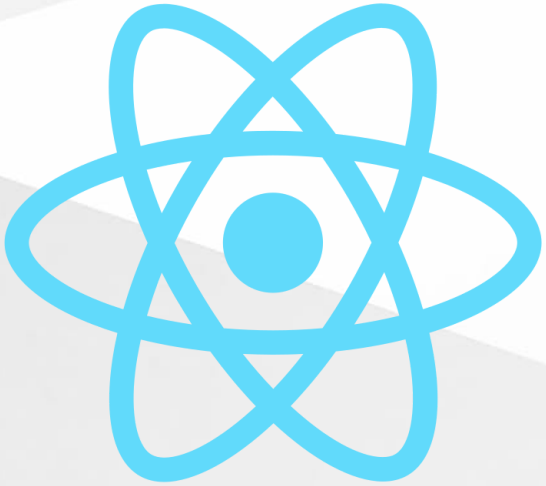


React Native Introduction

First steps into React Native development

Before we start

I hope, all the people interested in doing the exercises, have followed the steps of the `Install React Native dependencies` guide.

If you want to do the exercises but don't have the dependencies you can try to install `Expo` to follow the exercises.

Install guide

Expo guide

Before we start

Please, clone the repo with the exercises

```
TODO: Repo url here
```

Also, make npm install in the `MyExampleApp` directory

```
cd MyExampleApp && npm install
```

Objetives of the workshop

- A brief introduction to React Native.
- How works React Native.
- How to create a project
- Make your first little app

Introduction: kinds of mobile apps

Native

platform specific
good performance
a project per platform

Web apps

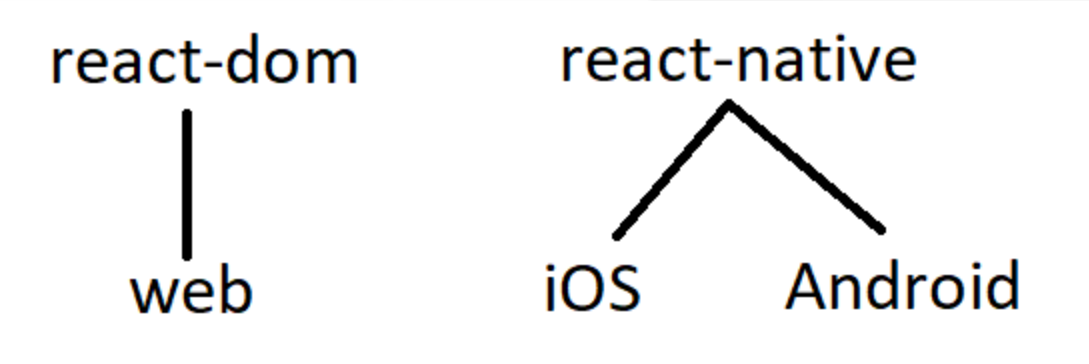
HTML apps adapted to mobile
PWA

Hybrid

App made in web technologies but wrapped in a native app

What is React Native

A framework for building native apps using React. In short, a kind of react renderer which targets mobile apps, iOS and Android.



There are also other renderes:

```
react-canvas react-blessed react-vr react-pdf react-fs-renderer
```

How is a React Native component?

In `react-dom` we use the building blocks from the web (div,span,table) and `react-native` use his own building blocks.

```
const MyComponent = () => {
  return (
    <div>
      <span>Hello world!</span>
      <button>click here</button>
    </div>
  )
}
```

```
3  const MyComponent = () => {
4    return (
5      <View>
6        <Text>Hello world!</Text>
7        <Button>click here</Button>
8      </View>
9    )
10 }
```

Lots of building blocks

SafeAreaView
StatusBar
ScrollView
Switch
TextInput
RefreshControl

Text
Button
View
Image
ImageBackground
Pressable

Every core component have traduction to a native component

Lots of apis

react-native-sensors (accelerometer, gyroscope, magnetometer, barometer)

react-native-device-info (Id, OS, manufacturer, battery, etc...)

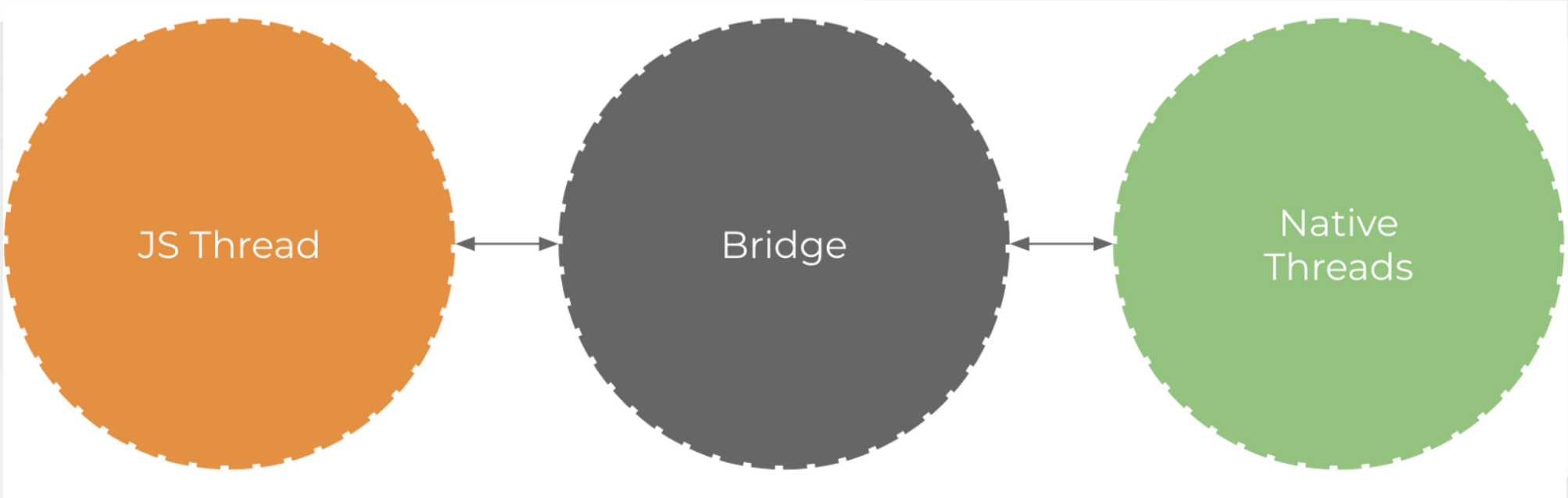
react-native-maps

react-native-camera

> If there is no package, you can always develop a native module

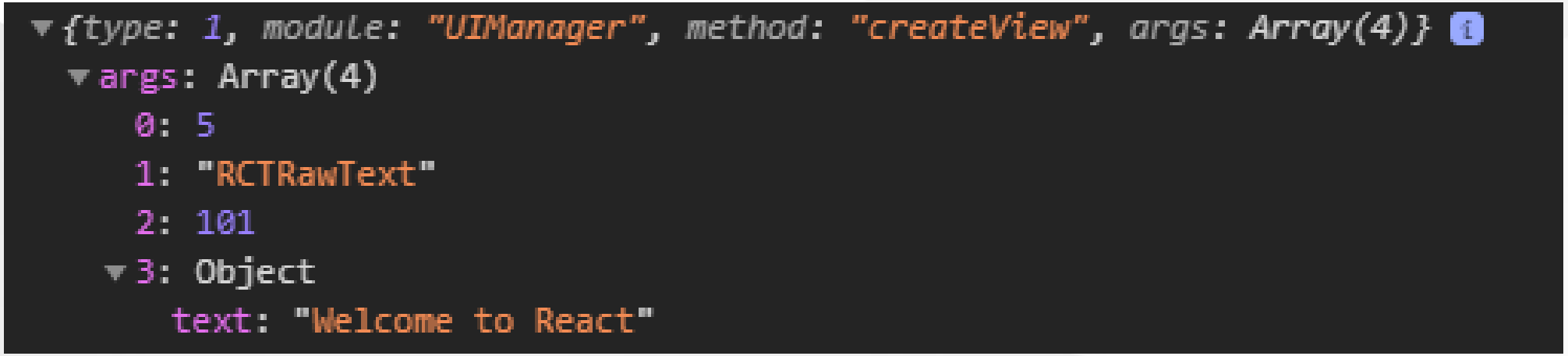
How works React Native?

Asynchronous bidirectional communication between Javascript and Native side via `bridge`.



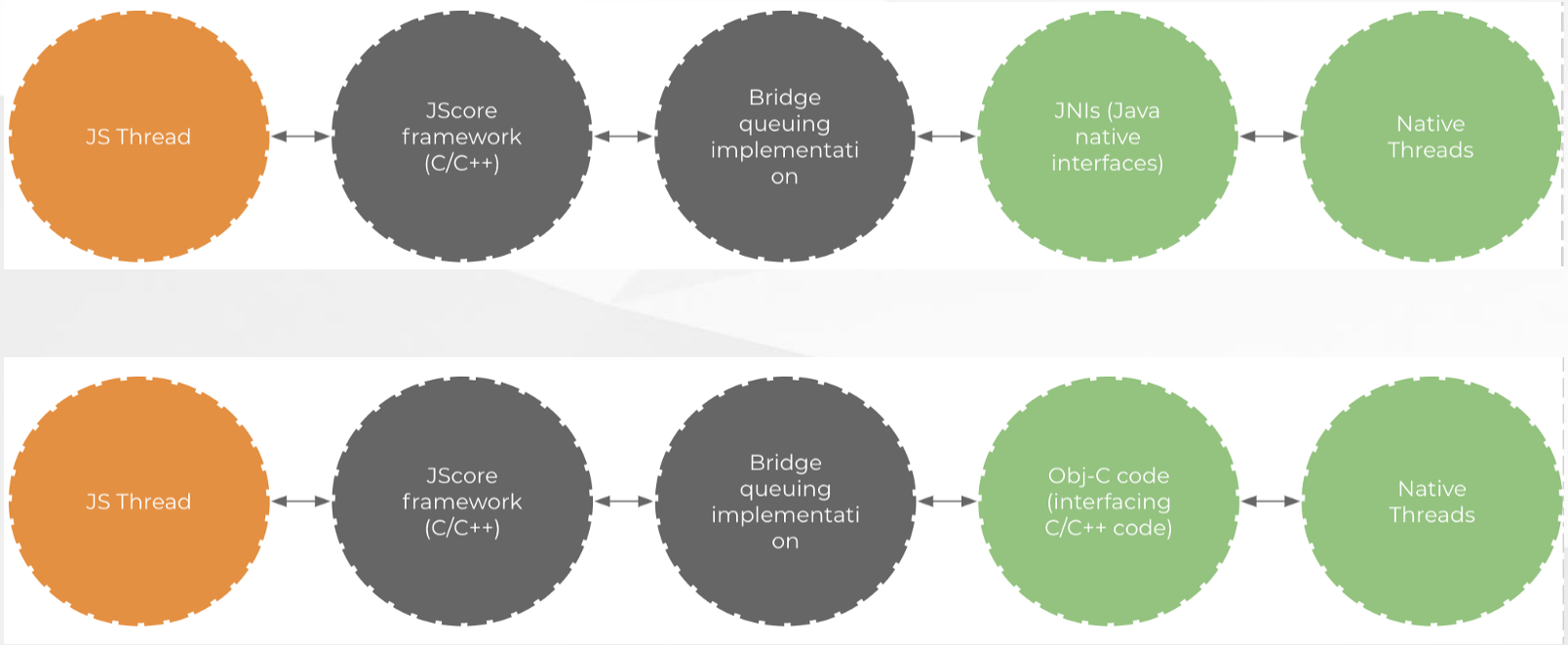
Bridge in a nutshell

In short, the bridge component is a message broker which interprets messages sent in json.



Bridge implementations

Every platform has its own implementation of the bridge on the native side.



Other implementations

React Native can target other platforms because the agnostics nature of the bridge. For example we have:

React Native for Windows + macOS

React Native web 🤯

This is very interesting.... but where is the code?

How to start a project

ES6 version

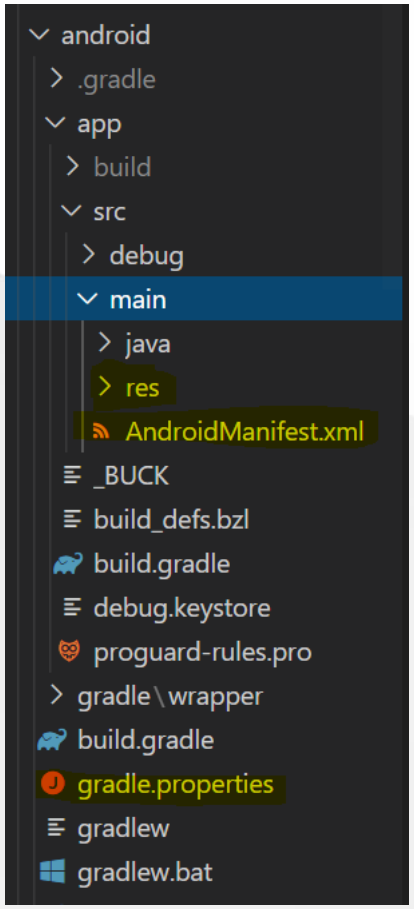
```
npx react-native init AwesomeProject
```

Typescript version

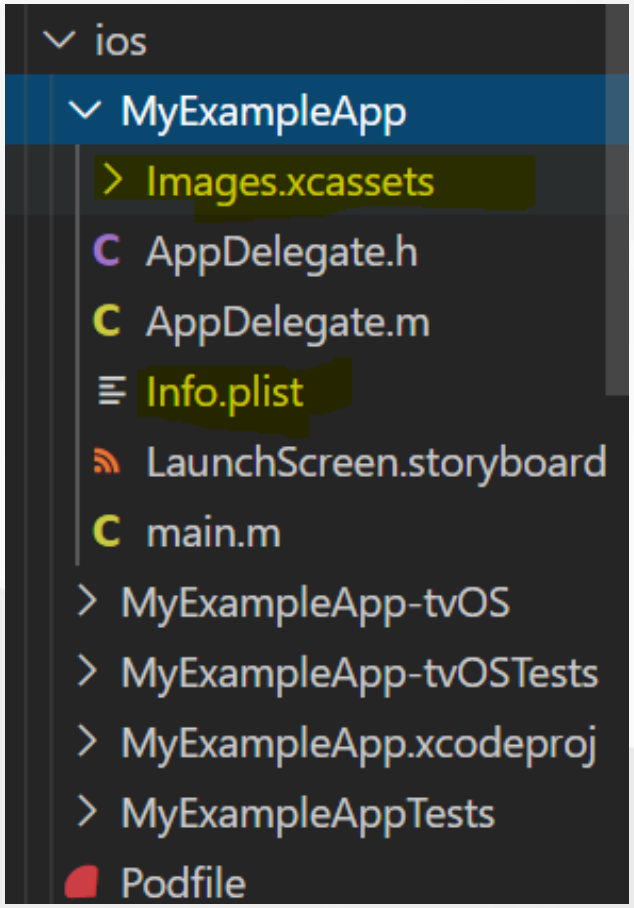
```
npx react-native init AwesomeProject --template react-native-template-typescript
```

Project anatomy

Android



iOS



Let's start the workshop

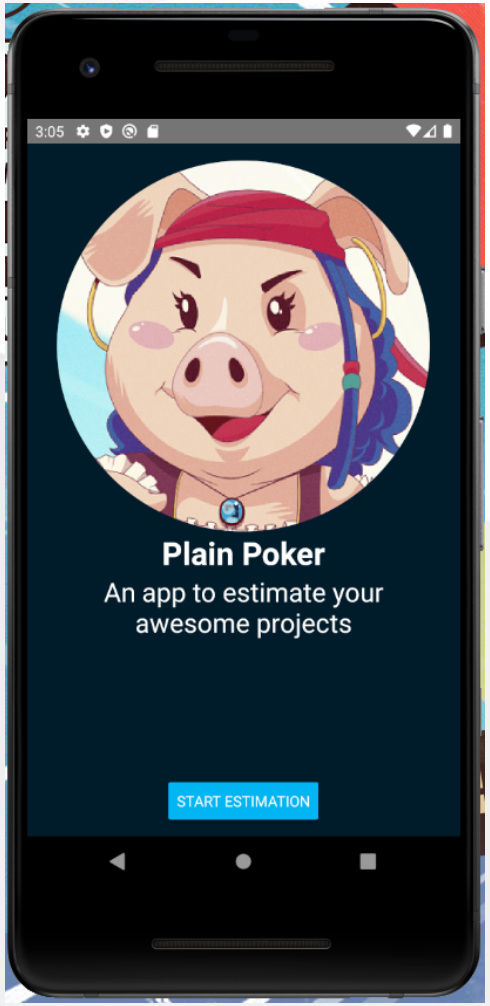
Does everybody have his environment ready?

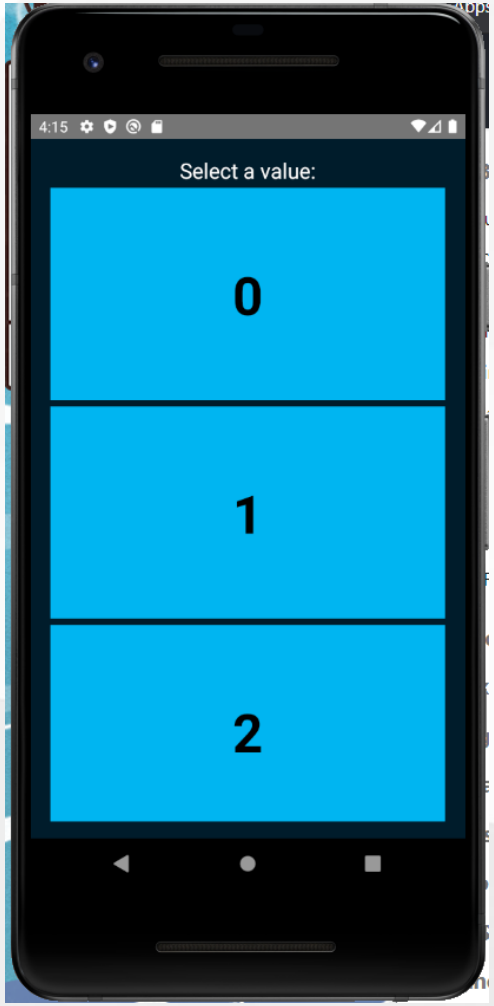
Ready Steady Open Emulators!

Start project!

```
npx react-native run-android
```

Plain poker: an estimation tool





Example 1: Building blocks

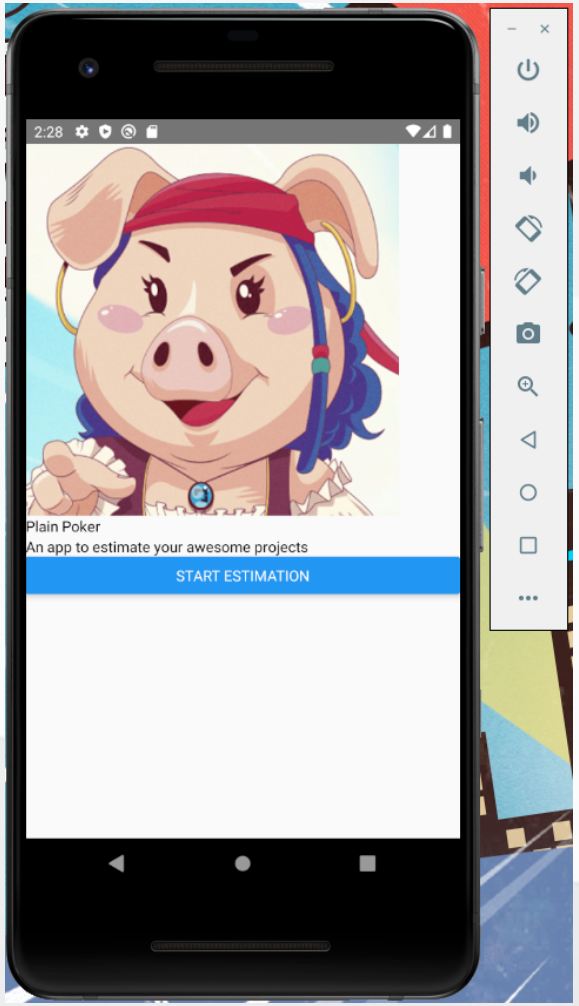
```
export const Home: React.FC = () => {
  return (
    <SafeAreaView>

      <Image source={require('../assets/piggy.png')}/>

      <Text>Plain Poker</Text>

      <Text>An app to estimate your awesome projects</Text>

      <Button
        onPress={() => { }}
        title="Start estimation" />
    </SafeAreaView>
  )
};
```



Example 2: Styling components

```
const style1 = StyleSheet.create({
  myStyle: {
    backgroundColor: 'red',
    fontWeight: 'bold'
  }
});

const style2 = StyleSheet.create({
  myOtherStyle: {
    ...style1.myStyle,
    fontSize: 25
  }
})
```

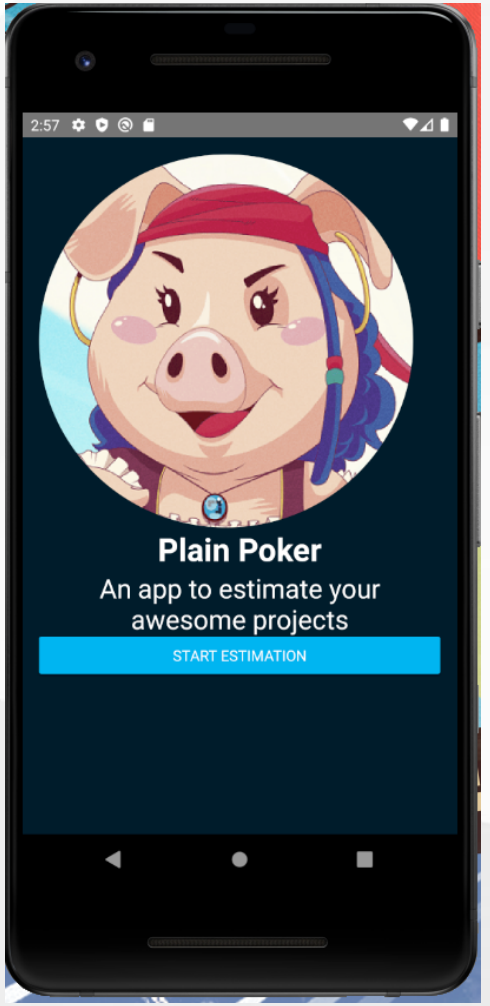
```
<Text style={[style2.myOtherStyle]}>Plain Poker</Text>
```

Example 2: Styling components

This is our objective. Don't worry about the layout distribution. Feel fre to make other styles 😉

Buttons are special components and don't have style prop!

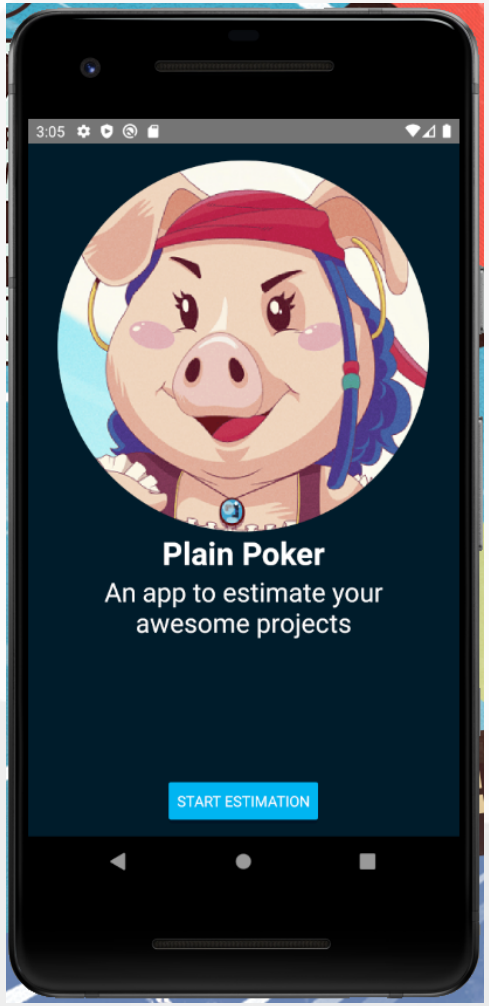
```
<Button
  color={'#00b5f1'}
```



Example 3: App layout

React Native lives in the flexbox realm. This is the display for all containers.

```
justifyContent: 'space-around',
alignItems: 'center',
flex: 1,
alignContent: 'flex-end',
```



Example 4: Navigation

In web development, we have the `history navigator` where we have a history of visited pages.

In mobile development, navigation follows other patterns. In our example, we are going to use a stack navigator. But there are other navigators like `drawer`.

Example 4: Navigation

Create a new stack navigator in the App component and also a new Page component.

```
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { ValuesList } from './src/ValuesList';

const Stack = createStackNavigator();

const App = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator headerMode={'none'}>
        <Stack.Screen name="Home" component={Home} />
        <Stack.Screen name="Values" component={ValuesList} />
      </Stack.Navigator>
    </NavigationContainer>
  );
};
```

```
const navigation = useNavigation();
return (
  <SafeAreaView style={HomeStyles.background}>

    <Image
      style={HomeStyles.logo}
      source={require('../assets/piggy.png')} />

    <Text style={HomeStyles.title}>Plain Poker</Text>

    <Text style={HomeStyles.subTitle}>An app to estimate your awes

    <Button
      color={'#00b5f1'}
      onPress={() => { navigation.navigate('Values'); }}
      title="Start estimation" />
  </SafeAreaView>
)
;
```

Example 5: Lists

React Native has also components to make a list of items with scroll. For example, `FlatList` is a good element to make a long list of data because is optimized to only renders elements that are currently showing on the screen.

```
const values = [
  { key: '0' },
  { key: '1' },
  { key: '2' },
  { key: '3' },
  { key: '5' },
  { key: '8' },
  { key: '13' },
];

export const ValuesList: React.FC = () => {
  const [selectedValue, setSelectedValue] = useState(undefined);

  return (
    <SafeAreaView style={ValuesStyles.background}>
      <Text style={ValuesStyles.title}>Select a value:</Text>

      {selectedValue === undefined && <FlatList
        data={values}
        renderItem={({ item }) => (
          <TouchableOpacity
            onPress={() => setSelectedValue(item.key)}
            activeOpacity={.5}
            style={ValuesStyles.listItem}
            key={item.key}>
            <Text style={ValuesStyles.listItemText}>{item.key}</Text
          </TouchableOpacity>
        )}
      />}
```

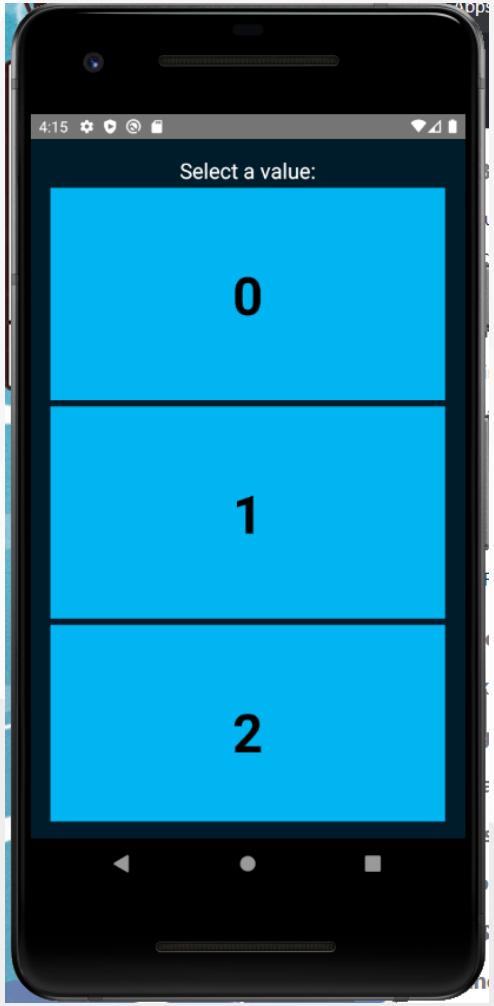
Example 5: Lists

```
const values = [
  { key: '0' },
  { key: '1' },
  { key: '2' },
  { key: '3' },
  { key: '5' },
  { key: '8' },
  { key: '13' },
];

export const ValuesList: React.FC = () => {
  const [selectedValue, setSelectedValue] = useState(undefined);

  return (
    <SafeAreaView style={ValuesStyles.background}>
      <Text style={ValuesStyles.title}>Select a value:</Text>

      {selectedValue === undefined && <FlatList
        data={values}
        renderItem={({ item }) => (
          <TouchableOpacity
            onPress={() => setSelectedValue(item.key)}
            activeOpacity={.5}
            style={ValuesStyles.listItem}
            key={item.key}>
            <Text style={ValuesStyles.listItemText}>{item.key}</Text
          </TouchableOpacity>
        )}
      />}
```



Example 6: Finishing the app

In the last exercise, everyone is going to finish the app on their own.

The objective is to press over a value a show it to your partners in the estimation.

Later is will show my final solution (very simple)

Bonus track: Building your app

To test the app like a real application we have to build the final bundle and install it on a real device.

In the repository, there is a Github Action pipeline with the steps to build our app and publish it to AppCenter.

Later, with the AppCenter app, we can install the application on a real device and tests it.

Let's check the process!

Thanks!!