



Guides

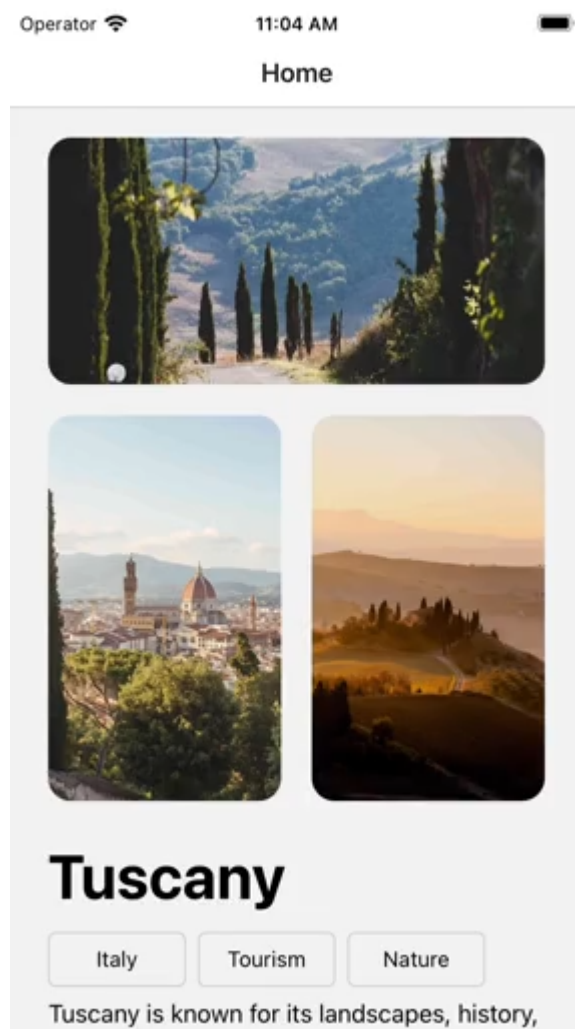
Animating elements between screens

Version: 6.x

# Animating elements between screens

This guide covers how to animate elements between screens. This feature is known as a Shared Element Transition and it's implemented in the `@react-navigation/native-stack` with React Native Reanimated.

Note: As of writing this guide, Shared Element Transitions are considered an experimental feature not recommended for production use.



## Pre-requisites

Before continuing this guide make sure your app meets these criteria:

- You are using `@react-navigation/native-stack`. The Shared Element Transitions feature isn't supported in JS-based `@react-navigation/stack`.
- You have `react-native-reanimated` **v3.0.0 or higher** installed and configured.

## Minimal example

To create a shared transition:

1. Use `Animated` components imported from `react-native-reanimated`.
2. Assign the same `sharedTransitionTag` to elements on different screens.
3. Navigate between screens. The transition will start automatically.

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

import Animated from 'react-native-reanimated';

const Stack = createNativeStackNavigator();

function HomeScreen({ navigation }) {
  return (
    <View style={styles.container}>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Details')}
      />
      <Animated.Image
        source={{ uri: 'https://picsum.photos/id/39/200' }}
        style={{ width: 300, height: 300 }}
        sharedTransitionTag="tag"
      />
    </View>
  );
}

function DetailsScreen({ navigation }) {
  return (
    <View style={styles.container}>
```

```
<Button title="Go back" onPress={() => navigation.goBack()} />
<Animated.Image
  source={{ uri: 'https://picsum.photos/id/39/200' }}
  style={{ width: 100, height: 100 }}
  sharedTransitionTag="tag"
/>
</View>
);
}

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Details" component={DetailsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

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

`sharedTransitionTag` is a string that has to be unique in the context of a single screen, but has to match elements between screens. This prop allows Reanimated to identify and animate the elements, similarly to the `key` property, which tells React which element in the list is which.

## Customizing the transition

By default, the transition animates the `width`, `height`, `originX`, `originY` and `transform` properties using `withTiming` with a 500 ms duration. You can easily customize `width`, `height`, `originX`, and `originY` props. Customizing `transform` is also possible but it's far beyond the scope of this guide.

Note: Custom SharedTransition API is not finalized and might change in a future release.

To customize the transition you need to pass all the properties besides `transform`.

```
import { SharedTransition } from 'react-native-reanimated';

const customTransition = SharedTransition.custom((values) => {
  'worklet';
  return {
    height: withSpring(values.targetHeight),
    width: withSpring(values.targetWidth),
    originX: withSpring(values.targetOriginX),
    originY: withSpring(values.targetOriginY),
  };
});

function HomeScreen() {
  return (
    <Animated.Image
      style={{ width: 300, height: 300 }}
      sharedTransitionTag="tag"
      sharedTransitionStyle={customTransition} // add this to both elements on
both screens
    />
  );
}
```


## Reference

You can find a full Shared Element Transitions reference in the [React Native Reanimated documentation](#).

## Alternatives

Alternatively, you can use `react-native-shared-element` library with a React Navigation binding which implements Shared Element Transitions in a JS-based `@react-navigation/stack` navigator. This solution, however, isn't actively maintained.

The `react-native-navigation` also comes with support for Shared Element Transitions. You can read more about it [here](#).

 [Edit this page](#)