ion-popover

SHADOW

A Popover is a dialog that appears on top of the current page. It can be used for anything, but generally it is used for overflow actions that don't fit in the navigation bar.

There are two ways to use `ion-popover`: inline or via the `popoverController`. Each method comes with different considerations, so be sure to use the approach that best fits your use case.

Inline Popovers

`ion-popover` can be used by writing the component directly in your template. This reduces the number of handlers you need to wire up in order to present the popover.

When using `ion-popover` with Angular, React, or Vue, the component you pass in will be destroyed when the popover is dismissed. As this functionality is provided by the JavaScript framework, using `ion-popover` without a JavaScript framework will not destroy the component you passed in. If this is a needed functionality, we recommend using the `popoverController` instead.

When to use

Using a popover inline is useful when you do not want to explicitly wire up click events to open the popover. For example, you can use the `trigger` property to designate a button that should present the popover when clicked. You can also use the `trigger-action` property to customize whether the popover should be presented when the trigger is left clicked, right clicked, or hovered over.

If you need fine grained control over when the popover is presented and dismissed, we recommend you use the `popoverController`.

Angular

Since the component you passed in needs to be created when the popover is presented and destroyed when the popover is dismissed, we are unable to project the content using `<ng-content>` internally. Instead, we use `<ng-container>` which expects an `<ng-template>` to be passed in. As a result, when passing in your component you will need to wrap it in an `<ng-template>`:

Copy

```
<ion-popover [isOpen]="isPopoverOpen">
  <ng-template>
    <app-popover-content></app-popover-content>
  </ng-template>
</ion-popover>
```

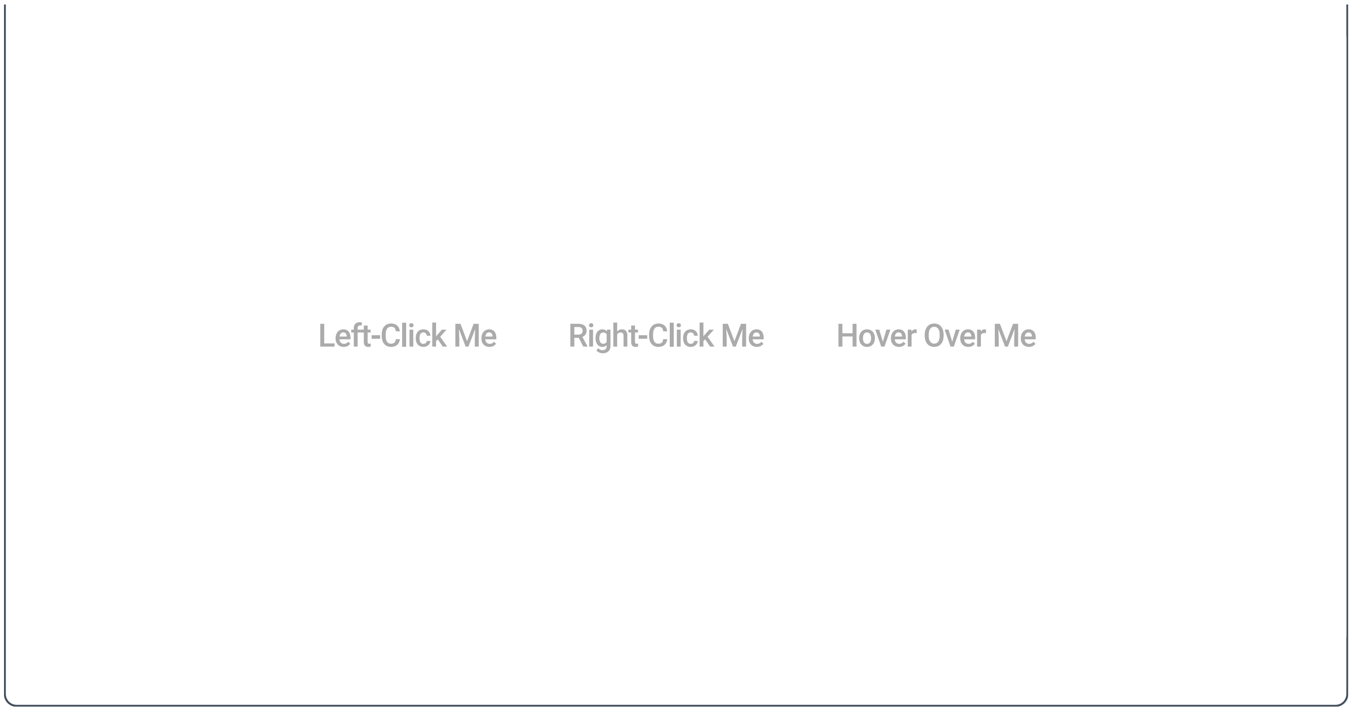
Triggers

A trigger for an inline `ion-popover` is the element that will open a popover when interacted with. The interaction behavior can be customized by setting the `trigger-action` property. Note that `trigger-action="context-menu"` will prevent your system's default context menu from opening.

Note

Triggers are not applicable when using the `popoverController` because the `ion-popover` is not created ahead of time.

Angular JavaScript React Vue | iOS MD



```
<ion-button id="click-trigger">Left-Click Me</ion-button>
<ion-popover trigger="click-trigger" triggerAction="click">
  <ng-template>
    <ion-content class="ion-padding">Hello World!</ion-content>
  </ng-template>
</ion-popover>

<ion-button id="context-menu-trigger">Right-Click Me</ion-button>
<ion-popover trigger="context-menu-trigger" triggerAction="context-menu">
  <ng-template>
    <ion-content class="ion-padding">Hello World!</ion-content>
  </ng-template>
</ion-popover>

<ion-button id="hover-trigger">Hover Over Me</ion-button>
<ion-popover trigger="hover-trigger" triggerAction="hover">
  <ng-template>
    <ion-content class="ion-padding">Hello World!</ion-content>
  </ng-template>
</ion-popover>
```

isOpen Property

Inline popovers can also be opened by setting the `isOpen` property to `true`. This method can be used if you need finer grained control over the popover than with a trigger.

`isOpen` uses a one-way data binding, meaning it will not automatically be set to `false` when the popover is dismissed. Developers should listen for the `ionPopoverDidDismiss` or `didDismiss` event and set `isOpen` to `false`. The reason for this is it prevents the internals of `ion-popover` from being tightly coupled with the state of the application. With a one way data binding, the popover only needs to concern itself with the boolean value that the reactive variable provides. With a two way data binding, the popover needs to concern itself with both the boolean value as well as the existence of the reactive variable itself. This can lead to non-deterministic behaviors and make applications harder to debug.

Angular JavaScript React Vue | iOS MD

Click Me

src/app/example.component.html TS src/app/example.component.ts

```
<ion-button (click)="presentPopover($event)">Click Me</ion-button>
<ion-popover #popover [isOpen]="isOpen" (didDismiss)="isOpen = false">
  <ng-template>
    <ion-content class="ion-padding">Hello World!</ion-content>
   </ng-template>
 </ion-popover>
```

Controller Popovers

`ion-popover` can also be presented programmatically by using the `popoverController` imported from Ionic Framework. This allows you to have complete control over when a popover is presented above and beyond the customization that inline popovers give you.

When to use

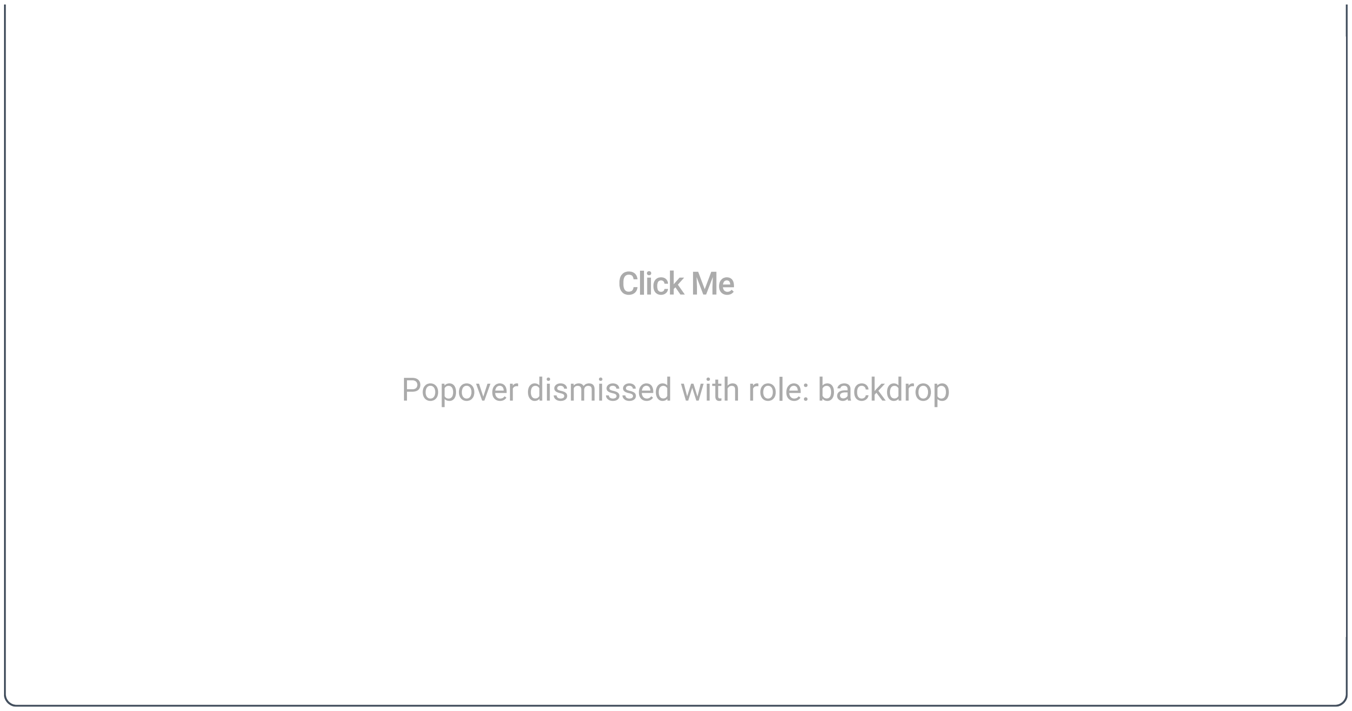
We typically recommend that you write your popovers inline as it streamlines the amount of code in your application. You should only use the `popoverController` for complex use cases where writing a popover inline is impractical. When using a controller, your popover is not created ahead of time, so properties such as `trigger` and `trigger-action` are not applicable here. In addition, nested popovers are not compatible with the controller approach because the popover is automatically added to the root of your application when the `create` method is called.

React

Instead of a controller, React has a hook called `useIonPopover` which behaves in a similar fashion. Note that `useIonPopover` requires being a descendant of `<IonApp>`. If you need to use a popover outside of an `<IonApp>`, consider using an inline popover instead.

Usage

Angular JavaScript React Vue | iOS MD



src/app/example.component.html   TS src/app/example.component.ts   src/app/popover.component.htm

```
<ion-button (click)="presentPopover($event)">Click Me</ion-button>
<p>{{ roleMsg }}</p>
```

Styling

Popovers are presented at the root of your application so they overlay your entire app. This behavior applies to both inline popovers and popovers presented from a controller. As a result, custom popover styles can not be scoped to a particular component as they will not apply to the popover. Instead, styles must be applied globally. For most developers, placing the custom styles in `global.css` is sufficient.

Note

If you are building an Ionic Angular app, the styles need to be added to a global stylesheet file.

src/app/example.component.html   src/global.css

```
<ion-button id="trigger-button">Click Me</ion-button>
<ion-popover trigger="trigger-button">
  <ng-template>
    <ion-content class="ion-padding">Hello Styled World!</ion-content>
  </ng-template>
</ion-popover>
```

Positioning

Reference

When presenting a popover, Ionic Framework needs a reference point to present the popover relative to. With `reference="event"`, the popover will be presented relative to the x-y coordinates of the pointer event that was dispatched on your trigger element. With `reference="trigger"`, the popover will be presented relative to the bounding box of your trigger element.

Side

Regardless of what you choose for your reference point, you can position a popover to the `top`, `right`, `left`, or `bottom` of your reference point by using the `side` property. You can also use the `start` or `end` values if you would like the side to switch based on LTR or RTL modes.

Alignment

The `alignment` property allows you to line up an edge of your popover with a corresponding edge on your trigger element. The exact edge that is used depends on the value of the `side` property.

Side and Alignment Demo

Angular JavaScript React Vue | iOS MD

src/app/example.component.html src/app/example.component.css src/app/example.component.ts

```
<div class="container">
  <ion-button id="top-center">Side=Top, Alignment=Center</ion-button>
  <ion-popover trigger="top-center" side="top" alignment="center">
    <ng-template>
      <ion-content class="ion-padding">Hello World!</ion-content>
    </ng-template>
  </ion-popover>
```

```
    <ion-button id="bottom-start">Side=Bottom, Alignment=Start</ion-button>
    <ion-popover trigger="bottom-start" side="bottom" alignment="start">
      <ng-template>
        <ion-content class="ion-padding">Hello World!</ion-content>
      </ng-template>
    </ion-popover>

    <ion-button id="left-start">Side=Left, Alignment=Start</ion-button>
    <ion-popover trigger="left-start" side="left" alignment="start">
      <ng-template>
        <ion-content class="ion-padding">Hello World!</ion-content>
      </ng-template>
    </ion-popover>

    <ion-button id="right-end">Side=Right, Alignment=End</ion-button>
    <ion-popover trigger="right-end" side="right" alignment="end">
      <ng-template>
        <ion-content class="ion-padding">Hello World!</ion-content>
      </ng-template>
    </ion-popover>
  </div>
```

Offsets

If you need finer grained control over the positioning of your popover you can use the `--offset-x` and `--offset-y` CSS Variables. For example, `--offset-x: 10px` will move your popover content to the right by `10px`.

Sizing

When making dropdown menus, you may want to have the width of the popover match the width of the trigger element. Doing this without knowing the trigger width ahead of time is tricky. You can set the `size` property to `'cover'` and Ionic Framework will ensure that the width of the popover matches the width of your trigger element.

If you are using the `popoverController`, you must provide an event via the `event` option and Ionic Framework will use `event.target` as the reference element. See the controller demo for an example of this pattern.

Angular JavaScript React Vue | iOS MD

```
<ion-button id="auto-trigger">Size=Auto</ion-button>
<ion-popover trigger="auto-trigger" size="auto">
  <ng-template>
    <ion-content class="ion-padding">Hello!</ion-content>
  </ng-template>
</ion-popover>

<ion-button id="cover-trigger">Size=Cover</ion-button>
<ion-popover trigger="cover-trigger" size="cover">
  <ng-template>
    <ion-content class="ion-padding">Hello!</ion-content>
  </ng-template>
</ion-popover>
```

Nested Popovers

When using `ion-popover` inline, you can nested popovers to create nested dropdown menus. When doing this, only the backdrop on the first popover will appear so that the screen does not get progressively darker as you open more popovers.

You can use the `dismissOnSelect` property to automatically close the popover when the popover content has been clicked. This behavior does not apply when clicking a trigger element for another popover.

Note

Nested popovers cannot be created when using the `popoverController` because the popover is automatically added to the root of your application when the `create` method is called.

Angular JavaScript React Vue | iOS MD

```
<ion-button id="popover-button">Open Menu</ion-button>
<ion-popover trigger="popover-button" [dismissOnSelect]="true">
  <ng-template>
    <ion-content>
      <ion-list>
        <ion-item [button]="true" [detail]="false">Option 1</ion-item>
        <ion-item [button]="true" [detail]="false">Option 2</ion-item>
        <ion-item [button]="true" id="nested-trigger">More options...</ion-item>

        <ion-popover trigger="nested-trigger" [dismissOnSelect]="true" side="end">
          <ng-template>
            <ion-content>
              <ion-list>
                <ion-item [button]="true" [detail]="false">Nested option</ion-item>
              </ion-list>
            </ion-content>
          </ng-template>
        </ion-popover>
      </ion-list>
    </ion-content>
  </ng-template>
</ion-popover>
```

Interfaces

Below you will find all of the options available to you when using the `popoverController`.

These options should be supplied when calling `popoverController.create()`.

```
interface PopoverOptions {
  component: any;
  componentProps?: { [key: string]: any };
  showBackdrop?: boolean;
  backdropDismiss?: boolean;
  translucent?: boolean;
  cssClass?: string | string[];
  event?: Event;
  animated?: boolean;
  mode?: 'ios' | 'md';
  keyboardClose?: boolean;
  id?: string;
  htmlAttributes?: { [key: string]: any };

  enterAnimation?: AnimationBuilder;
  leaveAnimation?: AnimationBuilder;

  size?: PopoverSize;
  dismissOnSelect?: boolean;
  reference?: PositionReference;
  side?: PositionSide;
  alignment?: PositionAlign;
  arrow?: boolean;
}
```

Types

Below you will find all of the custom types for `ion-popover`:

```
type PopoverSize = 'cover' | 'auto';
type TriggerAction = 'click' | 'hover' | 'context-menu';
type PositionReference = 'trigger' | 'event';
type PositionSide = 'top' | 'right' | 'bottom' | 'left' | 'start' | 'end';
type PositionAlign = 'start' | 'center' | 'end';
```

Accessibility

Keyboard Navigation

`ion-popover` has basic keyboard support for navigating between focusable elements inside of the popover. The following table details what each key does:

Key	Function
`Tab`	Moves focus to the next focusable element.
`Shift + Tab`	Moves focus to the previous focusable element.
`Esc`	Closes the popover.
`Space` or `Enter`	Clicks the focusable element.

`ion-popover` has full arrow key support for navigating between `ion-item` elements with the `button` property. The most common use case for this is as a dropdown menu in a desktop-focused application. In addition to the basic keyboard support, the following table details arrow key support for dropdown menus:

Key	Function
`ArrowUp`	Moves focus to the previous focusable element.
`ArrowDown`	Moves focus to the next focusable element.
`Home`	Moves focus to the first focusable element.
`End`	Moves focus to the last focusable element.
`ArrowLeft`	When used in a child popover, closes the popover and returns focus to the parent popover.
`Space`, `Enter`, and `ArrowRight`	When focusing a trigger element, opens the associated popover.

Performance

Mounting Inner Contents

The content of an inline `ion-popover` is unmounted when closed. If this content is expensive to render, developers can use the `keepContentsMounted` property to mount the content as soon as the popover is mounted. This can help optimize the responsiveness of your application as the inner contents will have already been mounted when the popover opens.

Angular JavaScript React Vue | iOS MD

```
<ion-button id="open-popover">Open Popover</ion-button>
<ion-popover [keepContentsMounted]="true" trigger="open-popover">
  <ng-template>
    <ion-content class="ion-padding">This content was mounted as soon as the popover
was created.</ion-content>
  </ng-template>
</ion-popover>
```

Developers should keep the following in mind when using `keepContentsMounted`:

- This feature should be used as a last resort in order to deal with existing performance problems. Try to identify and resolve performance bottlenecks before using this feature. Additionally, do not use this to anticipate performance problems.
- This feature is only needed when using a JavaScript Framework. Developers not using a framework can pass the contents to be rendered into the popover, and the contents will be rendered automatically.
- This feature only works with inline popovers. Popovers created with the `popoverController` are not created ahead of time, so the inner contents are not created either.
- Any JavaScript Framework lifecycle hooks on the inner component will run as soon as the popover is mounted, not when the popover is presented.

Properties

alignment

Description	Describes how to align the popover content with the `reference` point. Defaults to `'center'` for `ios` mode, and `'start'` for `md` mode.
Attribute	`alignment`
Type	`"center" | "end" | "start" | undefined`
Default	`undefined`

animated

Description	If `true`, the popover will animate.
Attribute	`animated`
Type	`boolean`
Default	`true`

arrow

Description	If `true`, the popover will display an arrow that points at the `reference` when running in `ios` mode. Does not apply in `md` mode.
Attribute	`arrow`
Type	`boolean`
Default	`true`

backdropDismiss

Description	If `true`, the popover will be dismissed when the backdrop is clicked.
Attribute	`backdrop-dismiss`
Type	`boolean`
Default	`true`

component

Description	The component to display inside of the popover. You only need to use this if you are not using a JavaScript framework. Otherwise, you can just slot your component inside of `ion-popover`.
Attribute	`component`
Type	`Function | HTMLElement | null | string | undefined`
Default	`undefined`

componentProps

Description	The data to pass to the popover component. You only need to use this if you are not using a JavaScript framework. Otherwise, you can just set the props directly on your component.
Attribute	`undefined`
Type	`undefined | { [key: string]: any; }`
Default	`undefined`

dismissOnSelect

Description	If `true`, the popover will be automatically dismissed when the content has been clicked.
Attribute	`dismiss-on-select`
Type	`boolean`
Default	`false`

enterAnimation

Description	Animation to use when the popover is presented.
Attribute	`undefined`
Type	`((baseEl: any, opts?: any) => Animation) | undefined`
Default	`undefined`

event

Description	The event to pass to the popover animation.
Attribute	`event`
Type	`any`
Default	`undefined`

htmlAttributes

Description	Additional attributes to pass to the popover.
Attribute	`undefined`
Type	`undefined | { [key: string]: any; }`
Default	`undefined`

isOpen

Description	If `true`, the popover will open. If `false`, the popover will close. Use this if you need finer grained control over presentation, otherwise just use the popoverController or the `trigger` property. Note: `isOpen` will not automatically be set back to `false` when the popover dismisses. You will need to do that in your code.
Attribute	`is-open`
Type	`boolean`
Default	`false`

keepContentsMounted

Description	If `true`, the component passed into `ion-popover` will automatically be mounted when the popover is created. The component will remain mounted even when the popover is dismissed. However, the component will be destroyed when the popover is destroyed. This property is not reactive and should only be used when initially creating a popover. Note: This feature only applies to inline popovers in JavaScript frameworks such as Angular, React, and Vue.

Attribute	`keep-contents-mounted`
Type	`boolean`
Default	`false`

keyboardClose

Description	If `true` , the keyboard will be automatically dismissed when the overlay is presented.
Attribute	`keyboard-close`
Type	`boolean`
Default	`true`

leaveAnimation

Description	Animation to use when the popover is dismissed.
Attribute	`undefined`
Type	`((baseEl: any, opts?: any) => Animation) | undefined`
Default	`undefined`

mode

Description	The mode determines which platform styles to use.

Attribute	`mode`
Type	`"ios" | "md"`
Default	`undefined`

reference

Description	Describes what to position the popover relative to. If `'trigger'`, the popover will be positioned relative to the trigger button. If passing in an event, this is determined via event.target. If `'event'`, the popover will be positioned relative to the x/y coordinates of the trigger action. If passing in an event, this is determined via event.clientX and event.clientY.
Attribute	`reference`
Type	`"event" | "trigger"`
Default	`'trigger'`

showBackdrop

Description	If `true`, a backdrop will be displayed behind the popover. This property controls whether or not the backdrop darkens the screen when the popover is presented. It does not control whether or not the backdrop is active or present in the DOM.
Attribute	`show-backdrop`
Type	`boolean`
Default	`true`

side

Description	Describes which side of the `reference` point to position the popover on. The `'start'` and `'end'` values are RTL-aware, and the `'left'` and `'right'` values are not.
Attribute	`side`
Type	`"bottom" | "end" | "left" | "right" | "start" | "top"`
Default	`'bottom'`

size

Description	Describes how to calculate the popover width. If `'cover'`, the popover width will match the width of the trigger. If `'auto'`, the popover width will be determined by the content in the popover.
Attribute	`size`
Type	`"auto" | "cover"`
Default	`'auto'`

translucent

Description	If `true`, the popover will be translucent. Only applies when the mode is `"ios"` and the device supports `backdrop-filter`.
Attribute	`translucent`
Type	`boolean`

Default	`false`

trigger

Description	An ID corresponding to the trigger element that causes the popover to open. Use the `trigger-action` property to customize the interaction that results in the popover opening.
Attribute	`trigger`
Type	`string | undefined`
Default	`undefined`

triggerAction

Description	Describes what kind of interaction with the trigger that should cause the popover to open. Does not apply when the `trigger` property is `undefined`. If `'click'`, the popover will be presented when the trigger is left clicked. If `'hover'`, the popover will be presented when a pointer hovers over the trigger. If `'context-menu'`, the popover will be presented when the trigger is right clicked on desktop and long pressed on mobile. This will also prevent your device's normal context menu from appearing.
Attribute	`trigger-action`
Type	`"click" | "context-menu" | "hover"`
Default	`'click'`

Events

Name	Description
`didDismiss`	Emitted after the popover has dismissed. Shorthand for ionPopoverDidDismiss.
`didPresent`	Emitted after the popover has presented. Shorthand for ionPopoverWillDismiss.
`ionPopoverDidDismiss`	Emitted after the popover has dismissed.
`ionPopoverDidPresent`	Emitted after the popover has presented.
`ionPopoverWillDismiss`	Emitted before the popover has dismissed.
`ionPopoverWillPresent`	Emitted before the popover has presented.
`willDismiss`	Emitted before the popover has dismissed. Shorthand for ionPopoverWillDismiss.
`willPresent`	Emitted before the popover has presented. Shorthand for ionPopoverWillPresent.

Methods

dismiss

Description	Dismiss the popover overlay after it has been presented.
Signature	`dismiss(data?: any, role?: string, dismissParentPopover?: boolean) => Promise<boolean>`

onDidDismiss

Description	Returns a promise that resolves when the popover did dismiss.
Signature	`onDidDismiss<T = any>() => Promise<OverlayEventDetail<T>>`

onWillDismiss

Description	Returns a promise that resolves when the popover will dismiss.
Signature	`onWillDismiss<T = any>() => Promise<OverlayEventDetail<T>>`

present

Description	Present the popover overlay after it has been created. Developers can pass a mouse, touch, or pointer event to position the popover relative to where that event was dispatched.
Signature	`present(event?: MouseEvent | TouchEvent | PointerEvent | CustomEvent) => Promise<void>`

CSS Shadow Parts

Name	Description
`arrow`	The arrow that points to the reference element. Only applies on `ios` mode.
`backdrop`	The `ion-backdrop` element.
`content`	The wrapper element for the default slot.

CSS Custom Properties

Name	Description
`--backdrop-opacity`	Opacity of the backdrop
`--background`	Background of the popover
`--box-shadow`	Box shadow of the popover
`--height`	Height of the popover
`--max-height`	Maximum height of the popover
`--max-width`	Maximum width of the popover
`--min-height`	Minimum height of the popover
`--min-width`	Minimum width of the popover
`--offset-x`	The amount to move the popover by on the x-axis
`--offset-y`	The amount to move the popover by on the y-axis
`--width`	Width of the popover

Slots

Name	Description
``	Content is placed inside of the `.popover-content` element.

Edit this page