



Version: 6.x

Supporting safe areas

By default, React Navigation tries to ensure that the elements of the navigators display correctly on devices with notches (e.g. iPhone X) and UI elements which may overlap the app content. Such items include:

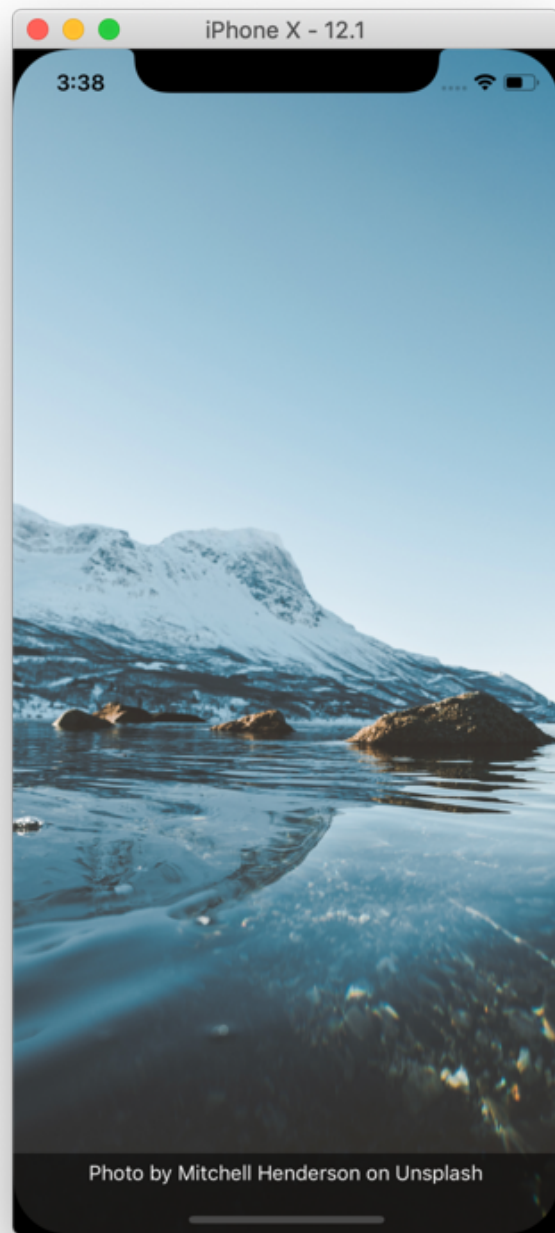
- Physical notches
- Status bar overlay
- Home activity indicator on iOS
- Navigation bar on Android

The area not overlapped by such items is referred to as "safe area".

We try to apply proper insets on the UI elements of the navigators to avoid being overlapped by such items. The goal is to (a) maximize usage of the screen (b) without hiding content or making it difficult to interact with by having it obscured by a physical display cutout or some operating system UI.

While React Navigation handles safe areas for the built-in UI elements by default, your own content may also need to handle it to ensure that content isn't hidden by these items.

It's tempting to solve (a) by wrapping your entire app in a container with padding that ensures all content will not be occluded. But in doing so, we waste a bunch of space on the screen, as pictured in the image on the left below. What we ideally want is the image pictured on the right.



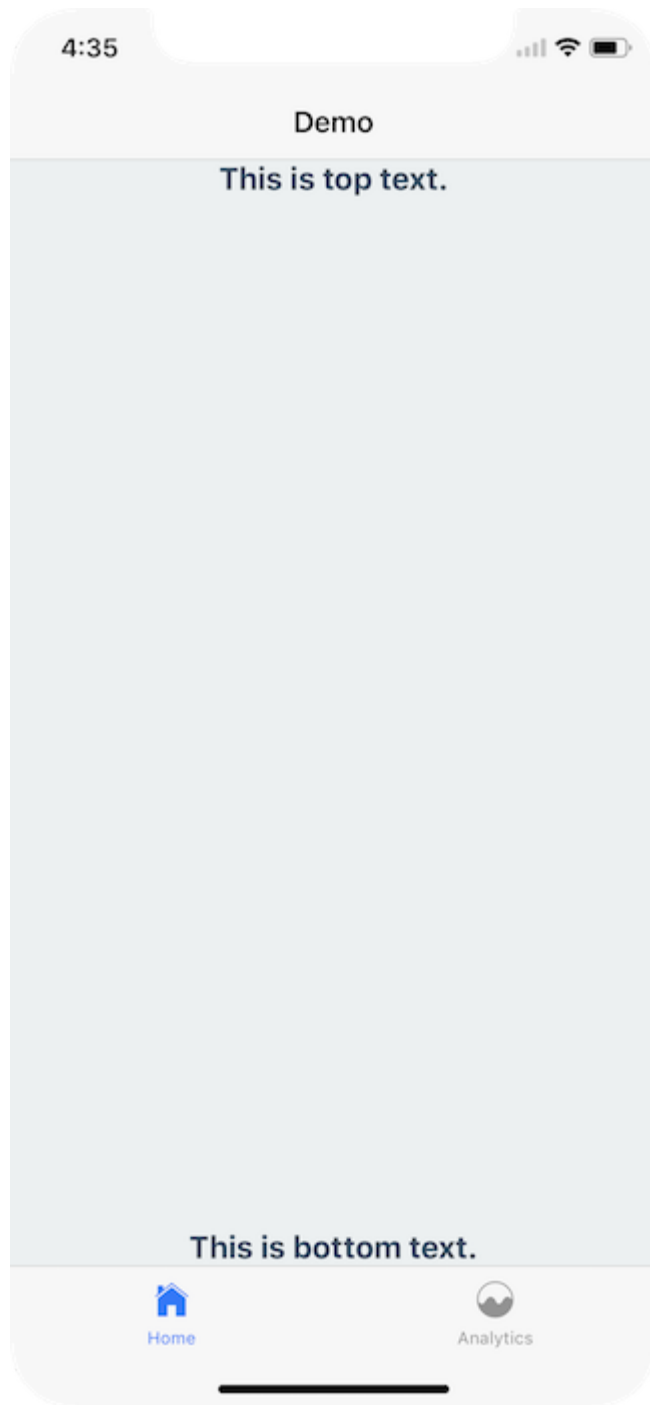
While React Native exports a `SafeAreaView` component, this component only supports iOS 10+ with no support for older iOS versions or Android. In addition, it also has some issues, i.e. if a screen containing safe area is animating, it causes jumpy behavior. So we recommend to use the `useSafeAreaInsets` hook from the `react-native-safe-area-context` library to handle safe areas in a more reliable way.

Note: The `react-native-safe-area-context` library also exports a `SafeAreaView` component. While it works on Android, it also has the same issues related to jumpy behavior

when animating. So we recommend always using the `useSafeAreaInsets` hook instead and avoid using the `SafeAreaView` component.

The rest of this guide gives more information on how to support safe areas in React Navigation.

Hidden/Custom Header or Tab Bar



React Navigation handles safe area in the default header. However, if you're using a custom header, it's important to ensure your UI is within the safe area.

For example, if I render nothing for the `header` or `tabBar`, nothing renders

```
import * as React from 'react';
import { Text, View } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

function Demo() {
  return (
    <View
      style={{ flex: 1, justifyContent: 'space-between', alignItems: 'center' }}
    >
      <Text>This is top text.</Text>
      <Text>This is bottom text.</Text>
    </View>
  );
}

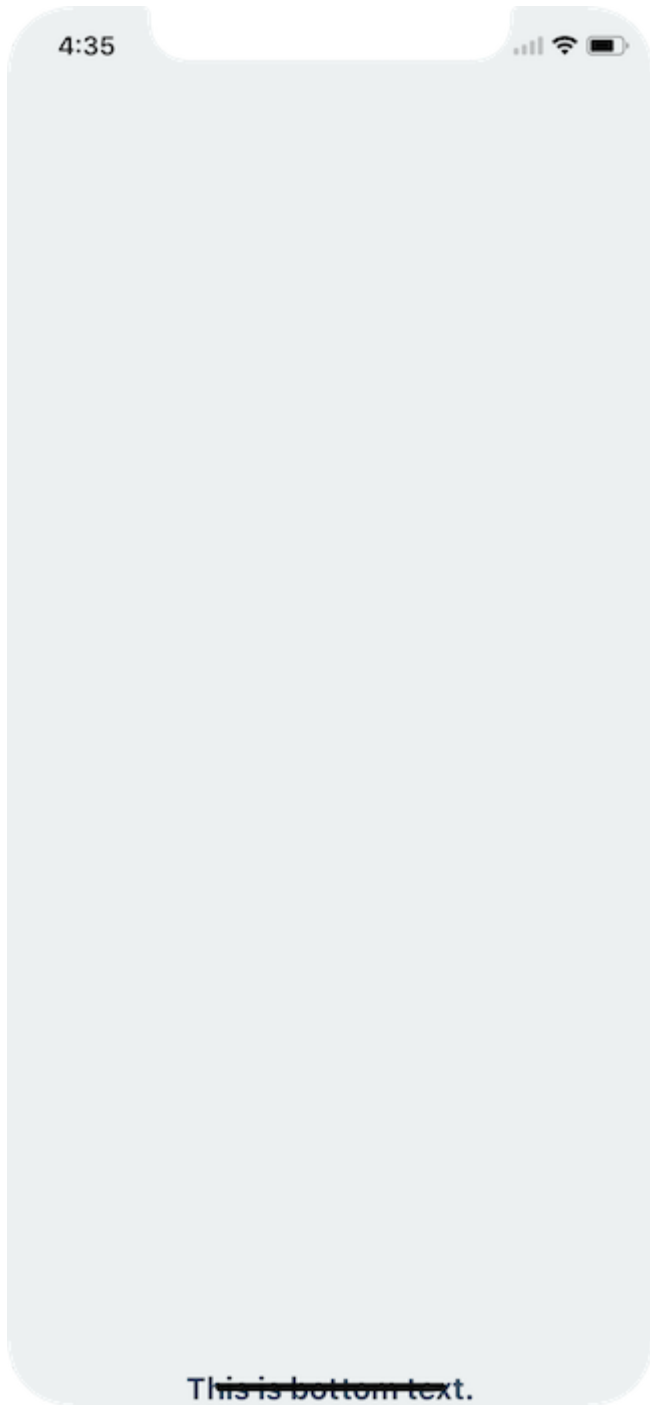
const Stack = createNativeStackNavigator();
const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator
        initialRouteName="Home"
        screenOptions={{ headerShown: false }}
      >
        <Stack.Screen name="Home">
          {() => (
            <Tab.Navigator
              initialRouteName="Analytics"
              tabBar={() => null}
              screenOptions={{ headerShown: false }}
            >
              <Tab.Screen name="Analytics" component={Demo} />
              <Tab.Screen name="Profile" component={Demo} />
            </Tab.Navigator>
          )}
        </Stack.Screen>

        <Stack.Screen name="Settings" component={Demo} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

```
    </NavigationContainer>  
  );  
}
```

Try this example on Snack [↗](#)



To fix this issue you can apply safe area insets on your content. This can be achieved using the `useSafeAreaInsets` hook from the `react-native-safe-area-context` library:

```
import {
  SafeAreaProvider,
  useSafeAreaInsets,
} from 'react-native-safe-area-context';

function Demo() {
  const insets = useSafeAreaInsets();

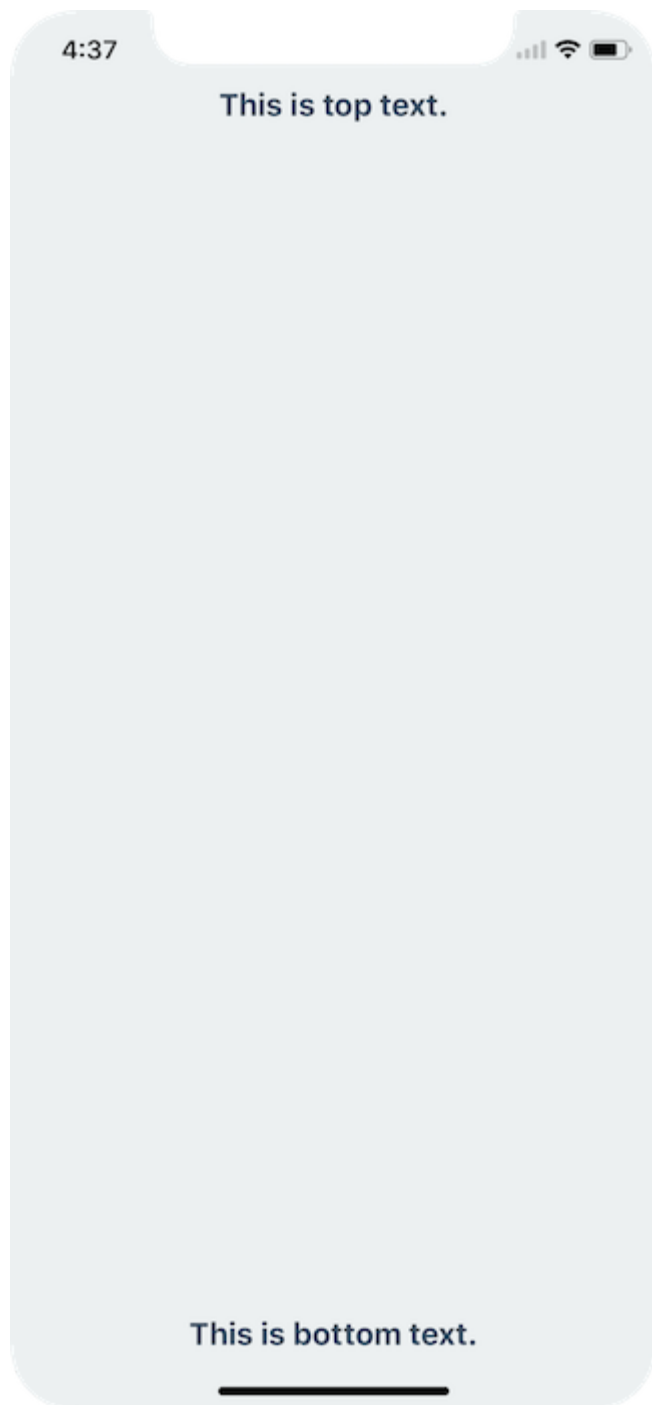
  return (
    <View
      style={{
        flex: 1,
        justifyContent: 'space-between',
        alignItems: 'center',

        // Paddings to handle safe area
        paddingTop: insets.top,
        paddingBottom: insets.bottom,
        paddingLeft: insets.left,
        paddingRight: insets.right,
      }}
    >
      <Text>This is top text.</Text>
      <Text>This is bottom text.</Text>
    </View>
  );
}

export default function App() {
  return (
    <SafeAreaProvider>
      <NavigationContainer>{/*(...) */</NavigationContainer>
    </SafeAreaProvider>
  );
}
```

Try this example on Snack [↗](#)

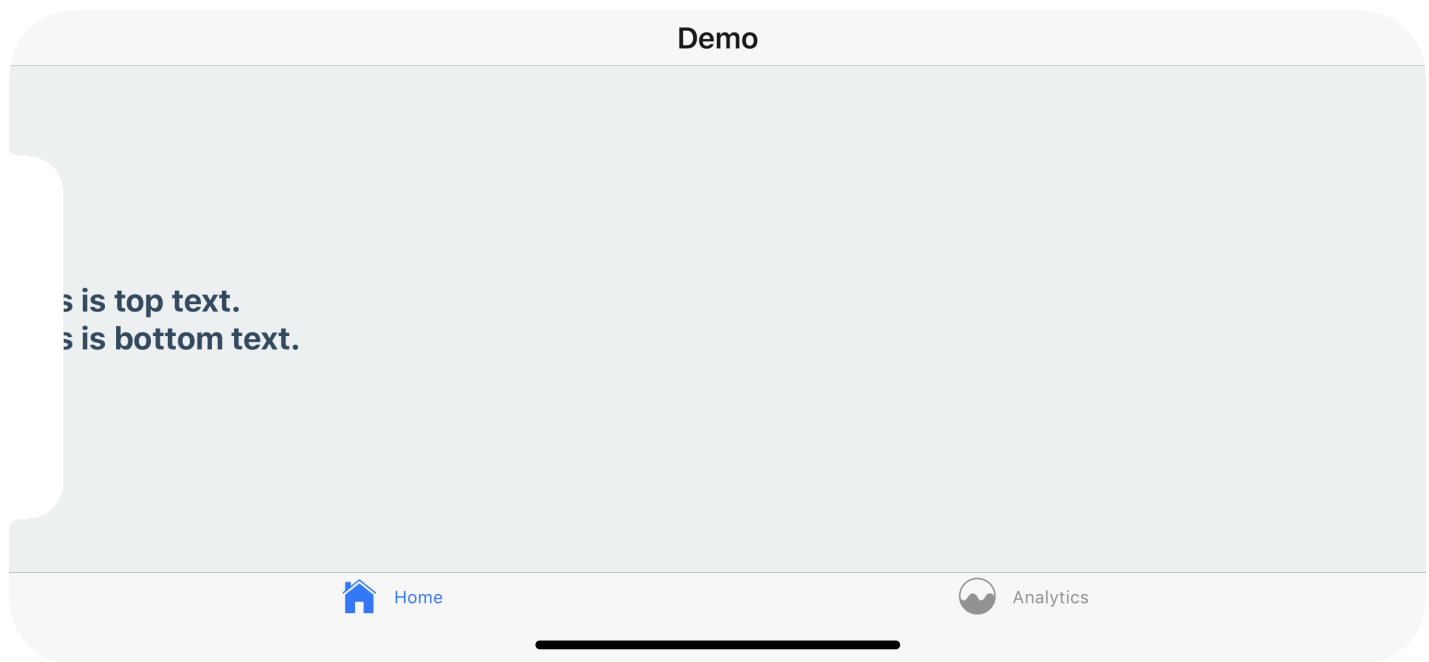
Make sure to wrap your app in `SafeAreaProvider` as per the instructions [here](#).



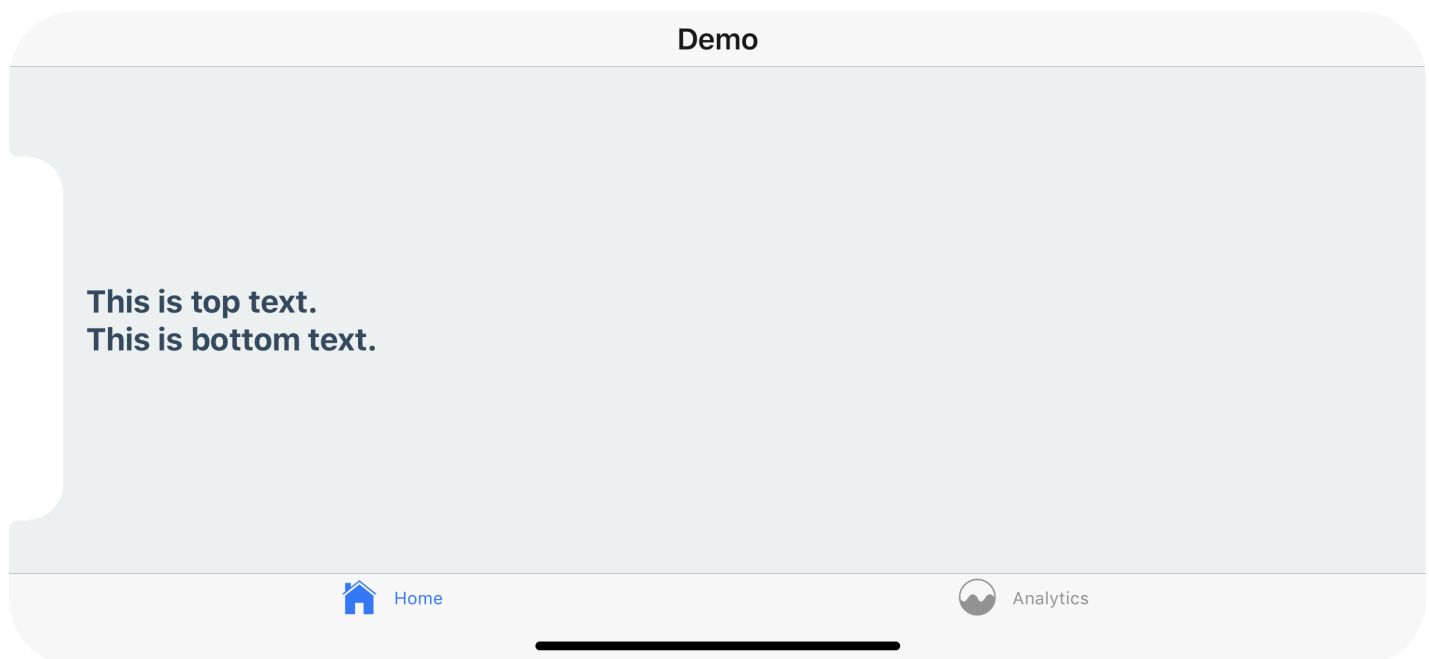
This will detect if the app is running on a device with notches, if so, ensure the content isn't hidden behind any hardware elements.

Landscape Mode

Even if you're using the default navigation bar and tab bar - if your application works in landscape mode it's important to ensure your content isn't hidden behind the sensor cluster.



To fix this you can, once again, apply safe area insets to your content. This will not conflict with the navigation bar nor the tab bar's default behavior in portrait mode.



Use the hook for more control

In some cases you might need more control over which paddings are applied. For example, you can only apply the top and the bottom padding by changing the `style` object:


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

function Demo() {
  const insets = useSafeAreaInsets();

  return (
    <View
      style={{
        paddingTop: insets.top,
        paddingBottom: insets.bottom,

        flex: 1,
        justifyContent: 'space-between',
        alignItems: 'center',
      }}
    >
      <Text>This is top text.</Text>
      <Text>This is bottom text.</Text>
    </View>
  );
}
```

Try this example on Snack [↗](#)

Similarly, you could apply these paddings in `contentContainerStyle` of `FlatList` to have the content avoid the safe areas, but still show them under the statusbar and navigation bar when scrolling.

Summary

- Use `useSafeAreaInsets` hook from `react-native-safe-area-context` instead of `SafeAreaView` component
- Don't wrap your whole app in `SafeAreaView`, instead apply the styles to content inside your screens
- Apply only specific insets using the `useSafeAreaInsets` hook for more control

[✎](#) Edit this page

