

ion-toast

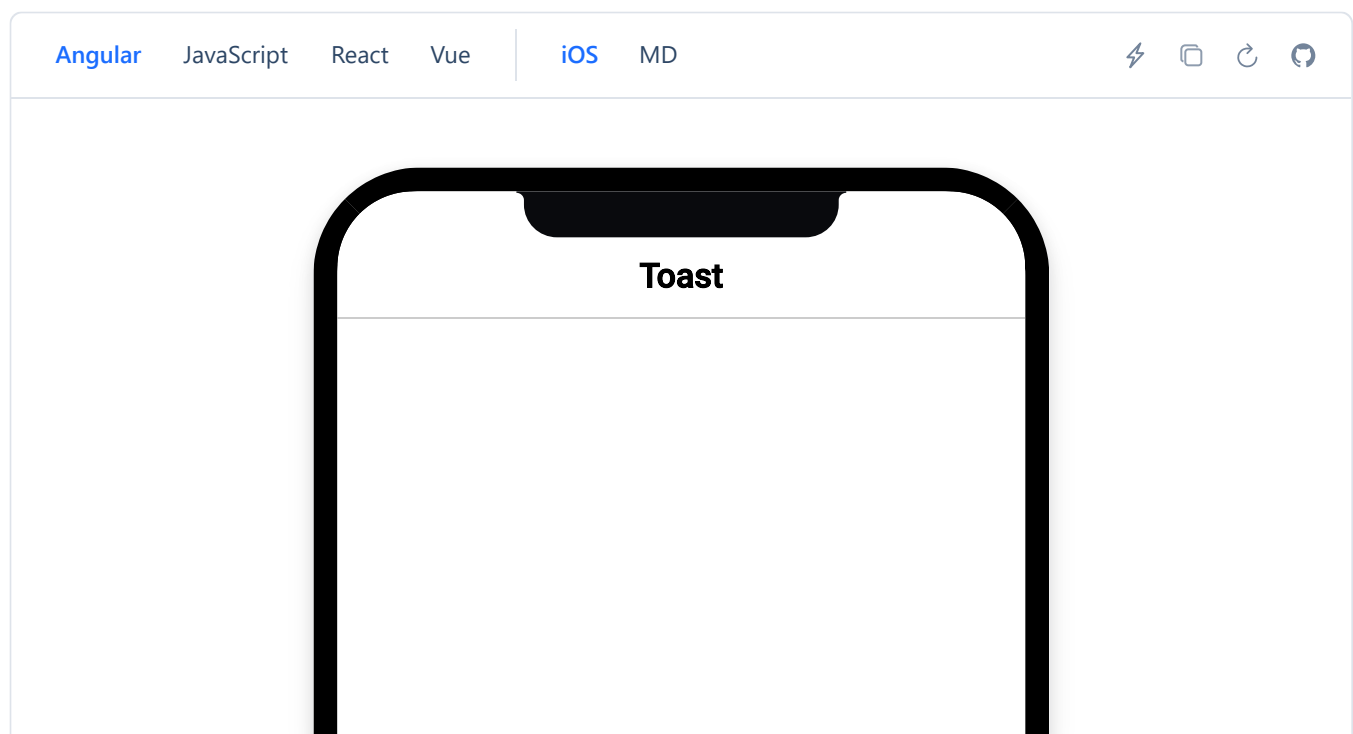
A Toast is a subtle notification commonly used in modern applications. It can be used to provide feedback about an operation or to display a system message. The toast appears on top of the app's content, and can be dismissed by the app to resume user interaction with the app.

Presenting

Positioning

Toasts can be positioned at the top, bottom or middle of the viewport. The position can be passed upon creation. The possible values are `top`, `bottom` and `middle`. If the position is not specified, the toast will be displayed at the bottom of the viewport.

Controller



Present Toast At the Top

Present Toast At the Middle

Present Toast At the Bottom

<> src/app/example.component.html

TS src/app/example.component.ts

Copy

```
<ion-button expand="block" (click)="presentToast('top')">Present Toast At the
Top</ion-button>
<ion-button expand="block" (click)="presentToast('middle')">Present Toast At the
Middle</ion-button>
<ion-button expand="block" (click)="presentToast('bottom')">Present Toast At the
Bottom</ion-button>
```

Inline

When using Ionic with React or Vue, `ion-toast` can also be placed directly in the template through use of the `isOpen` property. Note that `isOpen` must be set to `false` manually when the toast is dismissed; it will not be updated automatically.

```
import React, { useState } from 'react';
import { IonButton, IonToast } from '@ionic/react';

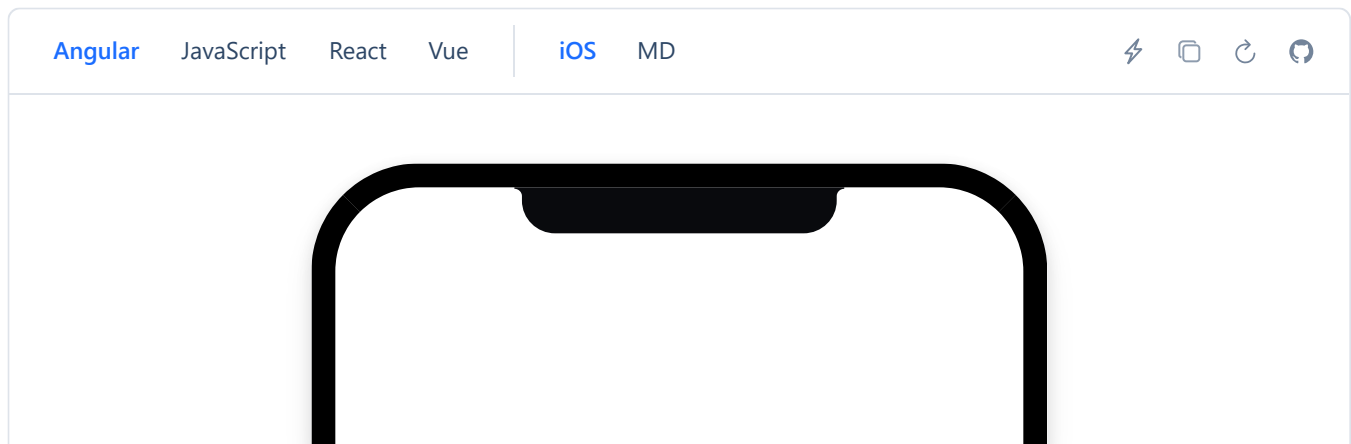
function Example() {
  const [showToast, setShowToast] = useState(false);

  return (
    <>
      <IonButton onClick={() => setShowToast(true)}>Show Toast</IonButton>
      <IonToast isOpen={showToast} onDidDismiss={() => setShowToast(false)}
message="Hello World!" duration={1500} />
    </>
  );
}
```

Dismissing

The toast can be dismissed automatically after a specific amount of time by passing the number of milliseconds to display it in the `duration` of the toast options. If a button with a role of "cancel" is added, then that button will dismiss the toast. To dismiss the toast after creation, call the `dismiss()` method on the instance.

The following example demonstrates how to use the `buttons` property to add a button that automatically dismisses the toast when clicked, as well as how to collect the `role` of the dismiss event.

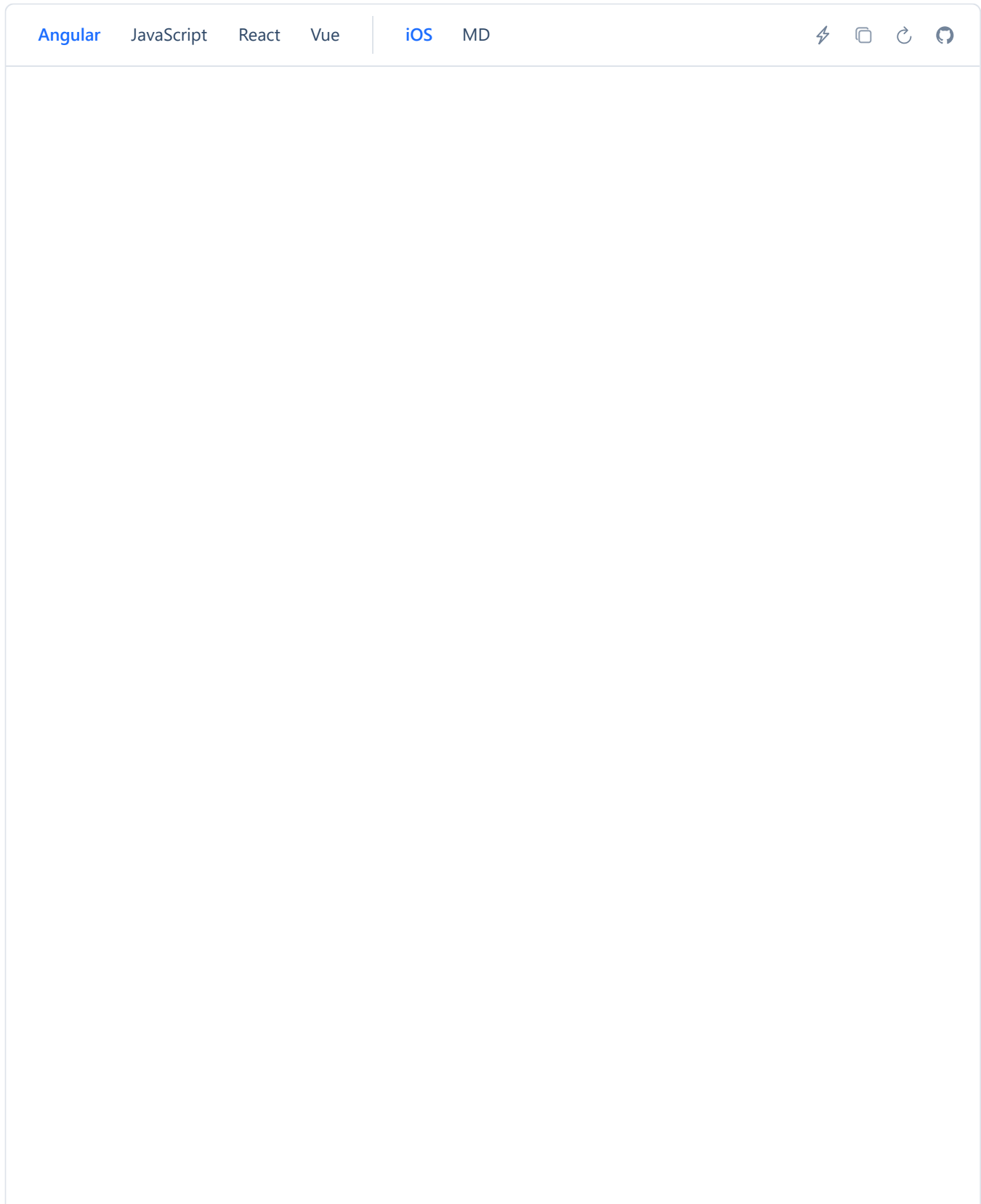


[src/app/example.component.html](#)[TS src/app/example.component.ts](#)

```
<ion-button (click)="presentToast()">Click Me</ion-button>
<p>{{ handlerMessage }}</p>
<p>{{ roleMessage }}</p>
```

Layout

Button containers within the toast can be displayed either on the same line as the message or stacked on separate lines using the `layout` property. The stacked layout should be used with buttons that have long text values. Additionally, buttons in a stacked toast layout can use a `side` value of either `start` or `end`, but not both.



[src/app/example.component.html](#)[src/app/example.component.ts](#)

```
import { Component } from '@angular/core';  
import { ToastController } from '@ionic/angular';  
import type { ToastOptions } from '@ionic/angular';
```

```
@Component({
```

```
selector: 'app-example',
templateUrl: 'example.component.html',
})
export class ExampleComponent {

  constructor(private toastController: ToastController) {}

  async presentToast(opts: ToastOptions) {
    const toast = await this.toastController.create(opts);

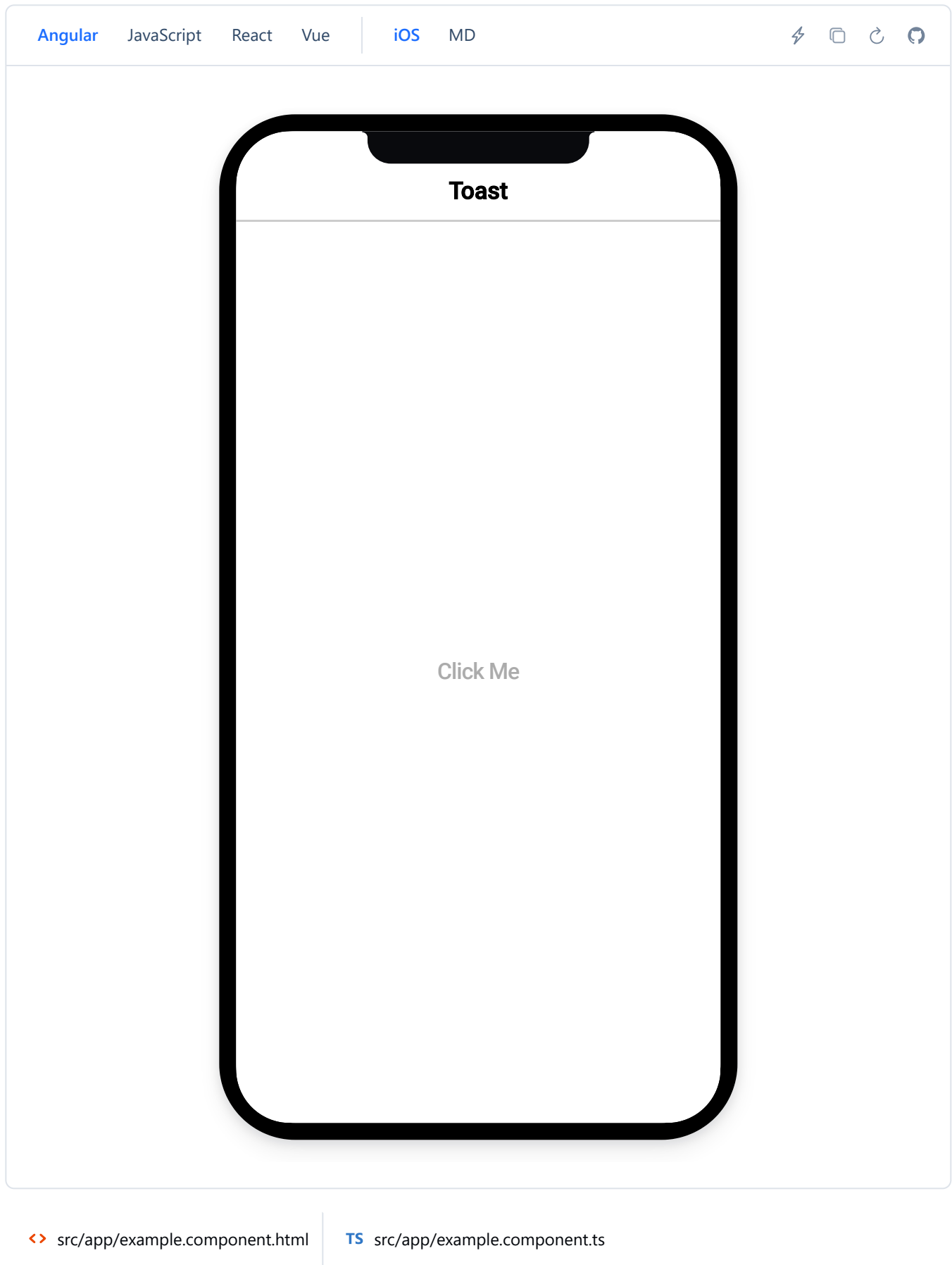
    await toast.present();
  }

  async presentBaselineToast() {
    await this.presentToast({
      duration: 3000,
      message: "This is a toast with a long message and a button that appears on the
same line.",
      buttons: [
        { text: 'Action With Long Text' }
      ]
    });
  }

  async presentStackedToast() {
    await this.presentToast({
      duration: 3000,
      message: "This is a toast with a long message and a button that appears on the
next line.",
      buttons: [
        { text: 'Action With Long Text' }
      ],
      layout: "stacked"
    });
  }
}
```

Icons

An icon can be added next to the content inside of the toast. In general, icons in toasts should be used to add additional style or context, not to grab the user's attention or elevate the priority of the toast. If you wish to convey a higher priority message to the user or guarantee a response, we recommend using an [Alert](#) instead.




```
<ion-button (click)="presentToast()">Click Me</ion-button>
```

Theming




[src/app/example.component.html](#)
[TS src/app/example.component.ts](#)
[src/global.css](#)

```
<ion-button (click)="presentToast()">Click Me</ion-button>
```

Interfaces

ToastButton

```
interface ToastButton {
  text?: string;
  icon?: string;
  side?: 'start' | 'end';
  role?: 'cancel' | string;
  cssClass?: string | string[];
  handler?: () => boolean | void | Promise<boolean | void>;
}
```

ToastOptions

```
interface ToastOptions {
  header?: string;
  message?: string | IonicSafeString;
  cssClass?: string | string[];
  duration?: number;
  buttons?: (ToastButton | string)[];
  position?: 'top' | 'bottom' | 'middle';
  translucent?: boolean;
  animated?: boolean;
  icon?: string;
  htmlAttributes?: { [key: string]: any };
}
```

```
color?: Color;  
mode?: Mode;  
keyboardClose?: boolean;  
id?: string;  
  
enterAnimation?: AnimationBuilder;  
leaveAnimation?: AnimationBuilder;  
}
```

Accessibility

Focus Management

Toasts are intended to be subtle notifications and are not intended to interrupt the user. User interaction should not be required to dismiss the toast. As a result, focus is not automatically moved to a toast when one is presented.

Screen Readers

`ion-toast` has `aria-live="polite"` and `aria-atomic="true"` set by default.

`aria-live` causes screen readers to announce the content of the toast when it is updated. However, since the attribute is set to `'polite'`, screen readers generally do not interrupt the current task. Developers can customize this behavior by using the `htmlAttributes` property to set `aria-live` to `'assertive'`. This will cause screen readers to immediately notify the user when a toast is updated, potentially interrupting any previous updates.

`aria-atomic="true"` is set to ensure that the entire toast is announced as a single unit. This is useful when dynamically updating the content of the toast as it prevents screen readers from announcing only the content that has changed.

Tips

While this is not a complete list, here are some guidelines to follow when using toasts.

- Do not require user interaction to dismiss toasts. For example, having a "Dismiss" button in the toast is fine, but the toast should also automatically dismiss on its own after a timeout period. If you need user interaction for a notification, consider using [ion-alert](#) instead.
- Avoid opening multiple toasts in quick succession. If `aria-live` is set to `'assertive'`, screen readers may interrupt the reading of the current task to announce the new toast, causing the context of the previous toast to be lost.
- For toasts with long messages, consider adjusting the `duration` property to allow users enough time to read the content of the toast.

Properties

animated

| | |
|-------------|--|
| Description | If <code>true</code> , the toast will animate. |
| Attribute | <code>animated</code> |
| Type | <code>boolean</code> |
| Default | <code>true</code> |

buttons

| | |
|-------------|------------------------------------|
| Description | An array of buttons for the toast. |
| Attribute | <code>undefined</code> |
| | |

| | |
|---------|--------------------------------------|
| Type | (string ToastButton)[] undefined |
| Default | undefined |

color

| | |
|-------------|---|
| Description | The color to use from your application's color palette. Default options are: "primary", "secondary", "tertiary", "success", "warning", "danger", "light", "medium", and "dark". For more information on colors, see theming . |
| Attribute | color |
| Type | "danger" "dark" "light" "medium" "primary" "secondary" "success" "tertiary" "warning" string & Record<never, never> undefined |
| Default | undefined |

cssClass

| | |
|-------------|--|
| Description | Additional classes to apply for custom CSS. If multiple classes are provided they should be separated by spaces. |
| Attribute | css-class |
| Type | string string[] undefined |
| Default | undefined |

duration

| | |
|-------------|---|
| Description | How many milliseconds to wait before hiding the toast. By default, it will show until <code>dismiss()</code> is called. |
| Attribute | <code>duration</code> |
| Type | <code>number</code> |
| Default | <code>config.getNumber('toastDuration', 0)</code> |

enterAnimation

| | |
|-------------|--|
| Description | Animation to use when the toast is presented. |
| Attribute | <code>undefined</code> |
| Type | <code>((baseEl: any, opts?: any) => Animation) undefined</code> |
| Default | <code>undefined</code> |

header

| | |
|-------------|----------------------------------|
| Description | Header to be shown in the toast. |
| Attribute | <code>header</code> |
| Type | <code>string undefined</code> |
| Default | <code>undefined</code> |

htmlAttributes

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

icon

| | |
|-------------|---|
| Description | The name of the icon to display, or the path to a valid SVG file. See <code>ion-icon</code> . https://ionic.io/ionicons |
| Attribute | icon |
| Type | string undefined |
| Default | undefined |

keyboardClose

| | |
|-------------|--|
| Description | If <code>true</code> , the keyboard will be automatically dismissed when the overlay is presented. |
| Attribute | keyboard-close |
| Type | boolean |
| Default | false |

layout

| | |
|-------------|---|
| Description | Defines how the message and buttons are laid out in the toast. 'baseline': The message and the buttons will appear on the same line. Message text may wrap within the message container. 'stacked': The buttons containers and message will stack on top of each other. Use this if you have long text in your buttons. |
| Attribute | layout |
| Type | "baseline" "stacked" |
| Default | 'baseline' |

leaveAnimation

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

message

| | |
|-------------|--------------------------------------|
| Description | Message to be shown in the toast. |
| Attribute | message |
| Type | IonicSafeString string undefined |
| Default | undefined |

mode

| | |
|-------------|---|
| Description | The mode determines which platform styles to use. |
| Attribute | mode |
| Type | "ios" "md" |
| Default | undefined |

position

| | |
|-------------|--|
| Description | The position of the toast on the screen. |
| Attribute | position |
| Type | "bottom" "middle" "top" |
| Default | 'bottom' |

translucent

| | |
|-------------|--|
| Description | If <code>true</code> , the toast will be translucent. Only applies when the mode is <code>"ios"</code> and the device supports backdrop-filter . |
| Attribute | translucent |
| Type | boolean |
| Default | false |

Events

| Name | Description |
|----------------------------------|---|
| <code>ionToastDidDismiss</code> | Emitted after the toast has dismissed. |
| <code>ionToastDidPresent</code> | Emitted after the toast has presented. |
| <code>ionToastWillDismiss</code> | Emitted before the toast has dismissed. |
| <code>ionToastWillPresent</code> | Emitted before the toast has presented. |

Methods

dismiss

| | |
|-------------|--|
| Description | Dismiss the toast overlay after it has been presented. |
| Signature | <code>dismiss(data?: any, role?: string) => Promise<boolean></code> |

onDidDismiss

| | |
|-------------|---|
| Description | Returns a promise that resolves when the toast did dismiss. |
| Signature | <code>onDidDismiss<T = any>() => Promise<OverlayEventDetail<T>></code> |

onWillDismiss

| | |
|-------------|--|
| Description | Returns a promise that resolves when the toast will dismiss. |
| | |

| | |
|------------------|--|
| Signature | <code>onWillDismiss<T = any>() => Promise<OverlayEventDetail<T>></code> |
|------------------|--|

present

| | |
|--------------------|--|
| Description | Present the toast overlay after it has been created. |
| Signature | <code>present() => Promise<void></code> |

CSS Shadow Parts

| Name | Description |
|-----------|---|
| button | Any button element that is displayed inside of the toast. |
| container | The element that wraps all child elements. |
| header | The header text of the toast. |
| icon | The icon that appears next to the toast content. |
| message | The body text of the toast. |

CSS Custom Properties


| Name | Description |
|--------------|---------------------------|
| --background | Background of the toast |
| --border- | Border color of the toast |

| Name | Description |
|-----------------|--|
| color | |
| --border-radius | Border radius of the toast |
| --border-style | Border style of the toast |
| --border-width | Border width of the toast |
| --box-shadow | Box shadow of the toast |
| --button-color | Color of the button text |
| --color | Color of the toast text |
| --end | Position from the right if direction is left-to-right, and from the left if direction is right-to-left |
| --height | Height of the toast |
| --max-height | Maximum height of the toast |
| --max-width | Maximum width of the toast |
| --min-height | Minimum height of the toast |
| --min-width | Minimum width of the toast |
| --start | Position from the left if direction is left-to-right, and from the right if direction is right-to-left |
| --white-space | White space of the toast message |
| | |

| Name | Description |
|---------|--------------------|
| --width | Width of the toast |

Slots

No slots available for this component.

 [Edit this page](#)