






Unstyled modal

The `Modal` component lets you create dialogs, popovers, lightboxes, and other elements that force the user to take action before continuing.

The `Modal` component renders its children in front of a backdrop. This lets you create an element that your users must interact with before continuing in the parent application.

Features:

-  Manages modal stacking when more than one is needed
-  Creates a backdrop to disable interaction with the rest of the app
-  Disables page scrolling while open
-  Manages focus correctly between the modal and its parent app
-  Adds the appropriate ARIA roles automatically

```
{{"component": "modules/components/ComponentLinkHeader.js", "design": false}}
```

Note: the term "modal" is sometimes used interchangeably with "dialog," but this is incorrect. A dialog may be modal or nonmodal (modeless).

A modal [blocks interaction with the rest of the application](#), forcing the user to take action. As such, it should be used sparingly—only when the app requires user input before it can continue.

`Modal` is a lower-level construct that is used in the following Material UI components:


- [Dialog](#)
- [Drawer](#)
- [Menu](#)
- [Popover](#)

Basic modal

```
import ModalUnstyled from '@mui/base/ModalUnstyled';
```

```
{{"demo": "ModalUnstyled.js", "defaultCodeOpen": false }}
```

Nested modal

 **Note:** though it is possible to create nested modals—for example, a select modal within a dialog—stacking more than two at a time is discouraged. This is because each successive modal blocks interaction with all elements behind it, making prior states inaccessible and overly complicated for the user to navigate through.

```
{{"demo": "NestedModal.js", "defaultCodeOpen": false}}
```

Transitions

You can animate the open and close states of a modal with a transition component, as long as that component fulfills the following requirements:

- Is a direct child descendent of the modal
- Has an `in` prop—this corresponds to the open/close state

- Calls the `onEnter` callback prop when the enter transition starts
- Calls the `onExited` callback prop when the exit transition is completed

The `onEnter` and `onExited` callbacks tell the modal to unmount the child content when closed and fully transitioned.

`Modal` has built-in support for [react-transition-group](#):

```
{{"demo": "TransitionsModal.js", "defaultCodeOpen": false}}
```

You can also use [react-spring](#) with the `Modal`, but it will require additional custom configuration:

```
{{"demo": "SpringModal.js", "defaultCodeOpen": false}}
```

Performance

The `Modal` component's content is unmounted when it is not open. This means that it will need to be re-mounted each time it is opened. If you are rendering expensive component trees inside your modal, and you want to optimize for interaction responsiveness, you can change this default behavior by enabling the `keepMounted` prop.

You can also use the `keepMounted` prop if you want to make the content of the modal available to search engines (even when the modal is closed).

```
<Modal keepMounted />
```

```
{{"demo": "KeepMountedModal.js", "defaultCodeOpen": false}}
```

You can use the `MuiModal-hidden` class to hide the modal when it is not open.

As with any performance optimization, the `keepMounted` prop won't necessarily solve all of your problems.

Explore other possible bottlenecks in performance where you could make more considerable improvements before implementing this prop.

Server-side modal

React [doesn't support](#) the [createPortal\(\)](#) API on the server.

In order to display a modal rendered on the server, you need to disable the portal feature with the

`disablePortal` prop:

```
{{"demo": "ServerModal.js", "defaultCodeOpen": false}}
```

Limitations

Focus trap

`Modal` moves the focus back to the body of the component if the focus tries to escape it.

This is done for accessibility purposes, but it can potentially create issues for your users.

If the user needs to interact with another part of the page—for example, to interact with a chatbot window while a modal is open in the parent app—you can disable the default behavior:

```
<Modal disableEnforceFocus />
```

Accessibility

(WAI-ARIA: https://www.w3.org/TR/wai-aria-practices/#dialog_modal)

- All interactive elements must have an [accessible name](#). Use the `aria-labelledby="id..."` to give your `Modal` component an accessible name. You can also use `aria-describedby="id..."` to provide a description of the `Modal` :

```
<Modal aria-labelledby="modal-title" aria-describedby="modal-description">
  <h2 id="modal-title">My Title</h2>
  <p id="modal-description">My Description</p>
</Modal>
```

- Follow the [WAI-ARIA authoring practices](#) to help you set the initial focus on the most relevant element based on the content of the modal.

⚠ Note: a modal window can sit on top of either the parent application, or another modal window. All windows under the topmost modal are **inert**, meaning the user cannot interact with them. This can lead to [conflicting behaviors](#).