Version: 6.x

Different status bar configuration based on route

If you don't have a navigation header, or your navigation header changes color based on the route, you'll want to ensure that the correct color is used for the content.

Stack

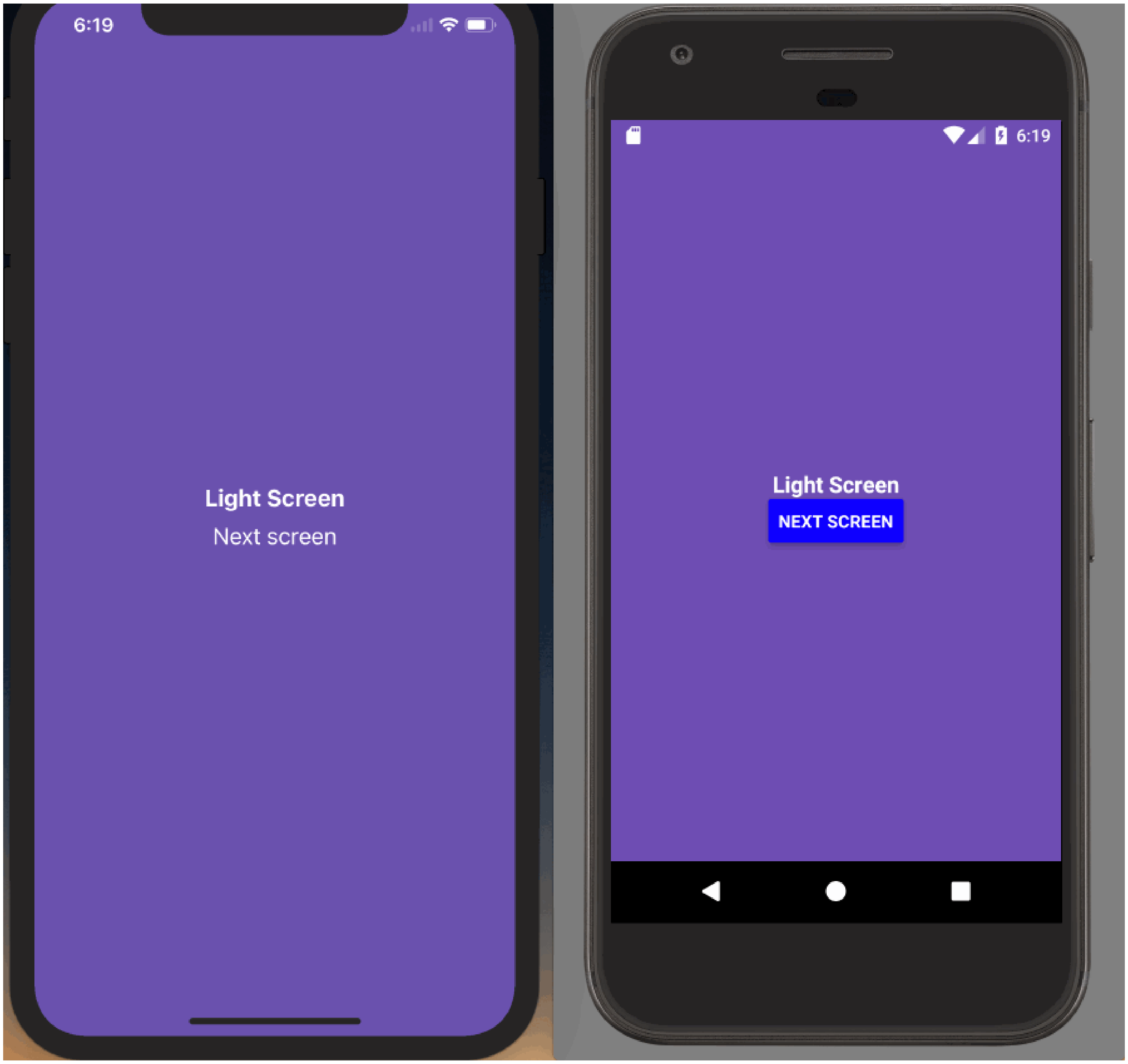
This is a simple task when using a stack. You can render the `StatusBar` component, which is exposed by React Native, and set your config.

```
import * as React from 'react';
import { View, Text, StatusBar, Button, StyleSheet } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import {
  SafeAreaProvider,
  useSafeAreaInsets,
} from 'react-native-safe-area-context';

function Screen1({ navigation }) {
  const insets = useSafeAreaInsets();

  return (
    <View
      style={[
        styles.container,
        {
          backgroundColor: '#6a51ae',
          paddingTop: insets.top,
          paddingBottom: insets.bottom,
          paddingLeft: insets.left,
          paddingRight: insets.right,
        },
      ]}
    >
```

```
      <StatusBar barStyle="light-content" backgroundColor="#6a51ae" />
      <Text style={{ color: '#fff' }}>Light Screen</Text>
      <Button
        title="Next screen"
        onPress={() => navigation.navigate('Screen2')}
        color="#fff"
      />
    </View>
  );
}

function Screen2({ navigation }) {
  const insets = useSafeAreaInsets();

  return (
    <View
      style={[
        styles.container,
        {
          backgroundColor: '#ecf0f1',
          paddingTop: insets.top,
          paddingBottom: insets.bottom,
          paddingLeft: insets.left,
          paddingRight: insets.right,
        },
      ]}
    >
      <StatusBar barStyle="dark-content" backgroundColor="#ecf0f1" />
      <Text>Dark Screen</Text>
      <Button
        title="Next screen"
        onPress={() => navigation.navigate('Screen1')}
      />
    </View>
  );
}

const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <SafeAreaProvider>
      <NavigationContainer>
        <Stack.Navigator screenOptions={{ headerShown: false }}>
```

```
          <Stack.Screen name="Screen1" component={Screen1} />
          <Stack.Screen name="Screen2" component={Screen2} />
        </Stack.Navigator>
      </NavigationContainer>
    </SafeAreaProvider>
  );
}

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

Try this example on Snack



Tabs and Drawer

If you're using a tab or drawer navigator, it's a bit more complex because all of the screens in the navigator might be rendered at once and kept rendered - that means that the last `StatusBar` config you set will be used (likely on the final tab of your tab navigator, not what the user is seeing).

To fix this, we'll have to do make the status bar component aware of screen focus and render it only when the screen is focused. We can achieve this by using the `useIsFocused` hook and creating a wrapper component:

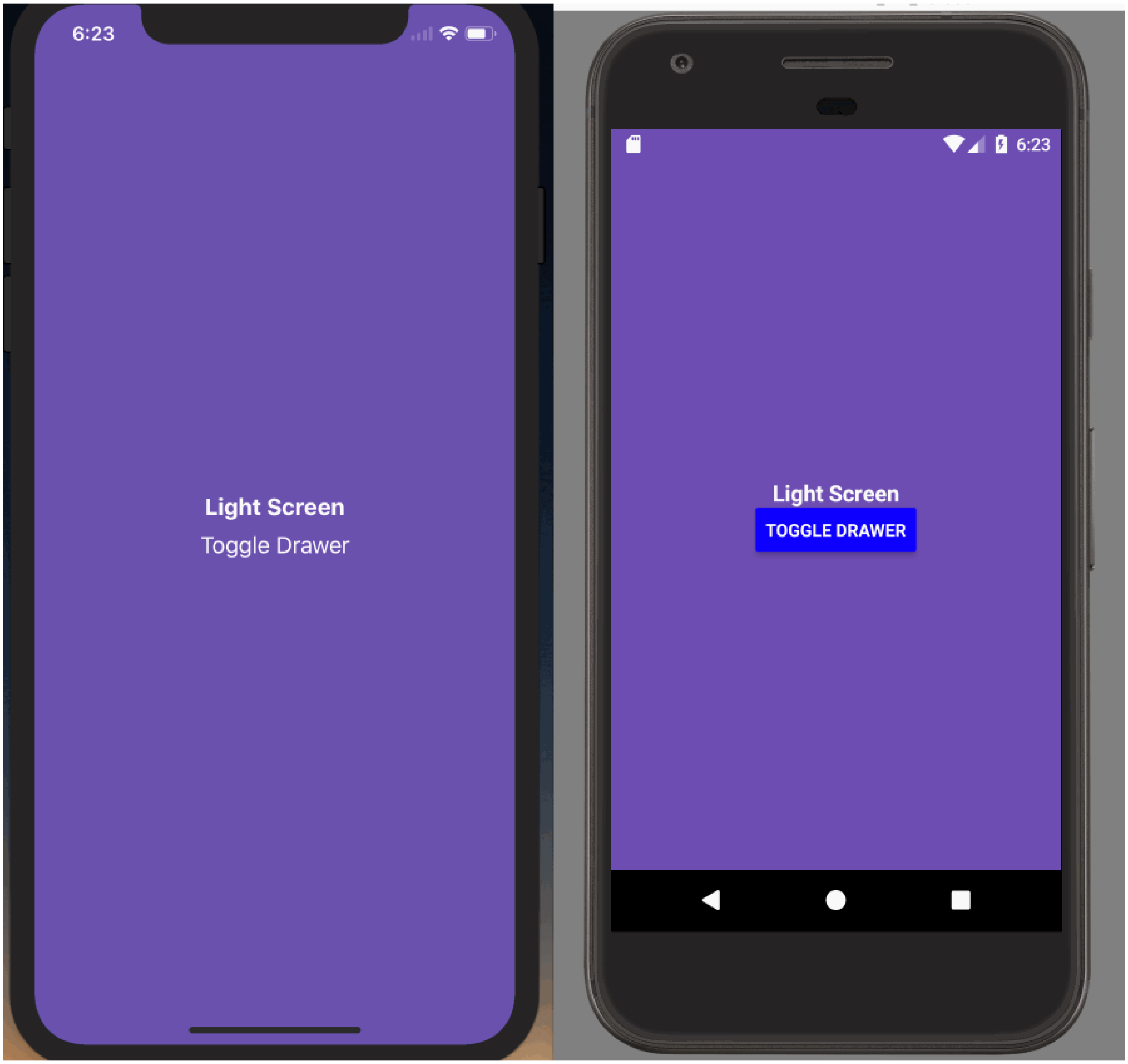
```
import * as React from 'react';
import { StatusBar } from 'react-native';
import { useIsFocused } from '@react-navigation/native';

function FocusAwareStatusBar(props) {
  const isFocused = useIsFocused();

  return isFocused ? <StatusBar {...props} /> : null;
}
```

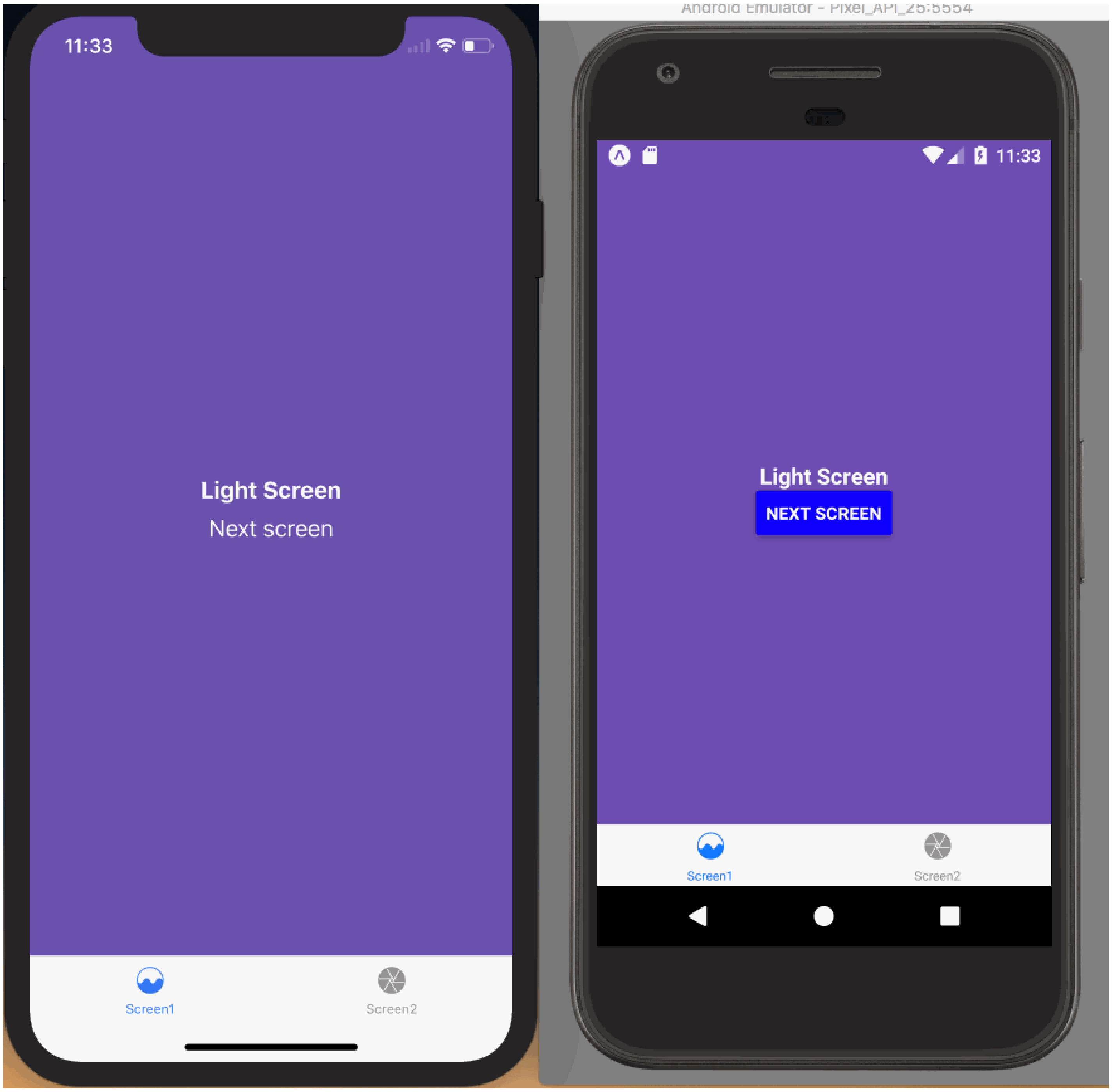
Now, our screens (both `Screen1.js` and `Screen2.js`) will use the `FocusAwareStatusBar` component instead of the `StatusBar` component from React Native:

```
function Screen1({ navigation }) {
  const insets = useSafeAreaInsets();

  return (
    <View
      style={[
        styles.container,
        {
          backgroundColor: '#6a51ae',
          paddingTop: insets.top,
          paddingBottom: insets.bottom,
          paddingLeft: insets.left,
          paddingRight: insets.right,
        },
      ]}
    >
      <FocusAwareStatusBar barStyle="light-content" backgroundColor="#6a51ae" />
      <Text style={{ color: '#fff' }}>Light Screen</Text>
      <Button
        title="Next screen"
        onPress={() => navigation.navigate('Screen2')}
        color="#fff"
      />
    </View>
  );
}

function Screen2({ navigation }) {
  const insets = useSafeAreaInsets();

  return (
    <View
      style={[
        styles.container,
        {
          backgroundColor: '#ecf0f1',
          paddingTop: insets.top,
          paddingBottom: insets.bottom,
          paddingLeft: insets.left,
          paddingRight: insets.right,
        },
      ]}
    >
      <FocusAwareStatusBar barStyle="dark-content" backgroundColor="#ecf0f1" />
      <Text>Dark Screen</Text>
      <Button
        title="Next screen"
        onPress={() => navigation.navigate('Screen1')}
      />
    </View>
  );
}
```

Try this example on Snack

Although not necessary, you can use the `FocusAwareStatusBar` component in the screens of the native stack navigator as well.


6:23
Light Screen
Toggle Drawer
6:23
Light Screen
TOGGLE DRAWER



Edit this page