Pressable

Pressable is a Core Component wrapper that can detect various stages of press interactions on any of its defined children.

```
<Pressable onPress={onPressFunction}>
  <Text>I'm pressable!</Text>
</Pressable>
```

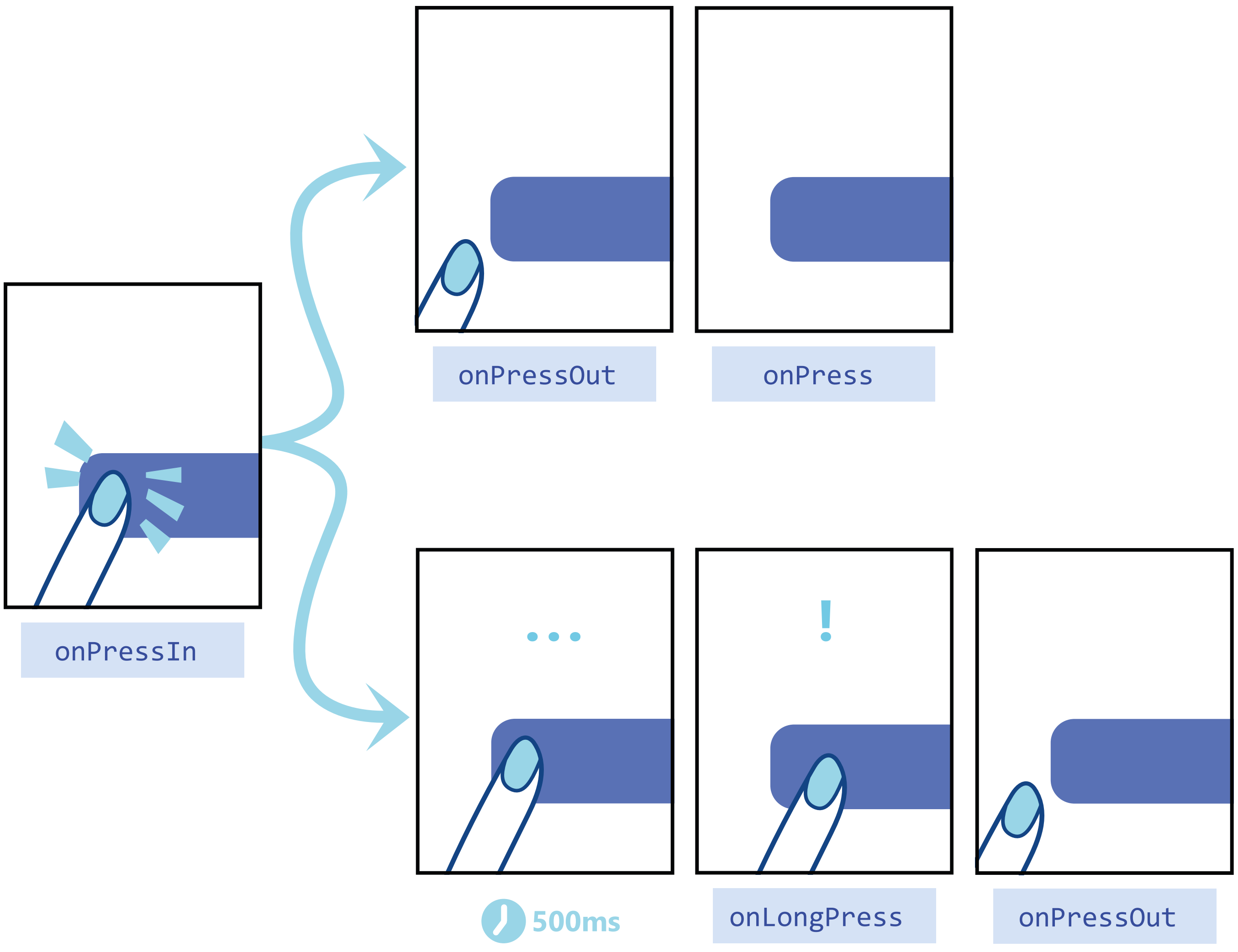
How it works

On an element wrapped by `Pressable`:

- `onPressIn` is called when a press is activated.
- `onPressOut` is called when the press gesture is deactivated.

After pressing `onPressIn`, one of two things will happen:

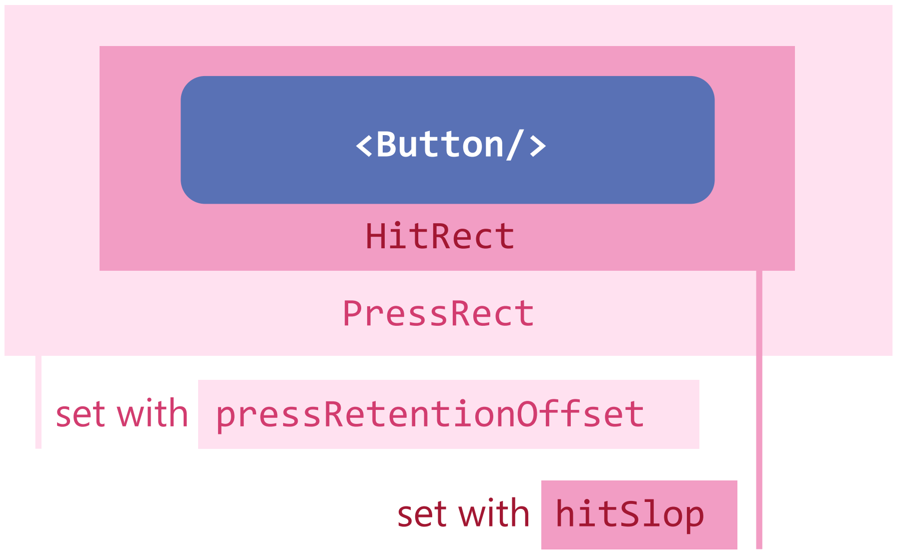
1. The person will remove their finger, triggering `onPressOut` followed by `onPress`.
2. If the person leaves their finger longer than 500 milliseconds before removing it, `onLongPress` is triggered. (`onPressOut` will still fire when they remove their finger.)



Fingers are not the most precise instruments, and it is common for users to accidentally activate the wrong element or miss the activation area. To help, `Pressable` has an optional `HitRect` you can use to define how far a touch can register away from the wrapped element. Presses can start anywhere within a `HitRect`.

`PressRect` allows presses to move beyond the element and its `HitRect` while maintaining activation and being eligible for a "press"—think of sliding your finger slowly away from a button you're pressing down on.

> The touch area never extends past the parent view bounds and the Z-index of sibling views always takes precedence if a touch hits two overlapping views.



You can set `HitRect` with `hitSlop` and set `PressRect` with `pressRetentionOffset`.

> `Pressable` uses React Native's `Pressability` API. For more information around the state machine flow of Pressability and how it works, check out the implementation for Pressability.

Example

Pressable ⋀ Expo

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

const App = () => {
  const [timesPressed, setTimesPressed] = useState(0);

  let textLog = '';
  if (timesPressed > 1) {
    textLog = timesPressed + 'x onPress';
  } else if (timesPressed > 0) {
    textLog = 'onPress';
  }

  return (
    <View style={styles.container}>
      <Pressable
        onPress={() => {
          setTimesPressed(current => current + 1);
        }}
        style={({pressed}) => [
          {
            backgroundColor: pressed ? 'rgb(210, 230, 255)'
: 'white',
          },
          styles.wrapperCustom,
```

Preview My Device iOS Android Web

Props

`android_disableSound` Android

If true, doesn't play Android system sound on press.

TYPE	DEFAULT
boolean	`false`

`android_ripple` Android

Enables the Android ripple effect and configures its properties.

TYPE
RippleConfig

`children`

Either children or a function that receives a boolean reflecting whether the component is currently pressed.

TYPE
React Node

`unstable_pressDelay`

Duration (in milliseconds) to wait after press down before calling `onPressIn`.

TYPE
number

`delayLongPress`

Duration (in milliseconds) from `onPressIn` before `onLongPress` is called.

TYPE	DEFAULT
number	`500`

`disabled`

Whether the press behavior is disabled.

TYPE	DEFAULT
boolean	`false`

`hitSlop`

Sets additional distance outside of element in which a press can be detected.

TYPE
Rect or number

`onHoverIn`

Called when the hover is activated to provide visual feedback.

TYPE
`({ nativeEvent: MouseEvent }) => void`

`onHoverOut`

Called when the hover is deactivated to undo visual feedback.

TYPE
`({ nativeEvent: MouseEvent }) => void`

`onLongPress`

Called if the time after `onPressIn` lasts longer than 500 milliseconds. This time period can be customized with `delayLongPress`.

TYPE
`({nativeEvent: PressEvent}) => void`

`onPress`

Called after `onPressOut`.

TYPE
`({nativeEvent: PressEvent}) => void`

`onPressIn`

Called immediately when a touch is engaged, before `onPressOut` and `onPress`.

TYPE
`({nativeEvent: PressEvent}) => void`

`onPressOut`

Called when a touch is released.

TYPE
`({nativeEvent: PressEvent}) => void`

`pressRetentionOffset`

Additional distance outside of this view in which a touch is considered a press before `onPressOut` is triggered.

TYPE	DEFAULT
Rect or number	`{bottom: 30, left: 20, right: 20, top: 20}`

`style`

Either view styles or a function that receives a boolean reflecting whether the component is currently pressed and returns view styles.

TYPE
View Style

`testOnly_pressed`

Used only for documentation or testing (e.g. snapshot testing).

TYPE	DEFAULT
boolean	`false`

Type Definitions

RippleConfig

Ripple effect configuration for the `android_ripple` property.

TYPE
object

Properties:

NAME	TYPE	REQUIRED	DESCRIPTION
color	color	No	Defines the color of the ripple effect.
borderless	boolean	No	Defines if ripple effect should not include border.
radius	number	No	Defines the radius of the ripple effect.
foreground	boolean	No	Set to true to add the ripple effect to the foreground of the view, instead of the background. This is useful if one of your child views has a background of its own, or you're e.g. displaying images, and you don't want the ripple to be covered by them.

Is this page useful? 👍 👎

Edit this page

*Last updated on **Aug 17, 2023***