

**Workshop Practice 3**

**CS - Mobile Development**

In this practical lab, we will be practicing on Simple Animation App. This practice we will try to attempt and build an application that test all the different Animation methods.

We will be trying out a few Animation types:

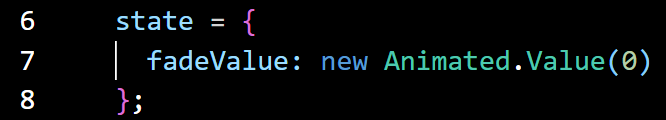
For the following steps, we will be using a Class component structure in your App.js



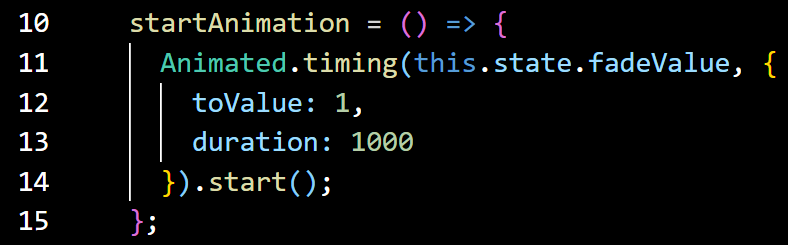
For the entire practice, we will be using the Animated API Library from react-native.

**Fade in Animation**

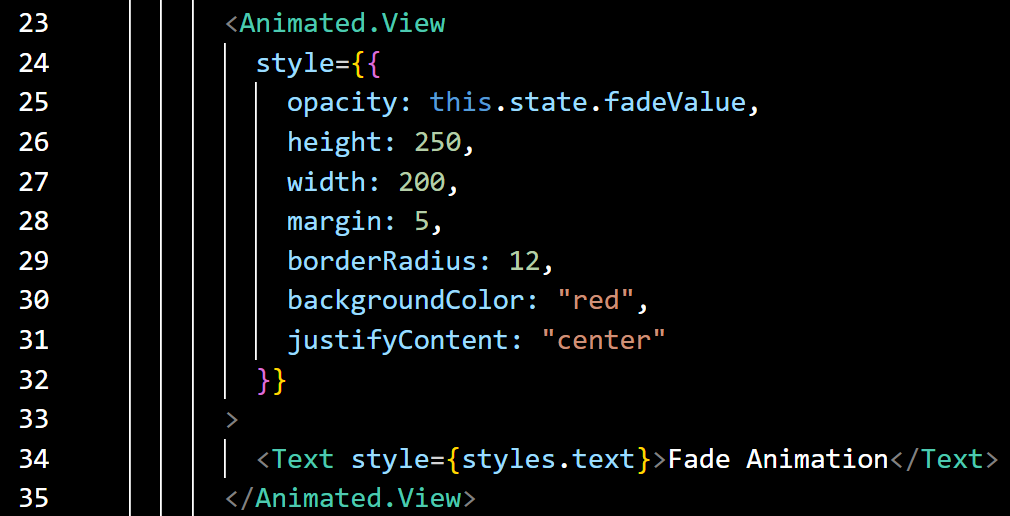
1. First, we need to initialize a value so we can give this value to the Animated method:



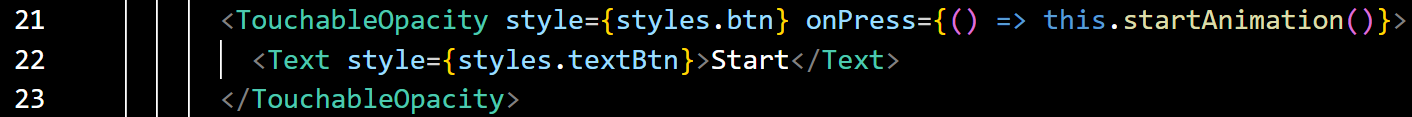
1. Then let’s create a function called **startAnimation** that will start our animations:



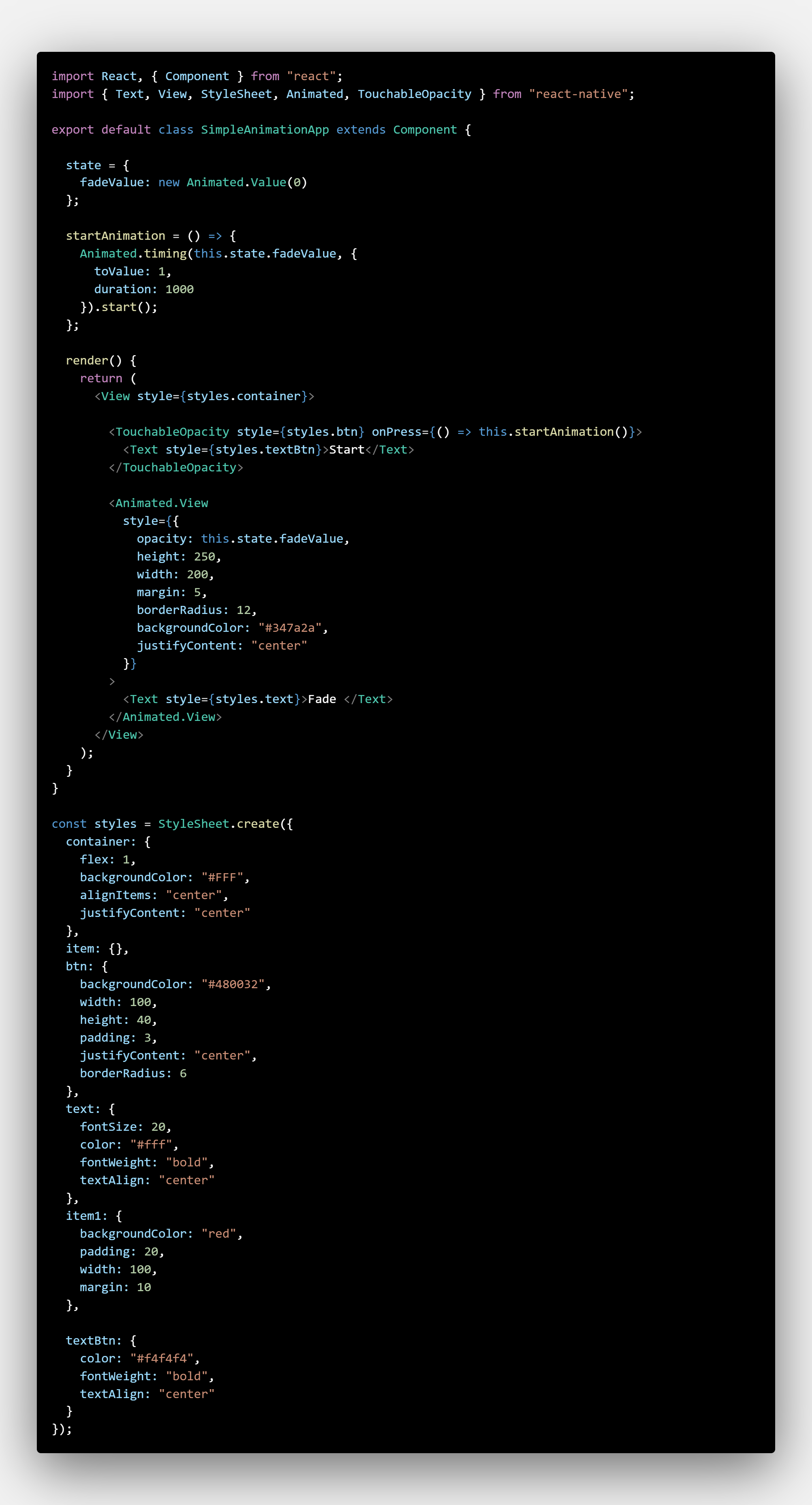
1. Now we have an animated value, the next step is to use it to animate the component. we use **Animated.View** as a wrapper to our component in order to make the animations happen!



1. Now let's call the startAnimation () function when an action is taken, in our case we call the function when button is clicked. you can trigger the function whenever you like for example within componentDidMount() .



1. In case, you have missed out some steps. Here is the overall code for the first Animation app.



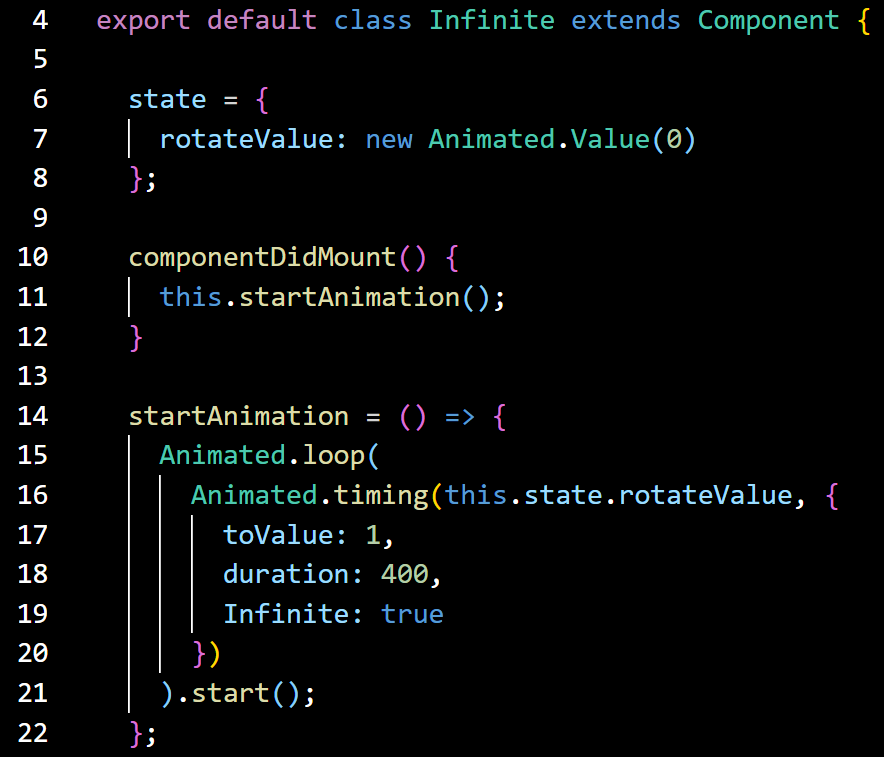
1. This is how your App should look like:



**Indefinite Animation**

Similar to the previous example’s steps. We will try to create an infinite loader that most apps have during loading.

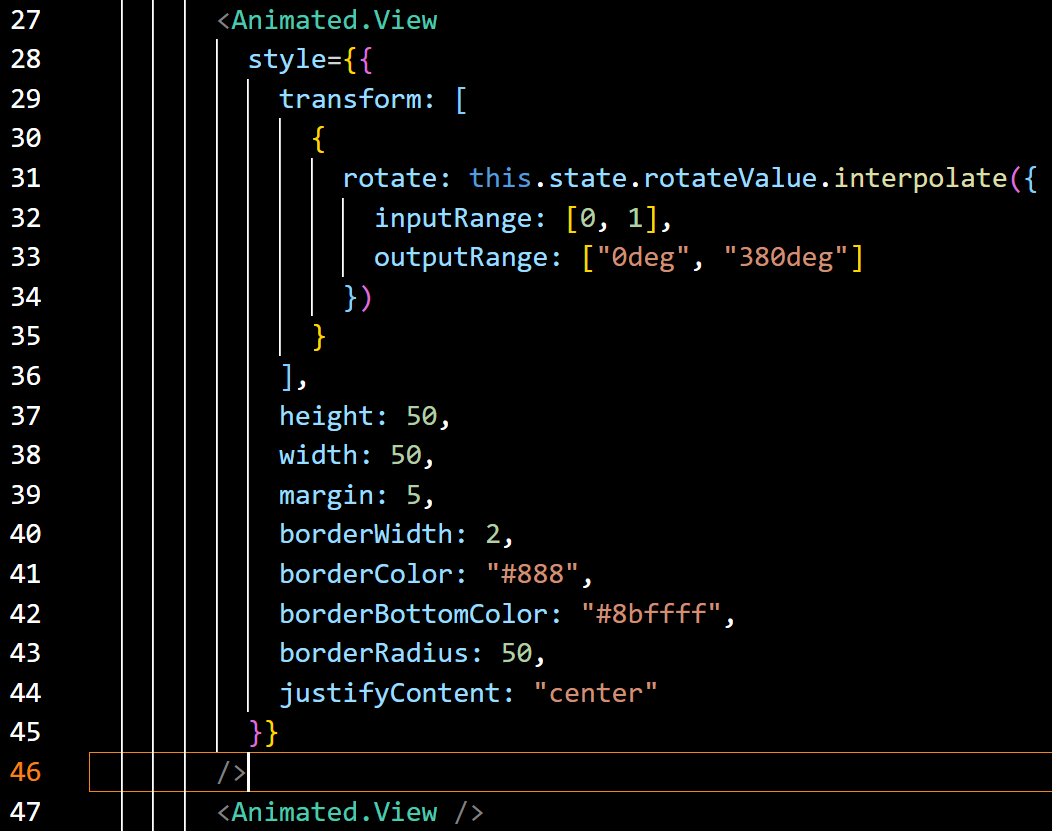
1. We will be using the Class component structure.



This time around, we will be using the **componentDidMount**() to execute the animation once the App is loaded.

We will be using the **Animated.loop** method to perform indefinite loop and something different would be the **Infinite** attribute set to true.

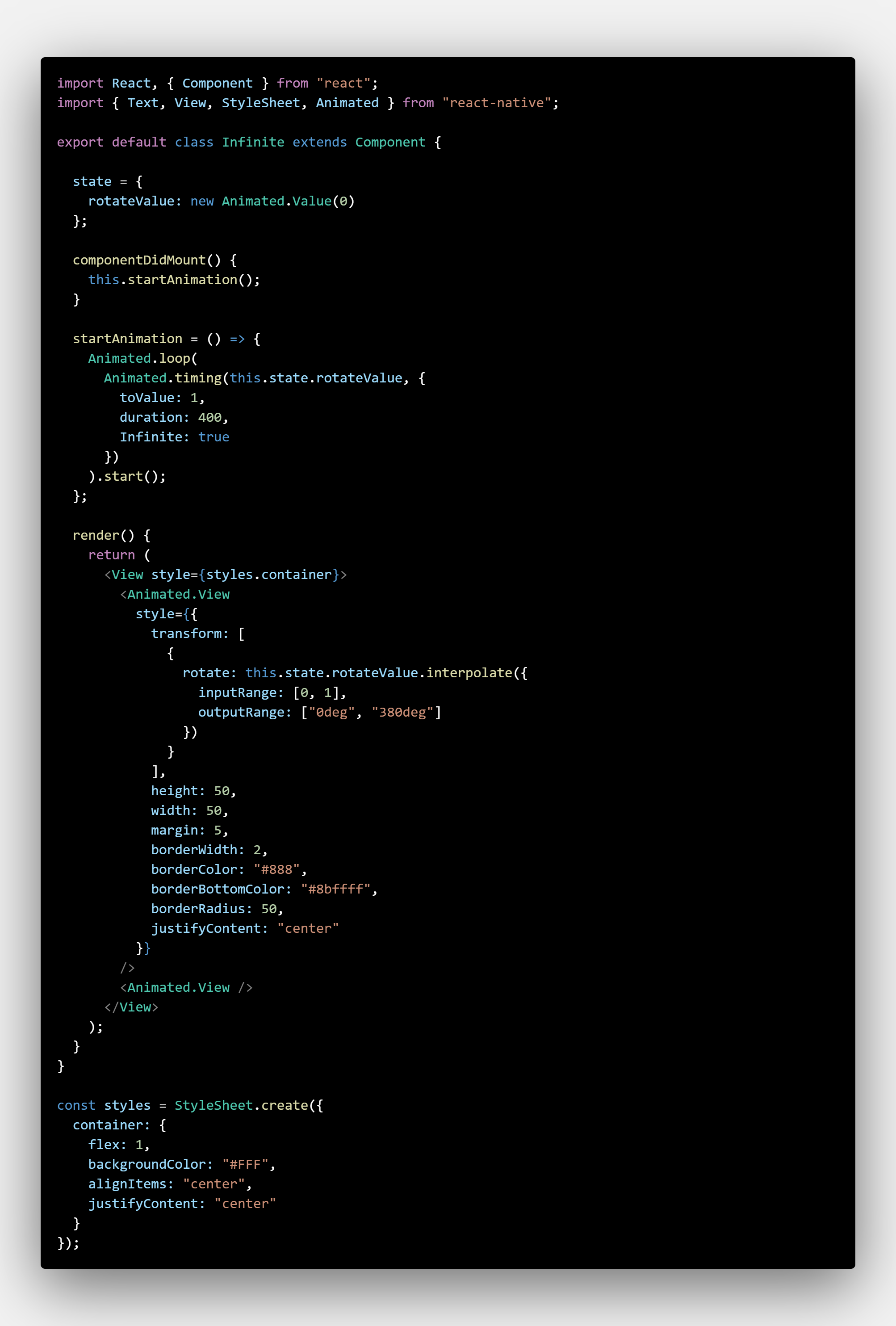
1. Next, we will use the Animated.View to render the infinite loader object.



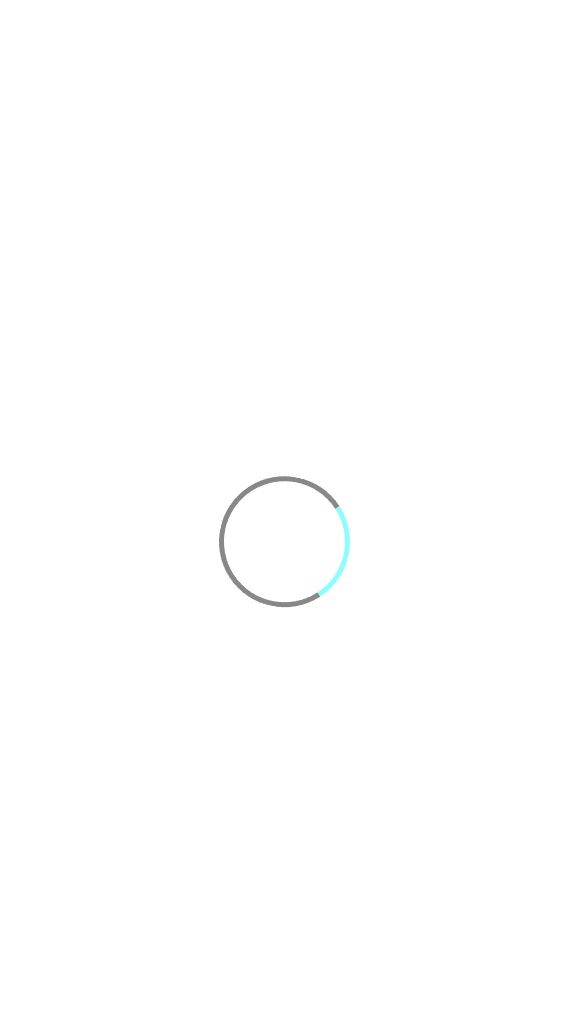
The **interpolate** method takes an Object of properties.

* **inputRange**: The start point of the **animatedValue**. For example, we want the **translateX** to start from 1 level.
* **outputRange**: Gives us an outputRange based on the inputRange.

1. Here, give it a try with the following full code-snippet.



1. The app should look similar to the screenshot below:



**A Group of Animation**

For executing a group of animation at once, we will require the use of the **parallel** method. Right after this practice, you can also try the **sequence** method, which loads the group of animation in sequence.

Give it a try with the following full code snippet. The code structure is similar to the previous two examples.

App.js Part 1 (Initialising and startAnimation method)



App.js Part 2 (render method)



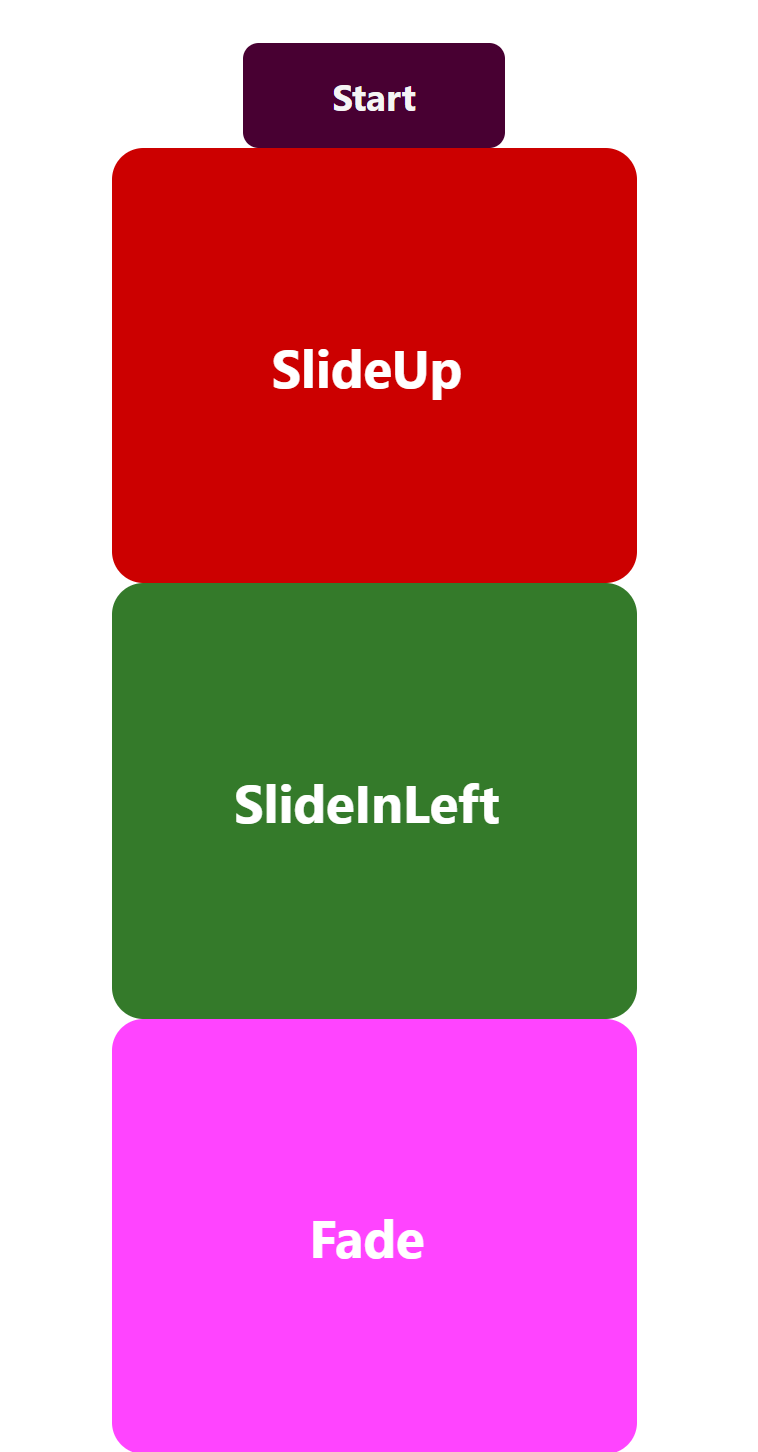
App.js Part 3 (render method)



App.js Part 4 (Stylesheet)



The App should look similar to the screenshot below:



Once you have completed the above, try changing the **Animated.parallel** to **Animated.sequence** and see what happens.