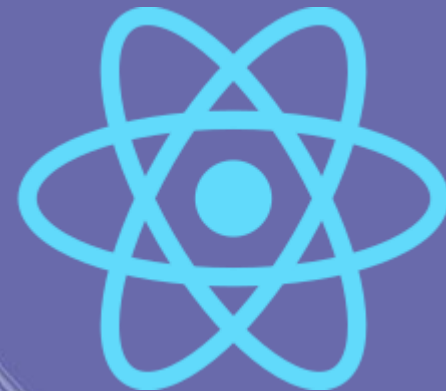
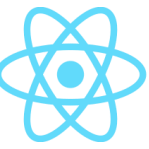


CORE COMPONENT



NGUYEN TRONG TIEN



Core Components

Basic Components

Most apps will end up using one of these basic components.

View

The most fundamental component for building a UI.

Text

A component for displaying text.

Image

A component for displaying images.

TextInput

A component for inputting text into the app via a keyboard.

ScrollView

Provides a scrolling container that can host multiple components and views.

StyleSheet

Provides an abstraction layer similar to CSS stylesheets.



Documents

1. Ebook: React Native Cook book 2019
2. <https://reactnative.dev/>
3. <https://docs.expo.dev/>

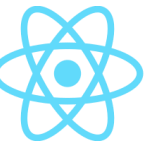




Setting Up Your Environment

We will cover the following topics in this chapter:

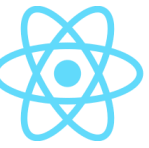
1. Installing dependencies
2. Initializing your first application
3. Running your application in a simulator/emulator
4. Running your application on a real device



Technical requirements

We will cover the following topics in this chapter:

1. Expo
2. Android Studio
3. Node.js

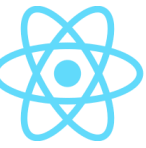


Installing Android Studio

Android Studio comes with the official Android emulator, which is the emulator that Expo recommends for use during development.

1. Download Android Studio from <https://developer.android.com/studio/>.
2. Open the downloaded file and drag the Android Studio.app icon to the Applications folder icon:

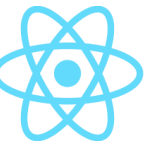




Installing Node.js

- Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine, and is designed to build scalable network applications.
- Node allows JavaScript to be executed in a Terminal, and is an indispensable tool for any web developer.
- Downloading and installing Node.js from the project's site at <https://nodejs.org/en/download/>.





Installing Expo





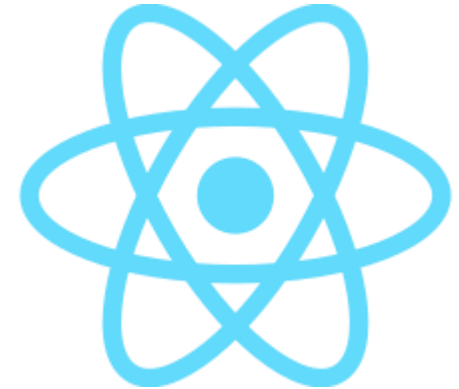
Initializing your first app

- This is all the setup you need in order to get started developing your first React Native app using Expo! There are however a few extra steps you'll need to perform for working with pure React Native apps (non-Expo apps)
- <https://facebook.github.io/react-native/docs/getting-started.html>



Initializing your first app

- We'll create our first app using Expo via the Expo CLI. Making a new application is as simple as running the following:
expo init project-name





Project Structure

~ / **AwesomeProject** /



..



App.js



package-lock.json



assets



app.json



package.json



node_modules



babel.config.js

REACTNATIVE

AwesomeProject

> .expo

> .expo-shared

> assets

> node_modules

.gitignore

JS App.js

{ } app.json

babel.config.js

{ } package-lock.json

{ } package.json

> eventproject

> firstproject

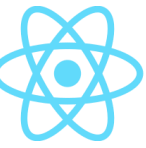
> starting-project

{ } package.json

starting-project.zip

AwesomeProject > JS App.js > ...

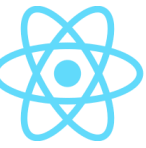
```
1 import { StatusBar } from 'expo-status-bar';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Text>Open up App.js to start working on your app!</Text>
8       <StatusBar style="auto" />
9     </View>
10   );
11 }
12
13 const styles = StyleSheet.create({
14   container: {
15     flex: 1,
16     backgroundColor: 'fff',
17     alignItems: 'center',
18     justifyContent: 'center',
19   },
20 });
```



Running your app on a real device

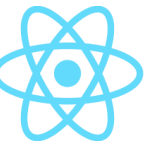
You can get the in-development app running on your phone in three simple steps:

1. Open the App Store on your iPhone, or the Google Play Store on your Android device.
2. Search for and download the Expo Client app.
3. While your app is running on your development machine, you should also have the Expo Developer Tools open in a browser.



Running your app on a real device

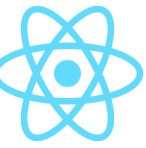
- You should see a QR code at the bottom of the left-hand side menu of the Expo Developer Tools.
- Use the iPhone's native Camera app, or the Scan QR Code button in the Expo Client app on Android, to scan the QR code. This will open your in-development app on the device within the Expo Client app



Creating a Simple React Native App

In this chapter, we'll cover the following recipes:

- Adding styles to elements
- Using images to mimic a video player
- Creating a toggle button
- Displaying a list of items
- Using flexbox to create a layout
- Setting up and using navigation



Style

With React Native, you style your application using JavaScript. All of the core components accept a prop named `style`. 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`.



Style

The style prop can be a plain old JavaScript object. That's what we usually use for example code. You can also pass an array of styles - the last style in the array has precedence, so you can use this to inherit styles.



Style

The style prop can be a plain old JavaScript object. That's what we usually use for example code. You can also pass an array of styles - the last style in the array has precedence, so you can use this to inherit styles.



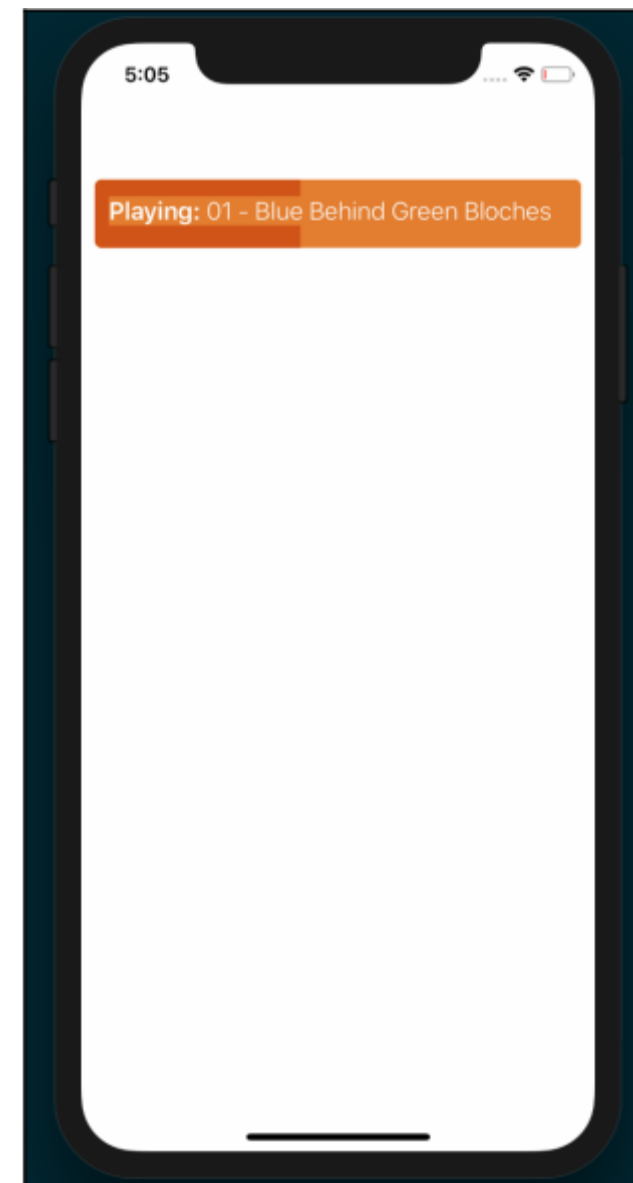
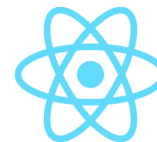
How to do it...

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

export default class App extends React.Component {
  render() {
    const name = '01 - Blue Behind Green Bloches';
    return (
      <View style={styles.container}>
        <View style={styles.innerContainer} />
        <Text style={styles.title}>
          <Text style={styles.subtitle}>Playing:</Text> {name}
        </Text>
      </View>
    );
  }
}
```

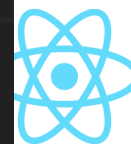
How to do it...

```
const styles = StyleSheet.create({
  container: {
    margin: 10,
    marginTop: 100,
    backgroundColor: '#e67e22',
    borderRadius: 5,
  },
  innerContainer: {
    backgroundColor: '#d35400',
    height: 50,
    width: 150,
    borderTopLeftRadius: 5,
    borderBottomLeftRadius: 5,
  },
  title: {
    fontSize: 18,
    fontWeight: '200',
    color: '#fff',
    position: 'absolute',
    backgroundColor: 'transparent',
    top: 12,
    left: 10,
  },
  subtitle: {
    fontWeight: 'bold',
  },
});
```



Style


```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const LotsOfStyles = () => {
5   return (
6     <View style={styles.container}>
7       <Text style={styles.red}>just red</Text>
8       <Text style={styles.bigBlue}>just bigBlue</Text>
9       <Text style={[styles.bigBlue, styles.red]}>bigBlue, then red</Text>
10      <Text style={[styles.red, styles.bigBlue]}>red, then bigBlue</Text>
11    </View>
12  );
13 };
14
15 const styles = StyleSheet.create({
16   container: {
17     marginTop: 50,
18   },
19   bigBlue: {
20     color: 'blue',
21     fontWeight: 'bold',
22     fontSize: 30,
23   },
24   red: {
25     color: 'red',
26   },
27 });
28
29 export default LotsOfStyles;
```



CORE COMPONENTS –STYLE AND COLOR

reactnative.dev/docs/text

Support Ukraine [UA Help Provide Humanitarian Aid to Ukraine.](#)

 **React Native** 0.69 ▾

Guides **Components** API Architecture Contributing Blog

Core Components ▾

Core Components and APIs

ActivityIndicator

Button

FlatList

Image

ImageBackground

KeyboardAvoidingView

Modal

Pressable NEW

RefreshControl

ScrollView

SectionList

StatusBar

Switch

Text

TextInput

TouchableHighlight

TouchableOpacity

TouchableWithoutFeedback

View

VirtualizedList

Text

A React component for displaying text.

`Text` supports nesting, styling, and touch handling.

In the following example, the nested title and body text will inherit the `fontFamily` from `styles.baseText`, but the title provides its own additional styles. The title and body will stack on top of each other on account of the literal newlines:

Function Component

Class Component

Text Functional Component Example ⓘ ↗

```
import React, { useState } from "react";
import { Text, StyleSheet } from "react-native";

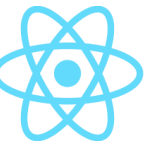
const TextInANest = () => {
  const [titleText, setTitleText] = useState("Bird's Nest");
  const bodyText = "This is not really a bird nest.";

  const onPressTitle = () => {
    setTitleText("Bird's Nest [pressed]");
  };

  return (
    <Text style={styles.baseText}>
      <Text style={styles.titleText} onPress={onPressTitle}>
```

Bird's Nest [pressed]

This is not really a bird nest.



Working with Core Components

```
const MyTitle = props =>{  
  return (  
    <View>  
      <Text> {props.title} </Text>  
    </View>  
  )  
};
```

View

Text

Button

TextInput

Image

.....



STYLING REACT NATIVE APPS

- INLINE STYLES
- STYLE SHEET OBJECTS





INLINE

{{...}}

```
export default function App() {
  return (
    <View style={styles.container}>
      <Text style= {{margin: 20}}>Hello world!</Text>
      <Button title='Click' />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: 'yellow',
    alignItems: 'center',
    justifyContent: 'center'
  }
})
```

- margin? (property)
- marginBottom?
- marginEnd?
- marginHorizontal?
- marginLeft?
- marginRight?
- marginStart?
- marginTop?
- marginVertical?



INLINE

```
export default function App() {  
  return (  
    <View style={styles.container}>  
      <Text style= {{margin: 10, borderWidth:2, borderColor: 'red' }}>Hello world!</Text>  
      <Button title='Click me!'/>  
    </View>  
  );  
}  
  
const styles = StyleSheet.create({  
  container:{  
    flex:1,  
    backgroundColor:'yellow',  
    alignItems:'center',  
    justifyContent:'center'  
  }  
})
```

Hello world!

CLICK ME!



STYLESHEET

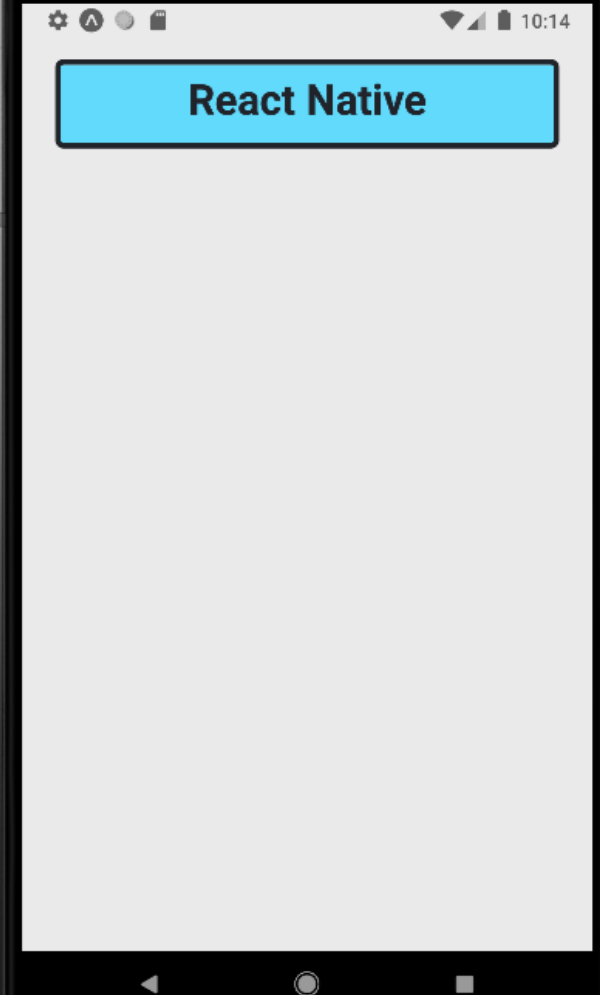
container

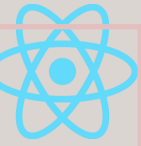
```
export default function App() {  
  return (  
    <View style={styles.container}>  
      <Text style={{margin: 20}}>Hello world!</Text>  
      <Button title='Click me!' />  
    </View>  
  );  
}  
  
const styles = StyleSheet.create({  
  container:{  
    flex:1,  
    backgroundColor:'yellow',  
    alignItems:'center',  
    justifyContent:'center'  
  }  
});
```



STYLESHEET

```
const App = () => (  
  <View style={styles.container}>  
    <Text style={styles.title}>React Native</Text>  
  </View>  
>;  
  
const styles = StyleSheet.create({  
  container: {  
    flex: 1,  
    padding: 24,  
    backgroundColor: "#eaeaea"  
  },  
  title: {  
    marginTop: 16,  
    paddingVertical: 8,  
    borderWidth: 4,  
    borderColor: "#2032a",  
    borderRadius: 6,  
    backgroundColor: "#61dafb",  
    color: "#2032a",  
    textAlign: "center",  
    fontSize: 30,  
    fontWeight: "bold"  
  }  
});
```





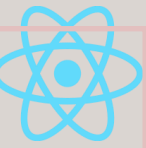
Core Components, Styling & Colors - More Information

- Official styling

documentation: <https://reactnative.dev/docs/styling>

- Official article about "Colors" to learn about different ways of setting & using colors in React Native apps: <https://reactnative.dev/docs/colors>





Core Components, Styling & Colors - More Information

official API reference articles for the
different core components

<https://reactnative.dev/docs/view>

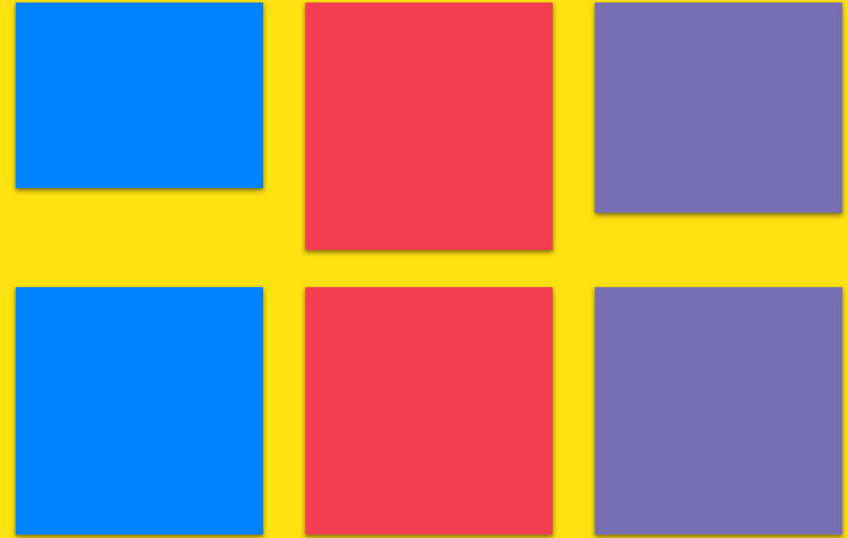
- <https://reactnative.dev/docs/view#style>): <https://reactnative.dev/docs/view-style-props>

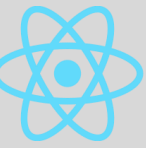


EXPLORING LAYOUTS & FLEXBOX

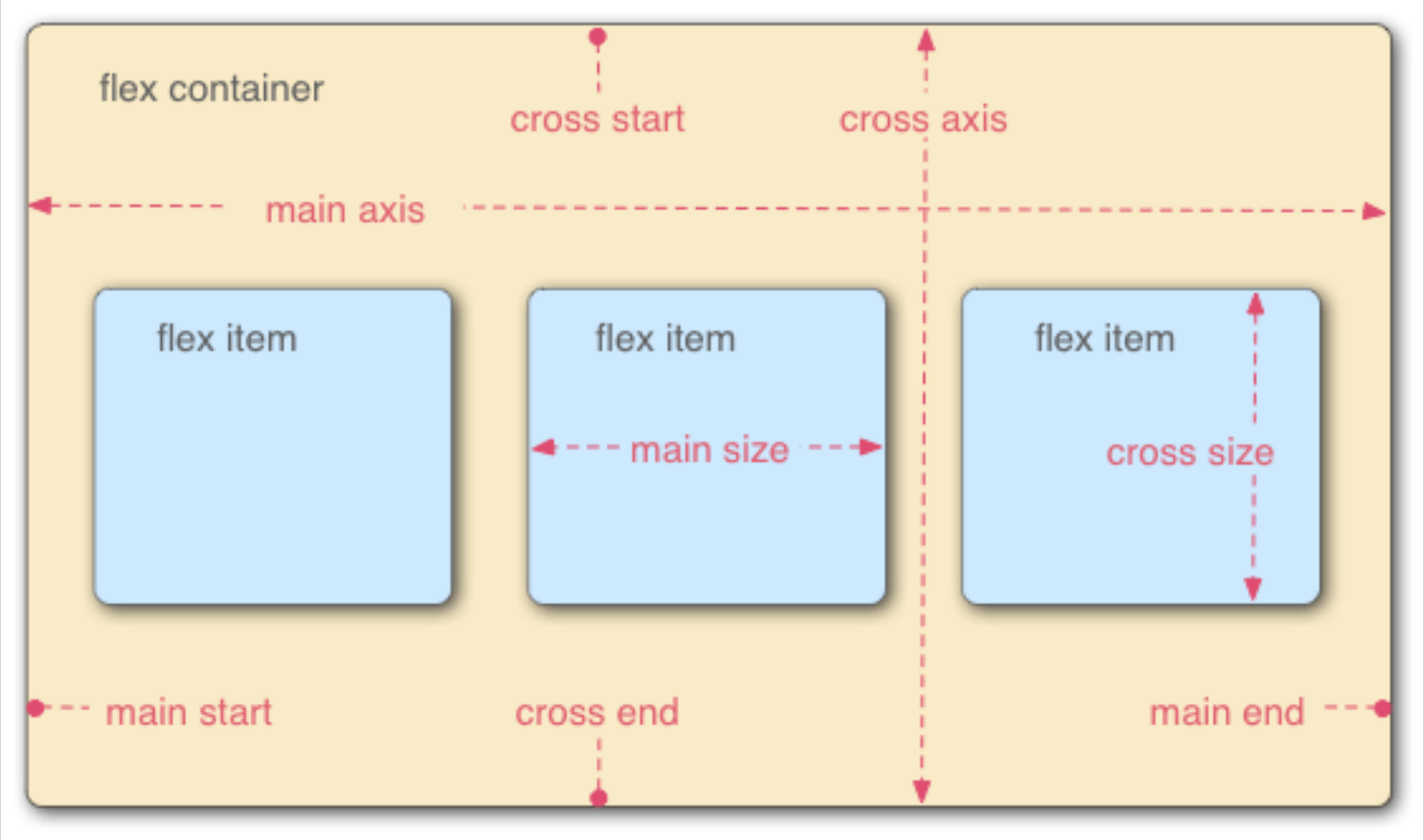
- Layouts are created with Flexbox
- Very similar to browser CSS Flexbox
- Elements are positioned inside of containers

Flexbox



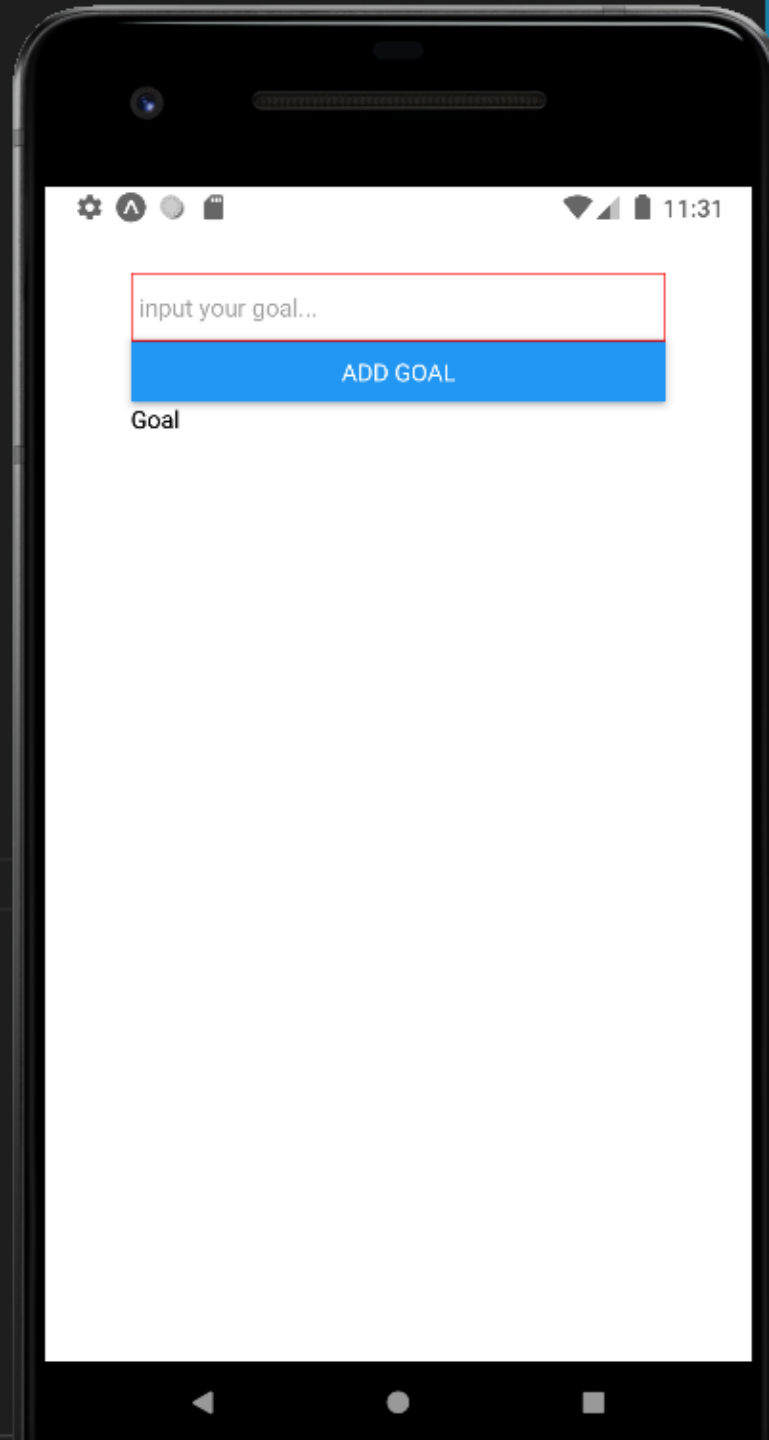


Flexbox



Using Flexbox to create Layout

```
const App = () => (  
  <View style={styles.container}>  
    <TextInput style={styles.textinput}  
      placeholder='input your goal...'  
    </TextInput>  
    <Button  
      style={styles.button}  
      title='Add Goal'>  
    </Button>  
    <Text>Goal</Text>  
  </View>  
>);  
  
const styles = StyleSheet.create({  
  container: {  
    padding: 50  
  },  
  textinput: {  
    borderWidth: 1,  
    borderColor: 'red', padding: 5  
  }  
});
```



flexDirection: 'row'

```
const App = () => (  
  <View style={styles.container}>  
    <TextInput style={styles.textinput}  
      placeholder='input your goal...'  
    </TextInput>  
    <Button  
      style={styles.button}  
      title='Add Goal'>  
    </Button>  
    <Text>Goal</Text>  
  </View>  
>;  
  
const styles = StyleSheet.create({  
  container: {  
    padding: 50,  
    ✨ flexDirection: 'row'  
  },  
  textinput: {  
    borderWidth: 1,  
    borderColor: 'red', padding: 5  
  }  
});
```

