Version: 3.x

Shared Element Transitions

ⓘ INFO

This page was ported from an old version of the documentation.

As we're rewriting the documentation some of the pages might be a little outdated.

ⓘ CAUTION

Shared Element Transitions is an experimental feature, not recommended for production use yet. We are waiting for your feedback to improve implementation.

A Shared Element Transition allows you to smoothly transform a component from one screen into a component on another screen. When Reanimated detects that a component with a `sharedTransitionTag` is being mounted or unmounted, it tries to find the last registered view with the same `sharedTransitionTag`. If it finds two matching components, it takes a snapshot of the styles for both components, and both shared views are detached from their parent and attached to a temporary transition container for the duration of the animation. After the animation is complete, they are attached back to their original parent. If you do not create a custom animation, all snapshot properties, including `width`, `height`, `originX`, `originY`, and `transformMatrix`, are animated by default with a duration of 500ms using the `withTiming` animation.

How to use it?

To create a shared transition animation between two components on different screens, simply assign the same `sharedTransitionTag` to both components. When you navigate between screens, the shared transition animation will automatically play. The shared transition feature works by searching for two components that have been registered with the same `sharedTransitionTag`. If you want to use more than one shared view on the same screen, be sure to assign a unique shared tag to each component.

Screen A

```
<View
  sharedTransitionTag="sharedTag"
  style={{ width: 150, height: 150, backgroundColor: 'green' }}
/>
```

Screen B

```
<View
  sharedTransitionTag="sharedTag"
  style={{ width: 100, height: 100, backgroundColor: 'green' }}
/>
```

Custom animation

1. You need to create custom transition with Reanimated API, for example:

```
import { SharedTransition } from 'react-native-reanimated';
const transition = SharedTransition.custom((values) => {
  'worklet';
  return {
    height: withSpring(values.targetHeight),
    width: withSpring(values.targetWidth),
  };
});
```

2. You need to add custom transition as `sharedTransitionStyle` prop to your both components (on both screens):

```
<View
  sharedTransitionTag="reanimatedTransition"
  sharedTransitionStyle={transition}
  style={{ backgroundColor: 'blue', width: 200, height: 100 }}
/>
```

Examples

More examples of usage you can find in Reanimated Example App.

```
import * as React from 'react';
import { View, Button } from 'react-native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { NavigationContainer } from '@react-navigation/native';
import Animated from 'react-native-reanimated';

const Stack = createNativeStackNavigator();

function Screen1({ navigation }) {
  return (
    <View style={{ flex: 1 }}>
      <Animated.View
        style={{ width: 150, height: 150, backgroundColor: 'green' }}
        sharedTransitionTag="sharedTag"
      />
      <Button title="Screen2" onPress={() => navigation.navigate('Screen2')} />
    </View>
  );
}

function Screen2({ navigation }) {
  return (
    <View style={{ flex: 1, marginTop: 50 }}>
      <Animated.View
        style={{ width: 100, height: 100, backgroundColor: 'green' }}
        sharedTransitionTag="sharedTag"
      />
      <Button title="Screen1" onPress={() => navigation.navigate('Screen1')} />
    </View>
  );
}

export default function SharedElementExample() {
  return (
    <NavigationContainer>
      <Stack.Navigator screenOptions={{ headerShown: true }}>
        <Stack.Screen name="Screen1" component={Screen1} />
        <Stack.Screen name="Screen2" component={Screen2} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Limitation and known issues

- Only the native stack is supported.
- You can only animate `width`, `height`, `originX`, `originY`, and `transformMatrix` properties when using the shared transition.
- The layout for shared view children is not computed during the transition.
- The current implementation supports only the old React Native architecture (Paper)

Plans for future

- Enable the animation of all possible styles during a shared transition.
- Calculate the layout for shared view children during the transition.
- Add possibility to implement own screen transition with a customized animation for screen changes.
- Allow shared transition to work for components located within the same screen.
- Create a progress-based transition animation that can be controlled by swiping back (iOS only).
- Support for the new React Native architecture (Fabric)

Edit this page