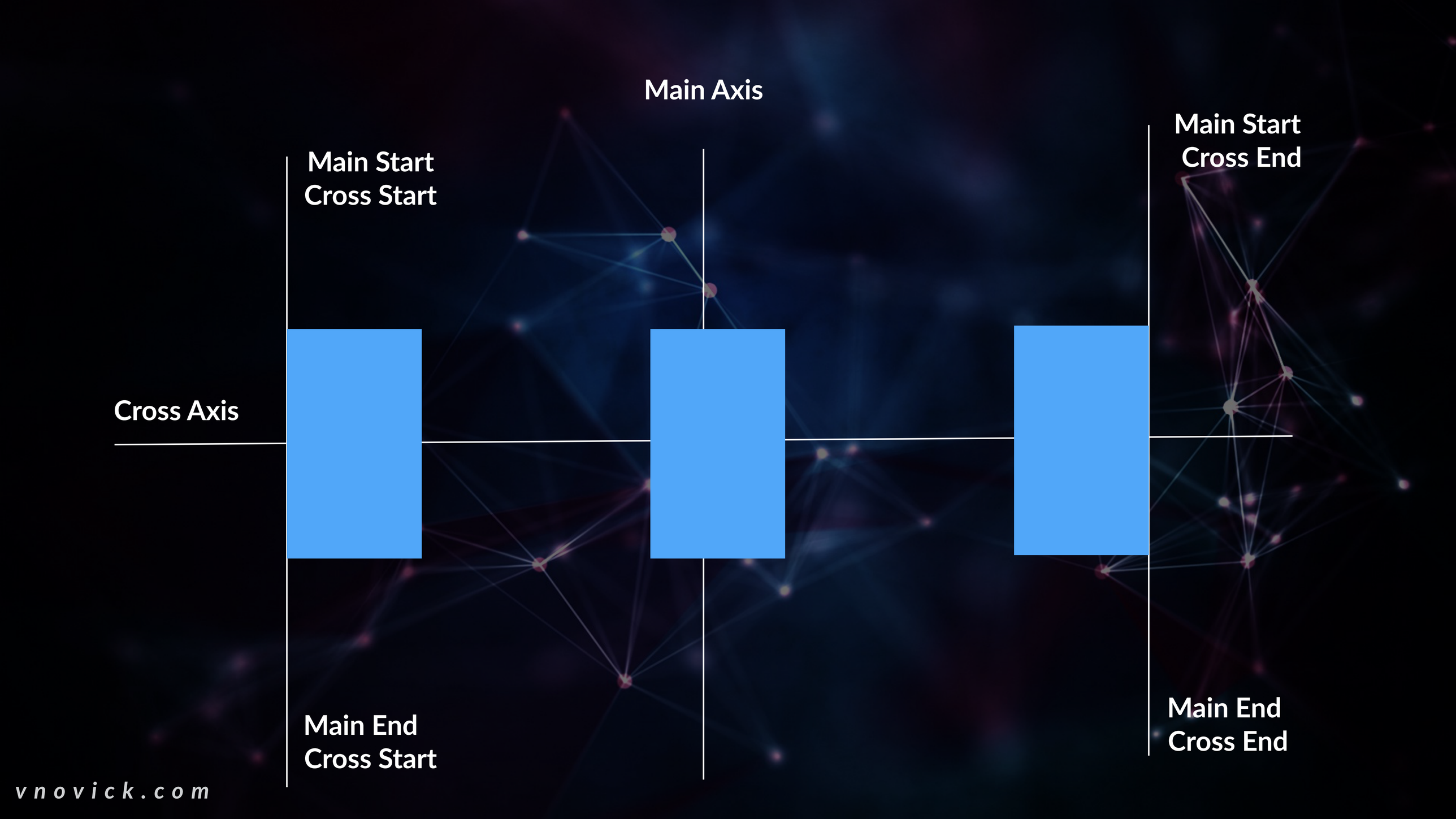


FlexBox

```
container: {
  flex: 1,
},
box1: {
  flex: 1,
  backgroundColor: 'red',
},
box2: {
  flex: 2,
  backgroundColor: 'orange',
},
```



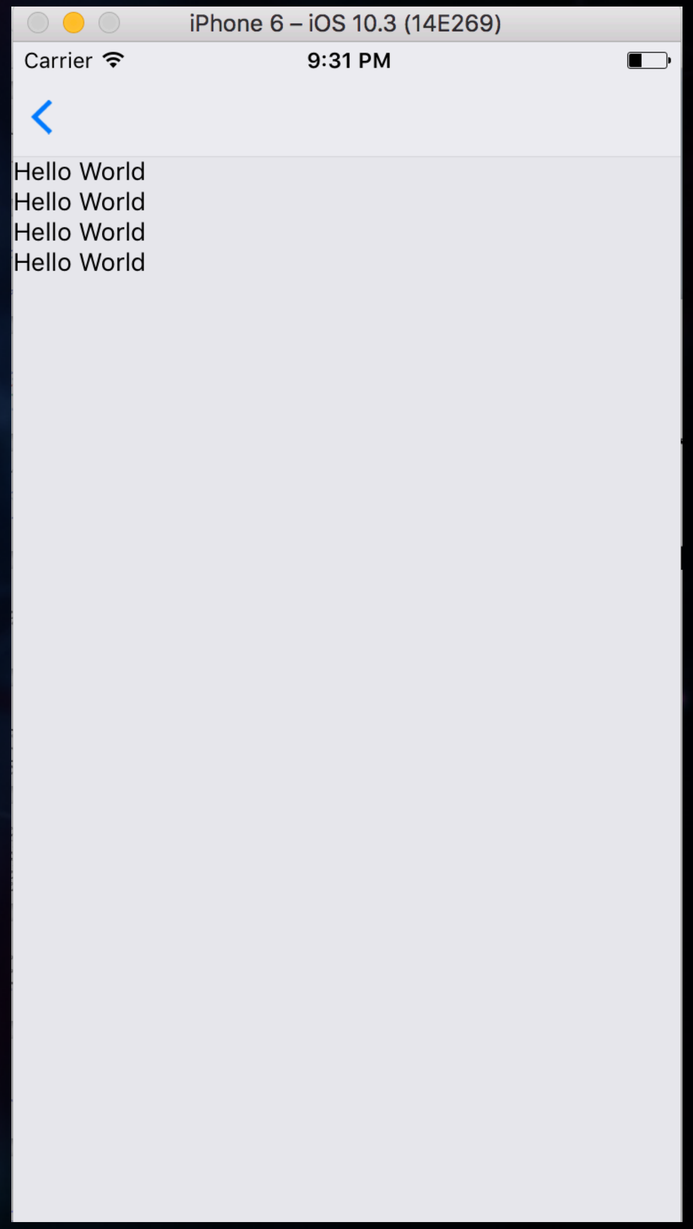

Main Axis
Main Start
Cross Start
Main Start
Cross End
Cross Axis
Main End
Cross Start
Main End
Cross End


Main Axis
Main Start
Cross Start
Main Start
Cross End
Cross Axis
Main End
Cross Start
Main End
Cross End


Main Axis
Main Start
Cross Start
Main Start
Cross End
Cross Axis
Main End
Cross Start
Main End
Cross End

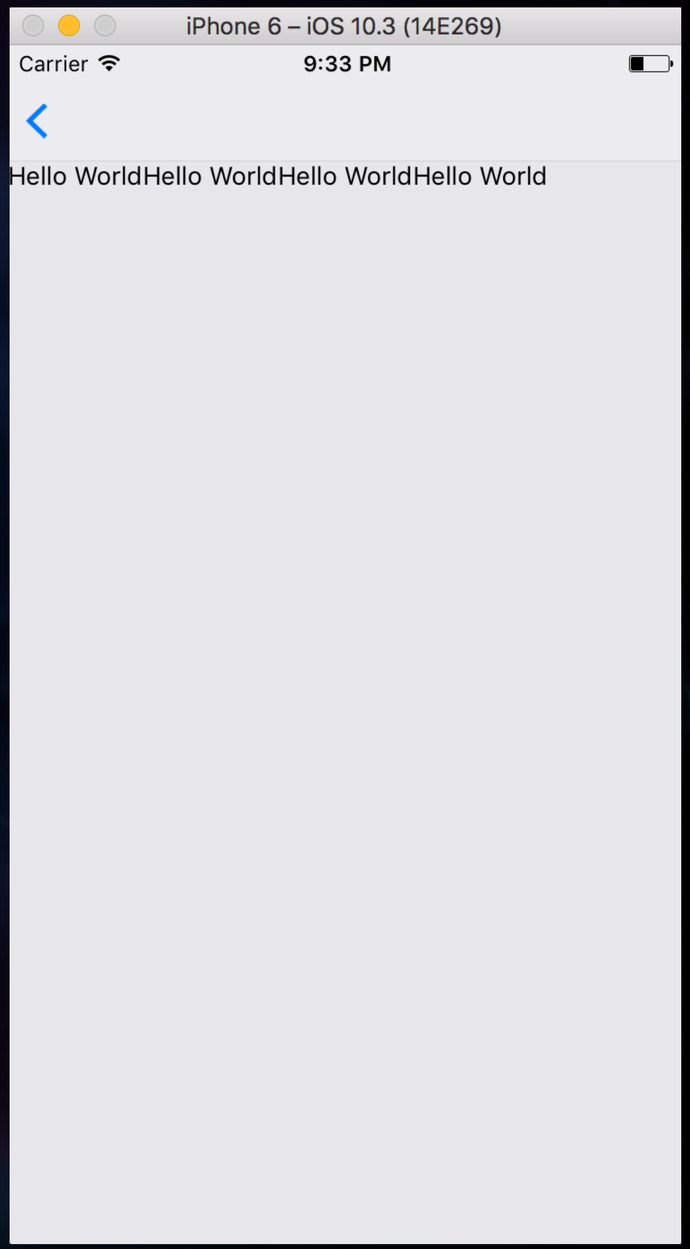
FlexBox

```
container: {
  flex: 1,
}

<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```



FlexBox

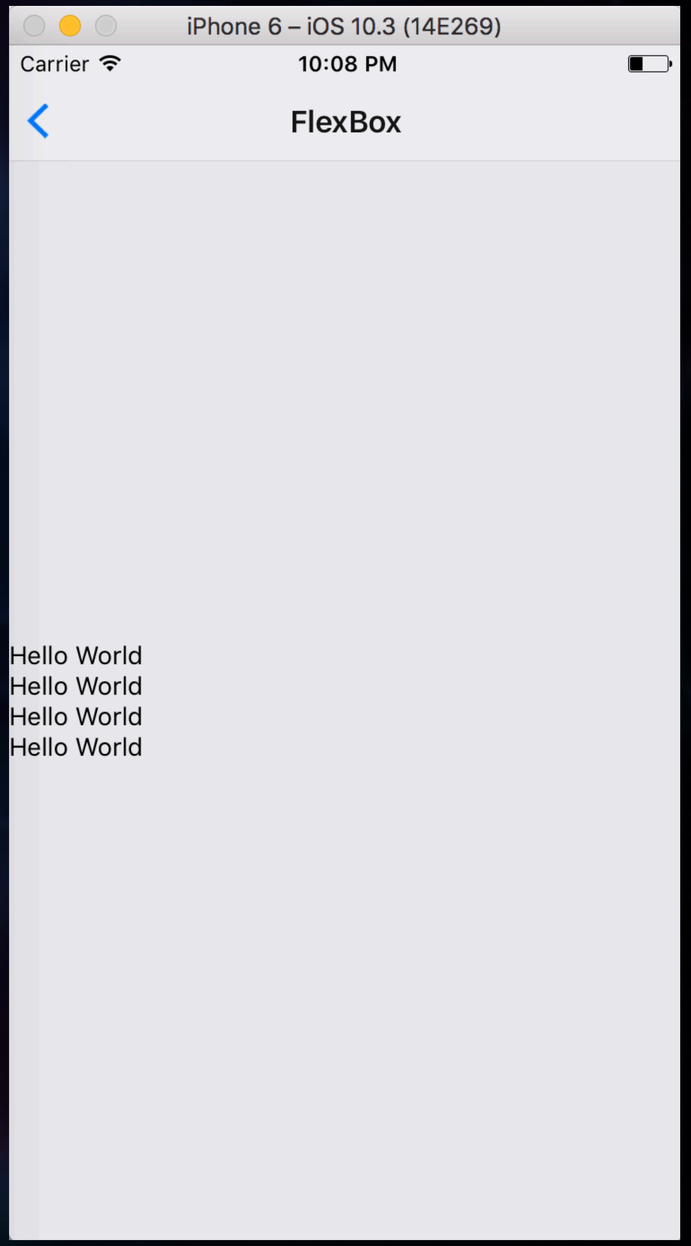
```
container: {
  flex: 1,
  flexDirection: 'row'
},
```



FlexBox

`justifyContent` determines the distribution of children along the main axis.

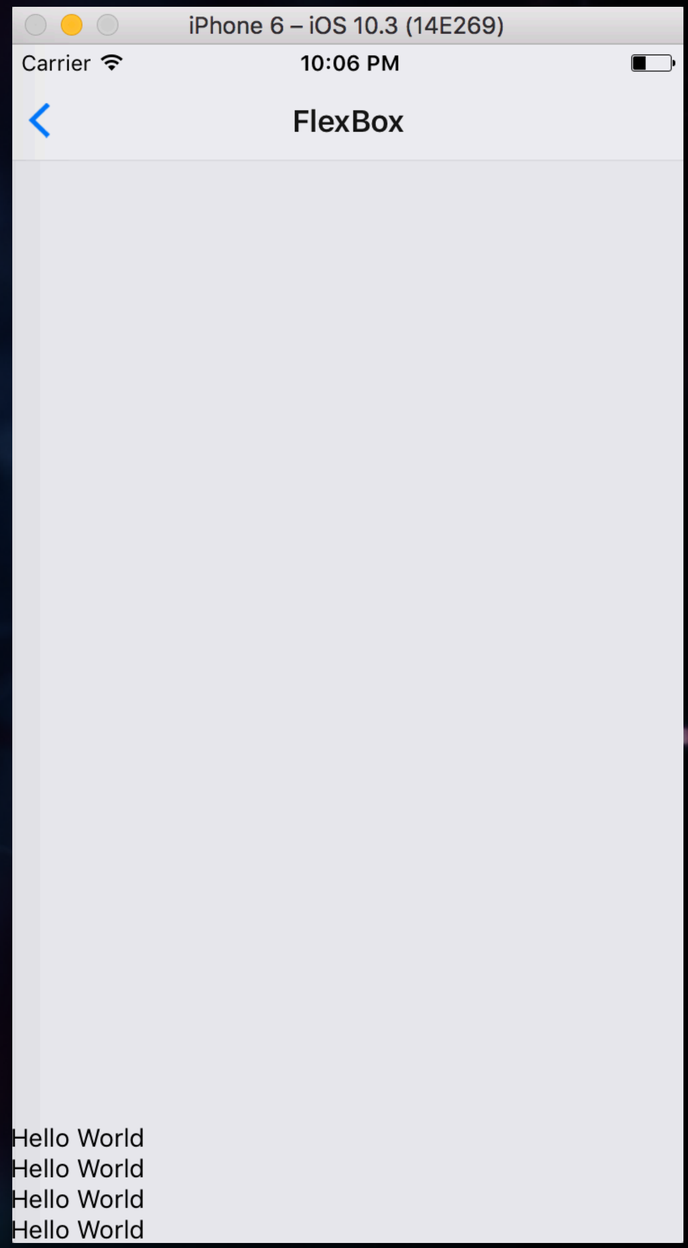
```
container: {
  flex: 1,
  justifyContent: 'center',
},
```



FlexBox

`justifyContent` determines the distribution of children along the main axis.

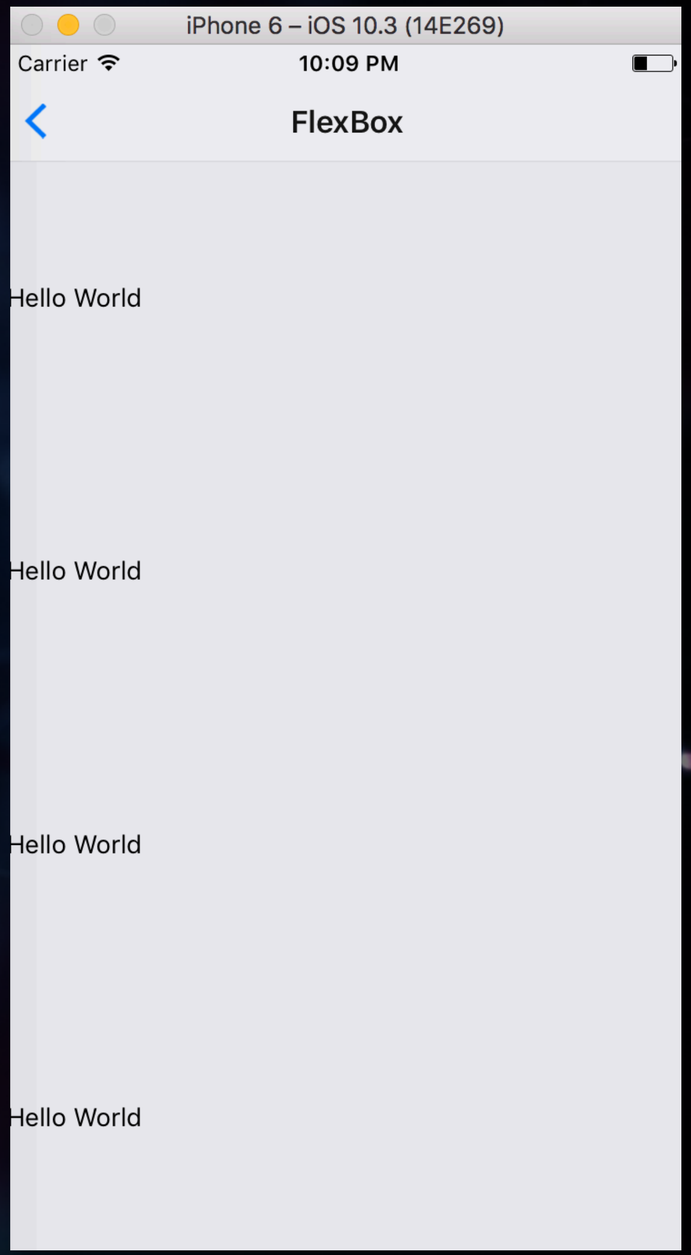
```
container: {
  flex: 1,
  justifyContent: ‘flex-end’,
},
```



FlexBox

`justifyContent` determines the distribution of children along the main axis.

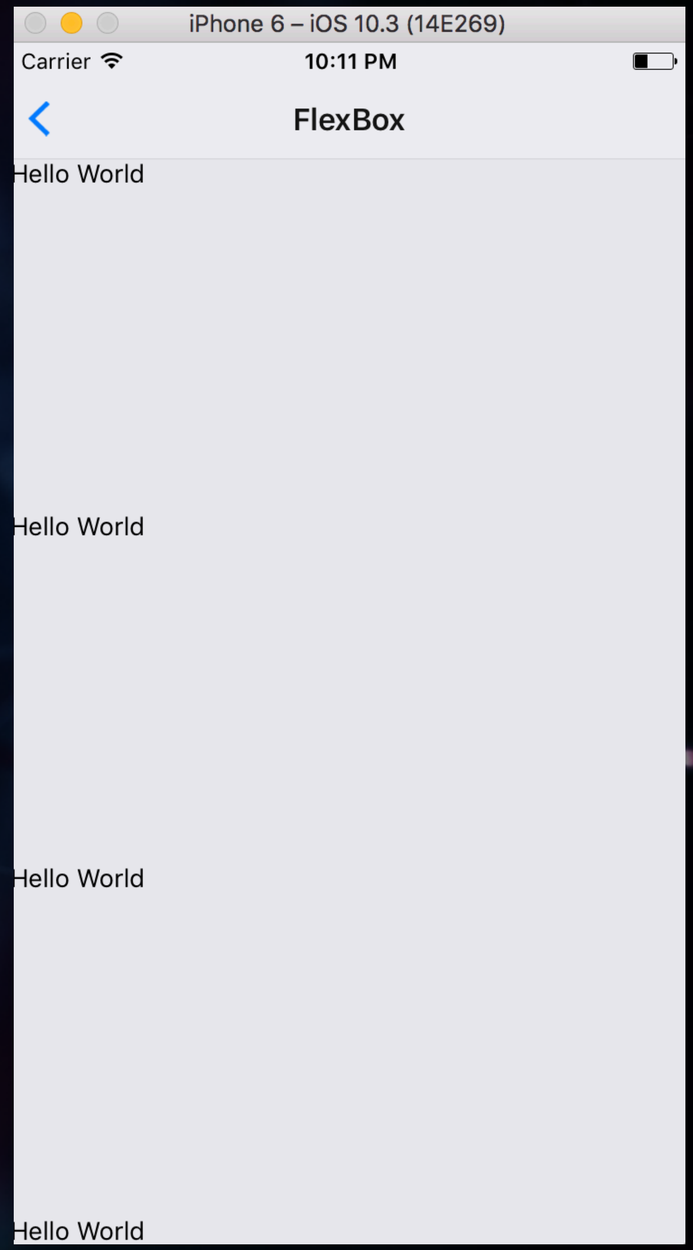
```
container: {
  flex: 1,
  justifyContent: ‘space-around’,
},
```



FlexBox

`justifyContent` determines the distribution of children along the main axis.

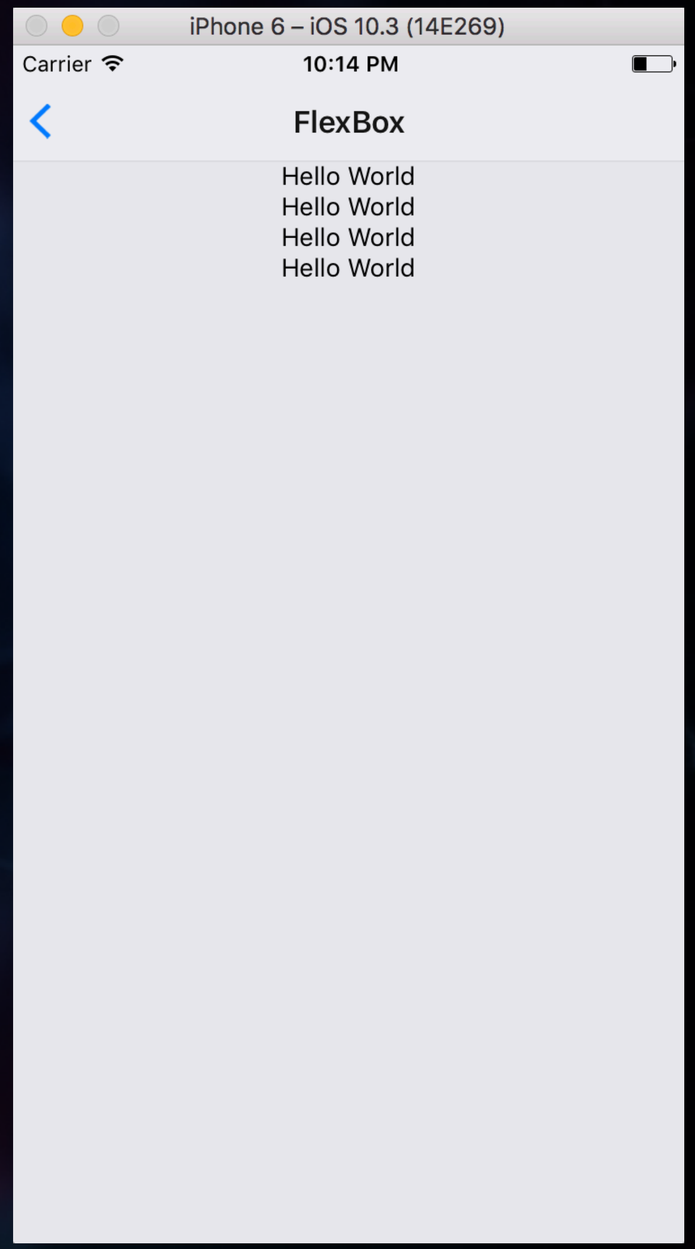
```
container: {
  flex: 1,
  justifyContent: 'space-between',
},
```



FlexBox

`alignItems` determines the distribution of children along the cross axis. Default stretch (`flex-start`, `center`, `flex-end`, and `stretch`)

```
container: {
  flex: 1,
  alignItems: ‘center’,
},
```



Images and SVG Icons in React Native

Images

Static Resources

```
<Image source={require('./my-icon.png')} />
```

iOS

```
<Image source={{uri: 'app_icon'}} style={{width: 40, height: 40}} />
```

Android

```
<Image source={{uri: 'asset:/app_icon.png'}} style={{width: 40, height: 40}} />
```

Network Images

```
<Image source={{uri: 'https://facebook.github.io/react/logo-og.png'}}
   style={{width: 400, height: 400}} />
```

Images

Data Uri

```
<Image
  style={{
    width: 51,
    height: 51,
    resizeMode: Image.resizeMode.contain,
  }}
  source={{
    uri:
      '...',
  }}
/>
```

Background

```
return (
  <ImageBackground source={...}>
    <Text>Inside</Text>
  </ImageBackground>
);
```

Icons

npm install react-native-vector-icons —save

react-native link

```
rnpm-install info Linking react-native-vector-icons ios dependency
rnpm-install info Platform 'ios' module react-native-vector-icons has been successfully linked
rnpm-install info Linking react-native-vector-icons android dependency
rnpm-install info Platform 'android' module react-native-vector-icons has been successfully linked
rnpm-install info Linking assets to ios project
rnpm-install WARN ERRGROUP Group 'Resources' does not exist in your Xcode project. We have created it automatical
rnpm-install info Linking assets to android project
rnpm-install info Assets have been successfully linked to your project
```

Icons

```
import Icon from 'react-native-vector-icons/MaterialIcons';


<Icon
  name="chat"
  color="#fff"
  size={23}
  style={{ padding: 5 }}
/>
```

https://oblador.github.io/react-native-vector-icons/

Lists

```
const people = [
  <Text>Chris</Text>,
  <Text>Amanda</Text>,
  <Text>Jason</Text>,
  <Text>Jennifer</Text>
]

return (
  <View>
    { people }
  </View>
);
```

```
render() {
  const people = ['Chris', 'Amanda', 'Jason', ‘Jennifer'];

  return (
    <ScrollView>
      { people.map(person => <Text>{person}</Text>) }
    </ScrollView>
  );
}
```

Lists

```
renderPerson = (people) => {
  return people.map((person, index) => {
    return (
      <Text key={index}>{person}</Text>
    );
  });
}

render() {
  const people = ['Chris', 'Amanda', 'Jason', 'Jennifer'];
  return (
    <ScrollView>
      { this.renderPerson(people) }
    </ScrollView>
  );
}
```

FlatList

```
state = {
  data: [{ name: 'Chris' }, { name: 'Amanda' }],
}

renderItem = ({ item }) => {
  return <Text>{item.name}</Text>
}

render() {
  return (
    <View>
     <FlatList
          data={this.state.data}
          renderItem={this.renderItem}
          keyExtractor={item => item.name}
          ItemSeparatorComponent={() => <View style={styles.divider} />}
          refreshing={this.state.refreshing}
          onRefresh={this.onRefresh}
        />
    </View>
  );
}
```


Follow the instructions

Building navigation workflow

react-navigation

```
import { createStackNavigator, createAppContainer } from "react-navigation";
import { ChatsScreen, ChatViewScreen } from './src/screens'

const AppNavigator = createStackNavigator({
  chatsScreen: {
    screen: ChatsScreen
  },
  chatView: {
    screen: ChatViewScreen
  }
},{
  initialRouteName: 'chatsScreen',
  navigationOptions: {
    headerStyle: {
      backgroundColor: '#006655',
    },
    headerTintColor: '#fff',
    headerTitleStyle: {
      fontWeight: 'bold',
    },
  }
});

const RootNavigator = createAppContainer(AppNavigator);
```

Navigation options

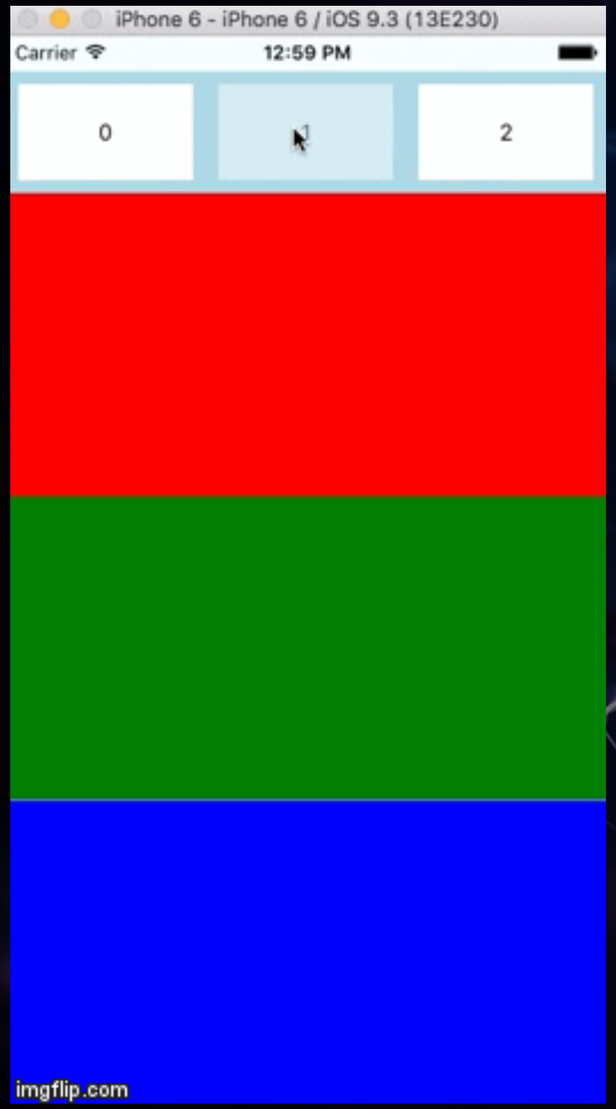
```
static navigationOptions = ({ navigation }) => {
  return {
    title: navigation.state.params.title,
    headerLeft: (
      <Icon name="chevron-left"
        size={40}
        color="#ffffff"
        onPress={() => navigation.goBack() }
      />)
  }
}
render() {
  return (
    <View>
      <Text> ChatViewScreen </Text>
    </View>
  )
}
```

Input and Keyboard

```
<TextInput
  style={styles.composeText}
  value={this.state.text}
  onChangeText={(text) => this.setState({text})}
  onSubmitEditing={this.submit}
  editable = {true}
  maxLength = {40}
/>


keyboardVerticalOffset = Platform.OS === 'ios' ? 60 : 0


<KeyboardAvoidingView
                behavior={Platform.OS === 'ios' ? 'padding' : null}
                keyboardVerticalOffset={this.keyboardVerticalOffset}
                style={styles.container}>
 { /*/ Here goes keyboard aware View with TextInput /*/}
</KeyboardAvoidingView>
```

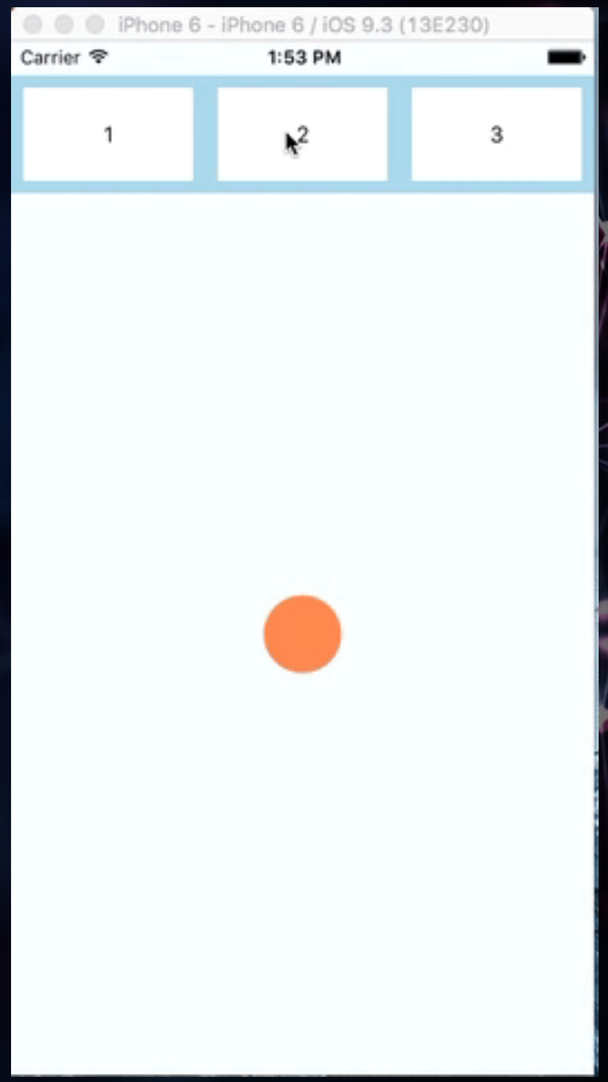

Follow the instructions

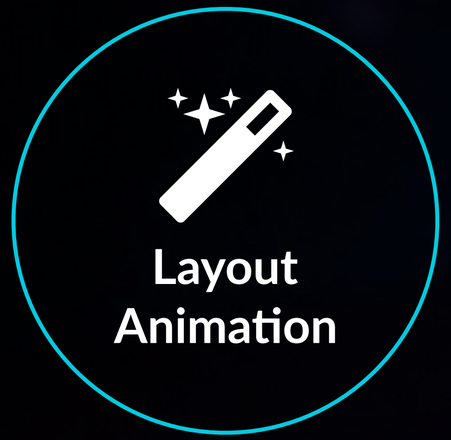
Animations



Layout Animation

Animated



	Pros	Cons
 **Layout Animation**	✔ Applied to all style properties of component ✔ No need for specific values (heights/ widths)	✔ Less configurable ✔ Animates everything on next render
 **Animated**	✔ Extremely configurable ✔ Can target specific components	✔ Uses requestAnimationFrame by default ✔ For complex computation animations can stutter


Animated

- Import Animated

```
import { Animated } from 'react-native';
```

- Create animated value within the class

```
animatedMargin = new Animated.Value(0);
```

- Declare Animated component in render or create your own

```
<Animated.View />

const AnimatedOpacityWrapper = createAnimatableComponent(TouchableOpacity)
```

- Add animated value as style

```
<Animated.View style={{ marginTop: this.animatedMargin }} />
```

- Interpolate on value if needed

```
<AnimatedOpacityWrapper
  style={[styles.chatItem, {
    opacity: this.animatedValue,
    transform: [{
      translateX: this.animatedValue.interpolate({
        inputRange: [0, 1],
        outputRange: [-100, 0]
      })
    }]
  }]}
```

- Trigger animation

```
animate = () => {
  Animated.timing(
    this.animatedMargin,
    {
      toValue: 1,
      duration: 1700,
    }
  ).start()
}
```

Follow the instructions

Connecting our app to API

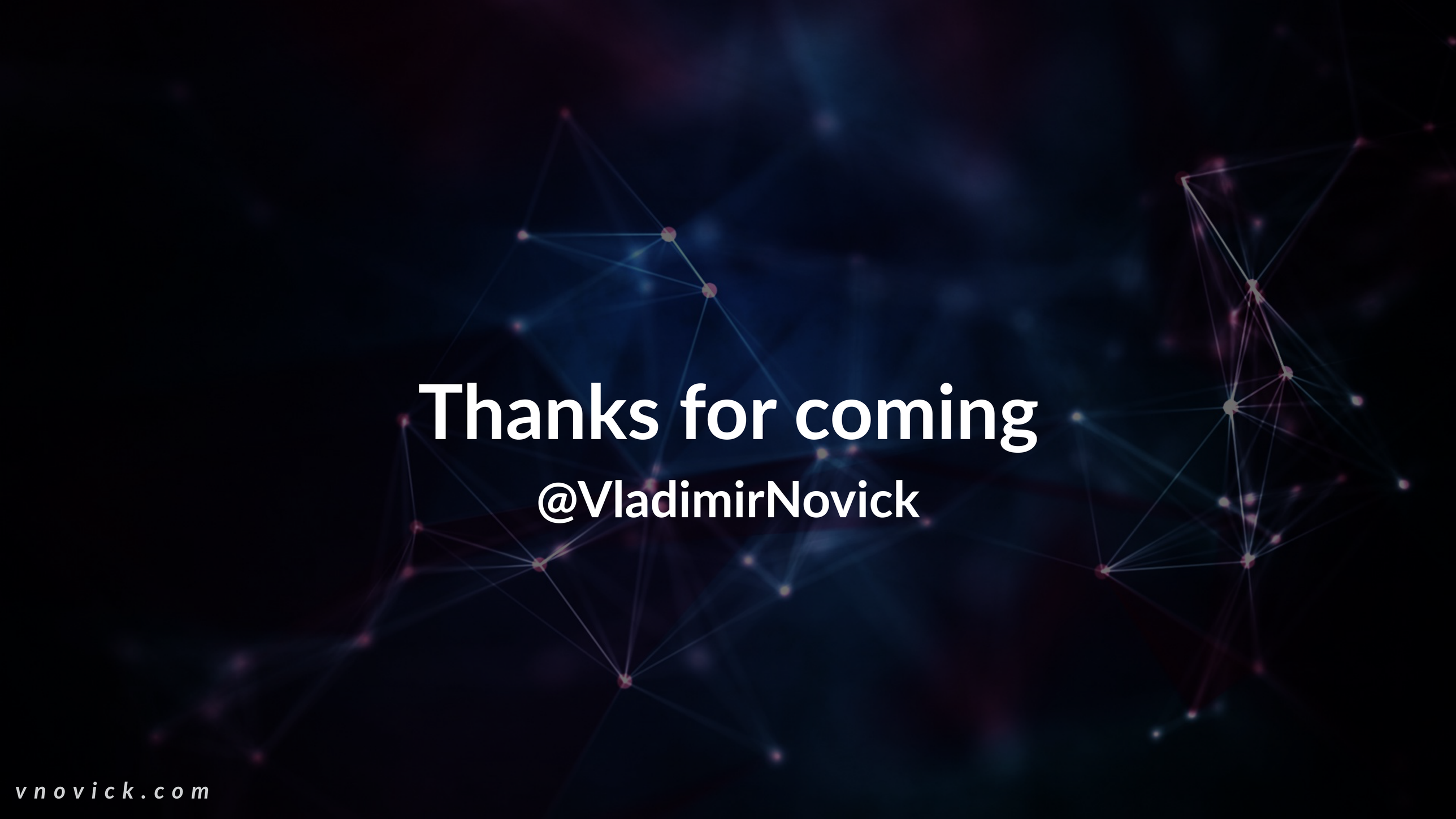
Use any server of your choice

Use fetch with async/await or Promises

Use setState or state management lib

Follow the instructions

Connecting your app to GraphQL


Thanks for coming
@VladimirNovick