

Reference:
<https://reactnative.dev/docs/touchableopacity>

Pressable

TouchableOpacity · React Native

reactnative.dev/docs/touchableopacity

Pressable1/2

Join us for React Conf on Oct 7-8. [Learn more.](#)

React Native0.82DevelopmentContributingCommunityShowcaseBlog

ButtonFlatListImageImageBackgroundKeyboardAvoidingViewModalPressableRefreshControlScrollViewSectionListStatusBarSwitchTextTextInputTouchableHighlightTouchableOpacity

TouchableOpacity

TIP

If you're looking for a more extensive and future-proof way to handle touch-based input, check out the [Pressable](#) API.

A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.

Opacity is controlled by wrapping the children in an `Animated.View`, which is added to the view hierarchy. Be aware that this can affect layout.

Example

Props

TouchableWithoutFeedback

Props

style

activeOpacity

hasTVPreferredFocus

nextFocusDown

nextFocusForward

nextFocusLeft

nextFocusRight

nextFocusUp

ref

onPress in `<Button>`, `<TouchableOpacity>`, `<Pressable>`

- It detects and responds to touch interactions on any of its child components.
- It's a flexible, low-level wrapper that provides callbacks for various press states (**press start**, **press end**, **long press**, etc.) and allows you to customize the visual feedback during user interactions.

It is as the modern, more versatile **replacement** for ~~TouchableOpacity~~, ~~TouchableHighlight~~, and other older touchable components such ~~as~~ `Button`.

```
return (  
  <View style={styles.container}>  
    <Pressable  
      onPress={handlePress}  
      style={({ pressed }) => [  
        styles.button,  
        pressed && styles.buttonPressed  
      ]}  
    >  
      {({ pressed }) => (  
        <Text style={styles.buttonText}>  
          {pressed ? 'Pressing...' : 'Press Me'}  
        </Text>  
      )}  
    </Pressable>  
  
    <Text style={styles.counter}>Press Count: {count}</Text>  
  </View>  
);
```

- **onPress callback** - Executes when the button is pressed
- **Dynamic styling** - The style prop receives a `pressed` boolean to change appearance during press
- **Dynamic children** - The children function receives `pressed` state to update text during interaction



- **State management** - Uses `useState` to track **press count**
- **Visual feedback** - Button changes color and opacity when pressed





```
import React, { useState } from 'react';
import { View, Text, Pressable, StyleSheet, Alert } from 'react-native';

const App = () => {
  const [count, setCount] = useState(0);

  const handlePress = () => {
    setCount(count + 1);
    Alert.alert('Button Pressed', `You've pressed the button ${count + 1} times`);
  };

  return();

};

export default App;
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#f5f5f5',
  },
  button: {
    backgroundColor: '#007AFF',
    paddingHorizontal: 30,
    paddingVertical: 15,
    borderRadius: 10,
    elevation: 3,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.25,
    shadowRadius: 3.84,
  },
  buttonPressed: {
    backgroundColor: '#0056b3',
    opacity: 0.8,
  },
  buttonText: {
    color: 'white',
    fontSize: 18,
    fontWeight: 'bold',
  },
  counter: {
    marginTop: 20,
    fontSize: 16,
    color: '#333',
  },
});
```