

Modal



- [Design guidelines](#)
 - [Usage](#)
 - [Principles](#)
 - [When to use](#)
 - [Anatomy](#)
 - [Modal box and scrim](#)
 - [Title](#)
 - [Buttons](#)
 - [Side-by-side buttons \(recommended\)](#)
 - [Stacked or full-width buttons](#)
 - [Behavior](#)
 - [Position](#)
 - [Scrolling](#)
 - [Dismissing modals](#)
- [Development guidelines](#)
 - [Usage](#)
 - [Props](#)
 - [aria.describedby: string](#)
 - [aria.labelledby: string](#)
 - [bodyOpenClassName: string](#)
 - [className: string](#)
 - [contentLabel: string](#)
 - [focusOnMount: boolean | 'firstElement' | 'firstContentElement'](#)
 - [headerActions](#)
 - [isDismissible: boolean](#)
 - [isFullScreen: boolean](#)
 - [size: 'small' | 'medium' | 'large' | 'fill'](#)
 - [onRequestClose: “](#)

- [overlayClassName: string](#)
 - [role: AriaRole](#)
 - [shouldCloseOnClickOutside: boolean](#)
 - [shouldCloseOnEsc: boolean](#)
 - [style: CSSProperties](#)
 - [title: string](#)
 - [__experimentalHideHeader: boolean](#)
 - [Related components](#)
-

Modals give users information and choices related to a task they're trying to accomplish. They can contain critical information, require decisions, or involve multiple tasks.

An alert modal for trashing a post

Design guidelines

Usage

A modal is a type of floating window that appears in front of content to provide critical information or ask for a decision. Modals disable all other functionality when they appear. A modal remains on screen until the user confirms it, dismisses it, or takes the required action.

While modals can be an effective way to disclose additional controls or information, they can also be a source of interruption for the user. For this reason, always question whether a modal is necessary, and work to avoid the situations in which they are required.

Principles

- **Focused.** Modals pull user attention away from the rest of the screen to focus their attention, ensuring that the modal's content is addressed.
- **Direct.** Modal text should communicate important information and be dedicated to helping the user appropriately complete a task.
- **Helpful.** Modals should appear in response to a user task or an action to offer relevant and contextual information.

When to use

Modals are used for:

- Errors that block normal operation.
- Critical information that requires a specific user task, decision, or acknowledgement.
- Contextual information that appears in response to a user task or action.

Anatomy

A modal diagram with labels

1. Container
2. Title
3. Supporting text
4. Buttons
5. Scrim
6. Close button (optional)

Modal box and scrim

A modal is a type of window. Access to the rest of the UI is disabled until the modal is addressed. All modals are interruptive by design – their purpose is to have the user focus on content, so the modal surface appears in front of all other surfaces.

To clarify that the rest of the screen is inaccessible and to focus attention on the modal, surfaces behind the modal are scrimmed — they get a temporary overlay to obscure their content and make it less prominent.

Title

A modal’s purpose is communicated through its title and button text.

All modals should have a title for accessibility reasons (the `contentLabel` prop can be used to set titles that aren’t visible).

Titles should:

- Contain a brief, clear statement or question
- Avoid apologies (“Sorry for the interruption”), alarm (“Warning!”), or ambiguity (“Are you sure?”).

A modal that asks "Trash post?"

Do

This modal title poses a specific question, concisely explains the purpose the request, and provides clear actions.

A modal that asks "Are you sure?"

Don't

This modal creates ambiguity, and therefore unease — it leaves the user unsure about how to respond, or causes them to second-guess their answer.

Buttons

Side-by-side buttons (recommended)

Side-by-side buttons display two text buttons next to one another.

A modal with two buttons next to each other

Stacked or full-width buttons

Use stacked buttons when you need to accommodate longer button text. Always place confirming actions above dismissive actions.

A modal with two buttons stacked on top of each other

Behavior

Modals appear without warning, requiring users to stop their current task. They should be used sparingly — not every choice or setting warrants this kind of abrupt interruption.

Position

Modals retain focus until dismissed or the user completes an action, like choosing a setting. They shouldn't be obscured by other elements or appear partially on screen.

Scrolling

Most modal content should avoid scrolling. Scrolling is permissible if the modal content exceeds the height of the modal (e.g. a list component with many rows). When a modal scrolls, the modal title is pinned at the top and the buttons are pinned at the bottom. This ensures that content remains visible alongside the title and buttons, even while scrolling.

Modals don't scroll with elements outside of the modal, like the background.

When viewing a scrollable list of options, the modal title and buttons remain fixed.

Dismissing modals

Modals are dismissible in three ways:

- Tapping outside of the modal
- Tapping the “Cancel” button
- Tapping the “Close” icon button, or pressing the `esc` key

If the user’s ability to dismiss a modal is disabled, they must choose a modal action to proceed.

Development guidelines

The modal is used to create an accessible modal over an application.

Note: The API for this modal has been mimicked to resemble [react-modal](#).

Usage

The following example shows you how to properly implement a modal. For the modal to properly work it’s important you implement the close logic for the modal properly.

```
import { useState } from 'react';
import { Button, Modal } from '@wordpress/components';

const MyModal = () => {
  const [ isOpen, setOpen ] = useState( false );
  const openModal = () => setOpen( true );
  const closeModal = () => setOpen( false );

  return (
    <>
      <Button variant="secondary" onClick={ openModal }>
        Open Modal
      </Button>
      { isOpen && (
        <Modal title="This is my modal" onRequestClose={ closeModal }>
          <Button variant="secondary" onClick={ closeModal }>
            My custom close button
          </Button>
        </Modal>
      ) }
    </>
  );
};
```

```
};
```

Props

The set of props accepted by the component will be specified below.
Props not included in this set will be applied to the input elements.

aria.describedby: string

If this property is added, it will be added to the modal content `div` as `aria-describedby`.

- Required: No

aria.labelledby: string

If this property is added, it will be added to the modal content `div` as `aria-labelledby`.

Use this when you are rendering the title yourself within the modal's content area instead of using the `title` prop. This ensures the title is usable by assistive technology.

Titles are required for accessibility reasons, see `contentLabel` and `title` for other ways to provide a title.

- Required: No
- Default: if the `title` prop is provided, this will default to the id of the element that renders `title`

bodyOpenClassName: string

Class name added to the body element when the modal is open.

If you use a custom class name, its styles must set `overflow: hidden` to preserve the Modal's scroll lock.

- Required: No
- Default: `modal-open`

className: string

If this property is added, it will add an additional class name to the modal content `div`.

- Required: No

contentLabel: string

If this property is added, it will be added to the modal content `div` as `aria-label`.

Titles are required for accessibility reasons, see `aria.labelledby` and `title` for other ways to provide a title.

- Required: No

`focusOnMount`: `boolean` | `'firstElement'` | `'firstContentElement'`

If this property is true, it will focus the first tabbable element rendered in the modal.

If this property is false, focus will not be transferred and it is the responsibility of the consumer to ensure accessible focus management.

If set to `firstElement` focus will be placed on the first tabbable element anywhere within the Modal.

If set to `firstContentElement`, focus will be placed on the first tabbable element within the Modal's **content** (i.e. children).

- Required: No
- Default: `true`

headerActions

An optional React node intended to contain additional actions or other elements related to the modal, for example, buttons. Content is rendered in the top right corner of the modal and to the left of the close button, if visible.

- Required: No
- Default: `null`

`isDismissible`: `boolean`

If this property is set to false, the modal will not display a close icon and cannot be dismissed.

- Required: No
- Default: `true`

`isFullScreen`: `boolean`

This property when set to `true` will render a full screen modal.

- Required: No
- Default: `false`

`size`: `'small'` | `'medium'` | `'large'` | `'fill'`

If this property is added it will cause the modal to render at a preset width, or expand to fill the screen. This prop will be ignored if `isFullScreen` is set to `true`.

- Required: No

Note: `Modal`'s width can also be controlled by adjusting the width of the modal's contents via CSS.

`onRequestClose`: “

This function is called to indicate that the modal should be closed.

- Required: Yes

`overlayClassName`: `string`

If this property is added, it will add an additional class name to the modal overlay `div`.

- Required: No

`role`: `AriaRole`

If this property is added, it will override the default role of the modal.

- Required: No
- Default: `dialog`

`shouldCloseOnClickOutside`: `boolean`

If this property is added, it will determine whether the modal requests to close when a mouse click occurs outside of the modal content.

- Required: No
- Default: `true`

`shouldCloseOnEsc`: `boolean`

If this property is added, it will determine whether the modal requests to close when the escape key is pressed.

- Required: No
- Default: `true`

`style`: `CSSProperties`

If this property is added, it will be added to the modal frame `div`.

- Required: No

`title: string`

This property is used as the modal header's title.

Titles are required for accessibility reasons, see `aria.labelledby` and `contentLabel` for other ways to provide a title.

- Required: No

`__experimentalHideHeader: boolean`

When set to `true`, the Modal's header (including the icon, title and close button) will not be rendered. A `title` is then applied to the dialog as its `aria-label`, so the dialog keeps an accessible name.

Warning: This property is still experimental. "Experimental" means this is an early implementation subject to drastic and breaking changes.

- Required: No
- Default: `false`

Related components

- To notify a user with a message of medium importance, use `Notice`.

👤 #2

👤 2026-10-31 18:13:56 UTC

👤 2026-10-31 18:13:57 UTC