Handling Touches

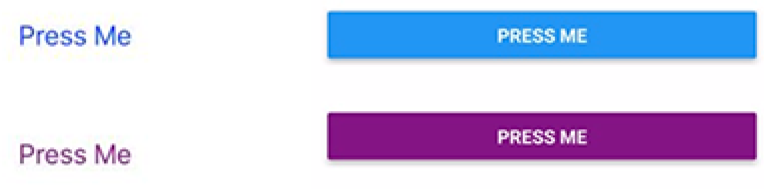
Users interact with mobile apps mainly through touch. They can use a combination of gestures, such as tapping on a button, scrolling a list, or zooming on a map. React Native provides components to handle all sorts of common gestures, as well as a comprehensive gesture responder system to allow for more advanced gesture recognition, but the one component you will most likely be interested in is the basic Button.

Displaying a basic button

Button provides a basic button component that is rendered nicely on all platforms. The minimal example to display a button looks like this:

```
<Button
  onPress={() => {
    console.log('You tapped the button!');
  }}
  title="Press Me"
/>
```

This will render a blue label on iOS, and a blue rounded rectangle with light text on Android. Pressing the button will call the "onPress" function, which in this case displays an alert popup. If you like, you can specify a "color" prop to change the color of your button.



Go ahead and play around with the `Button` component using the example below. You can select which platform your app is previewed in by clicking on the toggle in the bottom right and then clicking on "Tap to Play" to preview the app.

Button Basics Expo

```
import React, {Component} from 'react';
import {Alert, Button, StyleSheet, View} from 'react-
native';

export default class ButtonBasics extends Component {
  _onPressButton() {
    Alert.alert('You tapped the button!');
  }

  render() {
    return (
      <View style={styles.container}>
        <View style={styles.buttonContainer}>
          <Button onPress={this._onPressButton}
title="Press Me" />
        </View>
        <View style={styles.buttonContainer}>
          <Button
            onPress={this._onPressButton}
            title="Press Me"
            color="#841584"
          />
        </View>
        <View style=
{styles.alternativeLayoutButtonContainer}>
          <Button onPress={this._onPressButton} title="This
```

Preview My Device iOS Android Web

Touchables

If the basic button doesn't look right for your app, you can build your own button using any of the "Touchable" components provided by React Native. The "Touchable" components provide the capability to capture tapping gestures, and can display feedback when a gesture is recognized. These components do not provide any default styling, however, so you will need to do a bit of work to get them looking nicely in your app.

Which "Touchable" component you use will depend on what kind of feedback you want to provide:

- Generally, you can use **TouchableHighlight** anywhere you would use a button or link on web. The view's background will be darkened when the user presses down on the button.

- You may consider using **TouchableNativeFeedback** on Android to display ink surface reaction ripples that respond to the user's touch.
- **TouchableOpacity** can be used to provide feedback by reducing the opacity of the button, allowing the background to be seen through while the user is pressing down.
- If you need to handle a tap gesture but you don't want any feedback to be displayed, use **TouchableWithoutFeedback**.

In some cases, you may want to detect when a user presses and holds a view for a set amount of time. These long presses can be handled by passing a function to the `onLongPress` props of any of the "Touchable" components.

Let's see all of these in action:

Touchables

Expo

```
import React, {Component} from 'react';
import {
  Alert,
  Platform,
  StyleSheet,
  Text,
  TouchableHighlight,
  TouchableOpacity,
  TouchableNativeFeedback,
  TouchableWithoutFeedback,
  View,
} from 'react-native';

export default class Touchables extends Component {
  _onPressButton() {
    Alert.alert('You tapped the button!');
  }

  _onLongPressButton() {
    Alert.alert('You long-pressed the button!');
  }

  render() {
    return (
      <View style={styles.container}>
        <TouchableHighlight onPress={this._onPressButton}
```

Preview My Device iOS Android Web

Scrolling and swiping

Gestures commonly used on devices with touchable screens include swipes and pans. These allow the user to scroll through a list of items, or swipe through pages of content. For these, check out the ScrollView Core Component.

Known issues

- react-native#29308: The touch area never extends past the parent view bounds and on Android negative margin is not supported.