Layout Components

tl;dr

- Containers are the main thing we use to control the layout of the screen
- Most were designed to be nested inside one another, each adding their own capabilities
- View is the most fundamental container
- ScrollView allows us to scroll through content that won't fit on a screen
- Modal is a container that can't be ignored
- StatusBar only lets us change the appearance of the status bar

Categories of RN components

Category	Some components
Layout	Modal, View, SafeAreaView, ScrollView, RefreshControl, KeyboardAvoidingView, StatusBar, WebView
Single-value	Text, TextInput, Slider, Switch, Image
List	Picker, FlatList, SectionList
Touchable	Button, TouchableHighlight, TouchableNativeFeedback, TouchableOpacity, TouchableWithoutFeedback
Others	ActivityIndicator, Platform-specific components

We mentioned containers before but now it's time to dig in to ...

- View
- SafeAreaView
- ScrollView
- KeyboardAvoidingView
- Modal
- StatusBar

View

<View>

- The most fundamental, most generic of all components.
- Like a in html, this is a container.

The most used control of them all?

- View is the fundamental unit. Like a for native devices
- Allows layouts by supporting flexbox
- Can have as many children as you like, including more views.

SafeAreaView

SafeAreaView doesn't render in non-viewable areas

- Works cross-platform but doesn't do anything on Android
- Looks great on iPhone, though!

It wraps your regular view

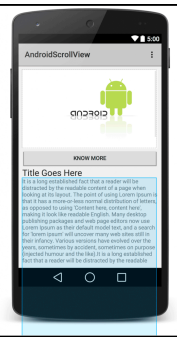
MyComponent.js

```
function MyComponent() {
 return (
  <SafeAreaView>
   <View>
    <Text>I'm the regular view</Text>
   </View>
  </SafeAreaView>
}
```

<ScrollView>

- Without any props, a <ScrollView> makes its contents scrollable immediately.
- Great for when you have too much content in a single view and you want the user to be able to scroll the page to see it all.

```
<ScrollView>
   <TonsMoreContentHere />
</ScrollView>
```



Makes other views scrollable

- Can set up scrolling horizontally as well as vertically
- Can also set up paging by swiping left/right.
- Nests inside of other Views to make their content scrollable within them.
- All elements inside a ScrollView are rendered even if they're off screen

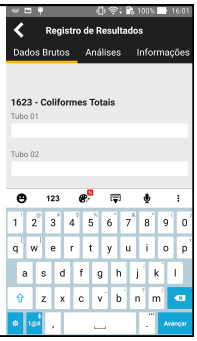
To pinch-to-zoom

- The ScrollView has to have a single item.
- Set minimumZoomScale and maximimZoomScale to enable. it.

KeyboardAvoidingView

- When the user focuses on a <TextInput>, the keyboard slides up.
- This sometimes covers the input they're trying to type into.

KeyboardAvoidingView will move the rest of the view out from behind the keyboard



Wrap the rest of the view in a <KeyboardAvoidingView>

webpack.config.js

```
function MyComponent() {
 return (
  <KeyboardAvoidingView behavior="padding">
   <View>
    <Text>All the other view goes here</Text>
   </View>
  </KeyboardAvoidingView>
)}
```

Some props

- enabled - boolean. Assumed to be true.
- behavior - height | position | padding
  - height - Force the height of the rest of the view to be the remaining height of the screen
  - position - Move the whole thing up
  - padding - Squeeze everything on the page as much as possible by decreasing the padding around elements

From the RN docs: "Note: Android and iOS both interact with behavior differently. Android may behave better when given no behavior prop at all, whereas iOS is the opposite.

<Modal>

<Modal>

- Creates a modal window.
- A pop-up that can't be ignored
- On top of the other Window
- Wraps other views

Modal props

- visible - a bool. Only a true boolean false will hide it. Falsey values cause the Modal to show -- even undefined.
- onShow
- onDismiss
- onOrientationChange

```
class MyComponent extends Component {
 state = {shown: false};
 render() {
  return <View>
   <Modal visible={this.state.shown}>
    <View>
     <Text>I'm the modal</Text>
     <Button title="Dismiss"
      onPress={()=>this.setState({shown:false})}/>
    </View>
   </Modal>
   <Button title="Show Modal"
    onPress={() => this.setState({shown:true})} />
  </View>
 }
}
```

Toggle its *visible* property to show/hide

StatusBar

- Your app is not in control of the status bar
- The status bar doesn't honor styles like everything else because it is controlled by the OS and not the app
- All this control can do is change a few styles on the bar

StatusBar

StatusBar props

- barStyle - An enum - 'dark-content', 'light-content', 'default'
- backgroundColor - Yep, just what the name says.
- translucent - bool. When true, the app draws behind it. When false, the app is placed below the statusBar.

tl;dr

- Containers are the main thing we use to control the layout of the screen
- Most were designed to be nested inside one another, each adding their own capabilities
- View is the most fundamental container
- ScrollView allows us to scroll through content that won't fit on a screen
- Modal is a container that can't be ignored
- StatusBar only lets us change the appearance of the status bar