

Lesson:

Touchable Highlight



The `TouchableHighlight` component in React Native provides a touchable area that highlights when pressed. It is a wrapper component that enables touch interactions for its child components. The `TouchableHighlight` component is useful when you want to add interactive elements to your app, such as buttons or clickable areas.

Using the TouchableHighlight Component

To use the `TouchableHighlight` component, wrap your content inside it and provide an `onPress` callback function to handle the press event. Here's an example:

```
1 import React from 'react';
2 import { View, TouchableHighlight, Text, StyleSheet } from 'react-native';
3
4 const App = () => {
5   const handlePress = () => {
6     console.log('Button Pressed!');
7   };
8
9   return (
10     <View style={styles.container}>
11       <TouchableHighlight
12         style={styles.button}
13         underlayColor="#DDDDDD"
14         onPress={handlePress}
15       >
16         <Text style={styles.buttonText}>Press Me</Text>
17       </TouchableHighlight>
18     </View>
19   );
20
21   const styles = StyleSheet.create({
22     container: {
23       flex: 1,
24       justifyContent: 'center',
25       alignItems: 'center',
26     },
27     button: {
28       backgroundColor: '#F2F2F2',
29       padding: 10,
30       borderRadius: 8,
31     },
32     buttonText: {
33       fontSize: 16,
34       color: '#333333',
35     },
36   });
37
38   export default App;
```

In this example, we have a simple button implemented using the `TouchableHighlight` component. The `TouchableHighlight` wraps the text "Press Me", and the `onPress` prop is assigned to the `handlePress` callback function, which logs a message to the console when the button is pressed.

The `TouchableHighlight` component applies a highlight effect when pressed, which is controlled by the `underlayColor` prop. In this case, the underlay color is set to `#DDDDDD`, providing a visual feedback to indicate the interaction.

Props of TouchableHighlight

The `TouchableHighlight` component accepts several props that allow you to customize its behavior and appearance. Here are the commonly used props with their default values:

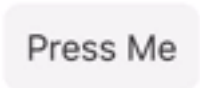
- **`onPress (function, required)`**: Specifies the callback function to be called when the component is pressed.
- **`underlayColor (string)`**: Sets the color of the background when the component is in the pressed state. The default value is 'black' on iOS and 'white' on Android.
- **`activeOpacity (number)`**: Determines the opacity of the button when it is pressed. The default value is 0.85.
- **`style (object)`**: Defines the style for the container of the `TouchableHighlight`. It allows you to specify dimensions, positioning, and other styling properties.
- **`disabled (boolean)`**: Disables the button if set to true. The default value is false.

Use Cases

The `TouchableHighlight` component is useful in various scenarios where you want to add interactivity to your app's user interface. Here are some common use cases for using `TouchableHighlight`:

- **Buttons**: Create interactive buttons that highlight when pressed, providing visual feedback to the user.
- **Navigation**: Implement clickable elements for navigation, such as menu items or navigation icons.
- **Expandable content**: Use `TouchableHighlight` to toggle the visibility of additional content or to expand and collapse sections.
- **Interactive lists**: Add touchable areas within lists to enable actions like item selection or expanding list items.

By utilizing the **`TouchableHighlight`** component in these scenarios, you can enhance the user experience by making your app more interactive and responsive to user input.

 Press Me