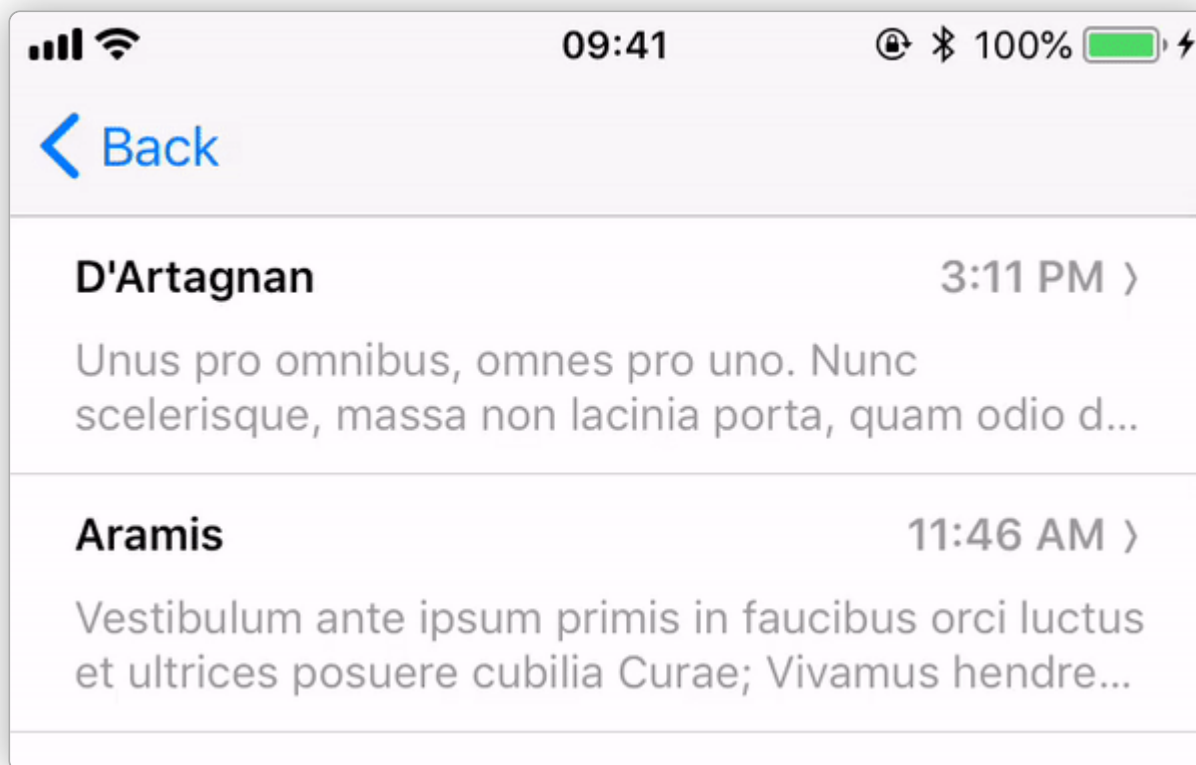


[API reference](#)[Components](#)[Swipeable](#)

Version: 2.6.0 – 2.12.0

# Swipeable



This component allows for implementing swipeable rows or similar interaction. It renders its children within a panable container allows for horizontal swiping left and right. While swiping one of two "action" containers can be shown depends on whether user swipes left or right (containers can be rendered by `renderLeftActions` or `renderRightActions` props).

## Usage:

Similarly to the `DrawerLayout`, `Swipeable` component isn't exported by default from the `react-native-gesture-handler` package. To use it, import it in the following way:

```
import Swipeable from 'react-native-gesture-handler/Swipeable';
```

## Properties

## **friction**

a number that specifies how much the visual interaction will be delayed compared to the gesture distance. e.g. value of 1 will indicate that the swipeable panel should exactly follow the gesture, 2 means it is going to be two times "slower".

## **leftThreshold**

distance from the left edge at which released panel will animate to the open state (or the open panel will animate into the closed state). By default it's a half of the panel's width.

## **rightThreshold**

distance from the right edge at which released panel will animate to the open state (or the open panel will animate into the closed state). By default it's a half of the panel's width.

## **overshootLeft**

a boolean value indicating if the swipeable panel can be pulled further than the left actions panel's width. It is set to `true` by default as long as the left panel render method is present.

## **overshootRight**

a boolean value indicating if the swipeable panel can be pulled further than the right actions panel's width. It is set to `true` by default as long as the right panel render method is present.

## **overshootFriction**

a number that specifies how much the visual interaction will be delayed compared to the gesture distance at overshoot. Default value is 1, it mean no friction, for a native feel, try 8 or above.

## **onSwipeableLeftOpen**

### CAUTION

This callback is deprecated and will be removed in the next version. Please use

`onSwipeableOpen(direction)`

method that is called when left action panel gets open.

### **onSwipeableRightOpen**

#### CAUTION

This callback is deprecated and will be removed in the next version. Please use

`onSwipeableOpen(direction)`

method that is called when right action panel gets open.

### **onSwipeableOpen**

method that is called when action panel gets open (either right or left). Takes swipe direction as an argument.

### **onSwipeableClose**

method that is called when action panel is closed. Takes swipe direction as an argument.

### **onSwipeableLeftWillOpen**

#### CAUTION

This callback is deprecated and will be removed in the next version. Please use

`onSwipeableWillOpen(direction)`

method that is called when left action panel starts animating on open.

### **onSwipeableRightWillOpen**

#### CAUTION

This callback is deprecated and will be removed in the next version. Please use

`onSwipeableWillOpen(direction)`

method that is called when right action panel starts animating on open.

**onSwipeableWillOpen**

method that is called when action panel starts animating on open (either right or left). Takes swipe direction as an argument.

**onSwipeableWillClose**

method that is called when action panel starts animating on close. Takes swipe direction as an argument.

**renderLeftActions**

method that is expected to return an action panel that is going to be revealed from the left side when user swipes right. This map describes the values to use as inputRange for extra interpolation: AnimatedValue: [startValue, endValue]

progressAnimatedValue: [0, 1] dragAnimatedValue: [0, +]

To support `rtl` flexbox layouts use `flexDirection` styling.

**renderRightActions**

method that is expected to return an action panel that is going to be revealed from the right side when user swipes left. This map describes the values to use as inputRange for extra interpolation: AnimatedValue: [startValue, endValue]

progressAnimatedValue: [0, 1] dragAnimatedValue: [0, -]

To support `rtl` flexbox layouts use `flexDirection` styling.

**containerStyle**

style object for the container (Animated.View), for example to override `overflow: 'hidden'`.

**childrenContainerStyle**

style object for the children container (Animated.View), for example to apply `flex: 1`.

**enableTrackpadTwoFingerGesture (iOS only)**

Enables two-finger gestures on supported devices, for example iPads with trackpads. If not enabled the gesture will require click + drag, with `enableTrackpadTwoFingerGesture` swiping with two fingers will also trigger the gesture.

## Methods

Using reference to `Swipeable` it's possible to trigger some actions on it

### `close`

method that closes component.

### `openLeft`

method that opens component on left side.

### `openRight`

method that opens component on right side.

## Example:

See the [swipeable example](#) from GestureHandler Example App or view it directly on your phone by visiting [our expo demo](#).

```
import React, { Component } from 'react';
import { Animated, StyleSheet, View } from 'react-native';
import { RectButton } from 'react-native-gesture-handler';
import Swipeable from 'react-native-gesture-handler/Swipeable';

class AppleStyleSwipeableRow extends Component {
  renderLeftActions = (progress, dragX) => {
    const trans = dragX.interpolate({
      inputRange: [0, 50, 100, 101],
      outputRange: [-20, 0, 0, 1],
    });
  };
  return (
    <RectButton style={styles.leftAction} onPress={this.close}>
      <Animated.Text
        style={[
          styles.actionText,
          {

```

```
        transform: [{ translateX: trans }],
      },
    ]}]>
    Archive
  </Animated.Text>
</RectButton>
);
};
render() {
  return (
    <Swipeable renderLeftActions={this.renderLeftActions}>
      <Text>"hello"</Text>
    </Swipeable>
  );
}
```