Layout Components Lab

ScrollView

By now you've surely seen that not all your movies fit on the landing scene. We're going to fix that first by using a ScrollView.

1. Open Landing.js and find where you're mapping through the films. Wrap them with a <ScrollView>

2. Run and test. You can scroll now! Well, that was easy, wasn't it?

3. Edit FilmDetails, Checkout, and Ticket, putting in a <ScrollView> so all of it can be seen.

Using a SafeAreaView

Now you might notice that some of the content is way too high on the screen, especially on an iPhone. And parts may be hidden behind a camera notch. Let's move it all below the status bar.

4. Add this to the top of your Landing.js.

```
import { SafeAreaView } from 'react-native-safe-area-context';
```

5. Add a <SafeAreaView> that wraps your <ScrollVew>.

6. Run and test. This should look great.

Adjusting the StatusBar

Notice that the status bar is covering the top of our app. Let's see what it looks like if we hide the status bar.

7. Open App.js. Notice the <StatusBar> control. First change it's backgroundColor property to 'red' or something. If you're on iOS, you won't see any difference because the status bar is always transparent on iOS. But on Android, you'll see the color change.

8. Then set the hidden property to true.

9. Run and test. The status bar should now be hidden. Note that you can still get to the status bar if you pull down from the top.

Use a KeyboardHidingView

After the user has selected their movie and their seats we'd like for them to actually pay us. That'd be nice, wouldn't it? So let's work with the Checkout component.

10. Run and test. Try to enter some text in each <TextInput>. Depending on the emulator you're using, the soft keyboard may slide up. If not, force it to come up through the emulator settings.
    Hint: Cmd K for it iOS simulator and Ctrl F11 on the Android emulator.

You probably don't notice a problem because our form only takes up half the screen but when the cart is added to the screen someday, we'll have an issue. Let's show you what we mean.

11. Add a new <Text> below the "Your cart" Text. Put a bunch of placeholder text (often called lorem Ipsum text to push the form nearly to the bottom of the screen.

12. Run and test again. Now you see the problem! When the soft keyboard slides in the view it covers (or occults) some content behind it. The solution is a KeyboardAvoidingView.

13. Edit Checkout.js. Add a <KeyboardAvoidingView> around the root <ScrollView> of the component.

14. If you run and test again, you may or may not see any difference depending on how you're viewing your screen; iOS and Android behave differently with a KeyboardAvoidingView.

Let's try a couple of settings to see if they make a difference.

15. Add a behavior property to the KeyboardAvoidingView. Switch between position, padding, and height to see the differences. Set it to the one you like best.

16. Lastly try these settings:

```
<KeyboardAvoidingView keyboardVerticalOffset={100} behavior="position" enabled={true}>
```

17. Adjust the keyboardVerticalOffset until you like what you're seeing.

18. Before finishing, don't forget to delete all your Lorem Ipsum text. Then, you can be finished.