Hardware Back Button

The hardware back button is found on most Android devices. In native applications it can be used to close modals, navigate to the previous view, exit an app, and more. By default in Ionic, when the back button is pressed, the current view will be popped off the navigation stack, and the previous view will be displayed. If no previous view exists in the navigation stack, nothing will happen. This guide will show how to customize the behavior of the hardware back button.

Note

The hardware back button refers to the physical back button on an Android device and should not be confused with either the browser back button or `ion-back-button`. The information in this guide only applies to Android devices.

Hardware Back Button in Capacitor and Cordova

Note

The `@capacitor/app` package must be installed in Capacitor apps to use the hardware back button.

When running in a Capacitor or Cordova application, Ionic Framework will emit an `ionBackButton` event when a user presses the hardware back button.

When listening for the `ionBackButton` event, you can register a handler to be fired. This handler can perform actions such as quitting the app or opening a confirmation dialog. Each handler must be assigned a priority. By default, only one handler is fired per hardware back button press. The priority value is used to determine which callback should be called. This is useful because if you have a modal open, you likely would not want the modal to close *and* the app to navigate backwards when pressing the hardware back button. Only running one handler at a time allows the modal to close but still requires another press of the hardware back button to navigate backwards.

There are situations where you might want to have multiple handlers fired. Each handler callback passes in a function as a parameter that can be used to tell the framework to call the next handler.

Hardware Back Button in a Browser

When running your app in a mobile browser or as a PWA, hardware back button customization will be limited. This is because Capacitor and Cordova expose additional features that are not exposed in a normal web browser. For example, closing overlays and menus via the hardware back button are functionalities that are currently not supported when running your app in a mobile browser. These are known limitations and do not currently have straightforward solutions.

For complete hardware back button support, we recommend using Capacitor or Cordova.

Note

The `ionBackButton` event will not be emitted when running an app in a browser or as a PWA.

Basic Usage

JavaScript Angular React Vue

```
document.addEventListener('ionBackButton', (ev) => {
  ev.detail.register(10, () => {
    console.log('Handler was called!');
  });
});
```

In this example, we are registering a handler to be called when the hardware back button is pressed. We have set the priority to be 10, and we have not indicated to the framework that we want the next handler to be called. As a result, any handlers with a priority less than 10 will not be called. A handler that has a priority greater than 10 will be called first.

In the event that there are handlers with the same priority value, the handler that was registered *last* will be called. See Handlers with the Same Priorities for more information.

Calling Multiple Handlers

Each hardware back button callback has a `processNextHandler` parameter. Calling this function allows you to continue calling hardware back button handlers.

JavaScript Angular React Vue

```
document.addEventListener('ionBackButton', (ev) => {
  ev.detail.register(5, () => {
    console.log('Another handler was called!');
  });

  ev.detail.register(10, (processNextHandler) => {
    console.log('Handler was called!');

    processNextHandler();
  });
  });
```

This example shows how to indicate to Ionic Framework that you want the next handler to be fired. All callbacks are provided with a `processNextHandler` function as a parameter. Calling this will cause the next handler, if any exists, to be fired.

Handlers with the Same Priorities

Internally, Ionic Framework uses something similar to a priority queue to manage hardware back button handlers. The handler with the largest priority value will be called first. In the event that there are multiple handlers with the same priority value, the *last* handler of the same priority added to this queue will be the first handler to be called.

```
document.addEventListener('ionBackButton', (ev) => {
  // Handler A
  ev.detail.register(10, (processNextHandler) => {
    console.log('Handler A was called!');

    processNextHandler();
  });

  // Handler B
  ev.detail.register(10, (processNextHandler) => {
    console.log('Handler B was called!');

    processNextHandler();
  });
});
```

In the example above, both handlers A and B have a priority of 10. Since handler B was registered last, Ionic Framework will call handler B before it calls handler A.

Exiting the App

In some scenarios, it may be desirable to quit the app when pressing the hardware back button. This can be achieved through the use of the `ionBackButton` event combined with methods that Capacitor/Cordova provide.

JavaScript Angular React Vue

```
import { BackButtonEvent } from '@ionic/core';
import { App } from '@capacitor/app';

...

const routerEl = document.querySelector('ion-router');
document.addEventListener('ionBackButton', (ev: BackButtonEvent) => {
  ev.detail.register(-1, () => {
    const path = window.location.pathname;
    if (path === routerEl.root) {
      App.exitApp();
    }
  });
});
```

This example shows the application exiting when the user presses the hardware back button and there is nothing left in the navigation stack. It is also possible to display a confirmation dialog before quitting the app.

It is recommended to check whether or not the user is on the root page prior to exiting the application. Developers can use the `canGoBack` method on `IonRouterOutlet` in Ionic Angular and `IonRouter` in Ionic React and Ionic Vue.

Internal Framework Handlers

The table below lists all of the internal hardware back button event handlers that Ionic Framework uses. The `Propagates` column notes whether or not that particular handler tells Ionic Framework to call the next back button handler.

Handler	Priority	Propagates	Description
Overlays	100	No	Applies to overlay components `ion-action-sheet`, `ion-alert`, `ion-loading`, `ion-modal`, `ion-popover`, and `ion-picker`.
Menu	99	No	Applies to `ion-menu`.
Navigation	0	Yes	Applies to routing navigation (i.e. Angular Routing).

Edit this page