

BackHandler

The Backhandler API detects hardware button presses for back navigation, lets you register event listeners for the system's back action, and lets you control how your application responds. It is Android-only.

The event subscriptions are called in reverse order (i.e. the last registered subscription is called first).

- **If one subscription returns true**, then subscriptions registered earlier will not be called.
- **If no subscription returns true or none are registered**, it programmatically invokes the default back button functionality to exit the app.

Warning for modal users: If your app shows an opened `Modal`, `BackHandler` will not publish any events (see [Modal docs](#)).

Pattern

```
BackHandler.addListener('hardwareBackPress', function () {  
  /**  
   * this.onMainScreen and this.goBack are just examples,  
   * you need to use your own implementation here.  
   *  
   * Typically you would use the navigator here to go to the last state.  
   */  
  
  if (!this.onMainScreen()) {  
    this.goBack();  
    /**  
     * When true is returned the event will not be bubbled up  
     * & no other back action will execute  
     */  
    return true;  
  }  
  /**
```

```
* Returning false will let the event to bubble up & let other event listeners
* or the system's default back action to be executed.
*/
return false;
});
```

Example

The following example implements a scenario where you confirm if the user wants to exit the app:

BackHandler

^ Expo

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

const App = () => {
  useEffect(() => {
    const backAction = () => {
      Alert.alert('Hold on!', 'Are you sure you want to go back?', [
        {
          text: 'Cancel',
          onPress: () => null,
          style: 'cancel',
        },
        {text: 'YES', onPress: () => BackHandler.exitApp()},
      ]);
      return true;
    };

    const backHandler = BackHandler.addEventListener(
      'hardwareBackPress',
      backAction,
    );

    return () => backHandler.remove();
  }, []);

  return (
```

Preview



My Device

Android

`BackHandler.addEventListener` creates an event listener & returns a `NativeEventSubscription` object which should be cleared using `NativeEventSubscription.remove` method.

Usage with React Navigation

If you are using React Navigation to navigate across different screens, you can follow their guide on [Custom Android back button behaviour](#)

Backhandler hook

[React Native Hooks](#) has a nice `useBackHandler` hook which will simplify the process of setting up event listeners.

Reference

Methods

`addEventListener()`

```
static addEventListener(  
  eventName: BackPressEventName,  
  handler: () => boolean | null | undefined,  
): NativeEventSubscription;
```

`exitApp()`


```
static exitApp();
```

`removeEventListener()`

```
static removeEventListener(  
  eventName: BackPressEventName,
```

```
handler: () => boolean | null | undefined,  
);
```

Is this page useful?  

 Edit this page

Last updated on **Jun 21, 2023**