Modal

The modal component provides a solid foundation for creating dialogs, popovers, lightboxes, or whatever else.

The component renders its `children` node in front of a backdrop component. The `Modal` offers important features:

- 💄 Manages modal stacking when one-at-a-time just isn't enough.
- 🔐 Creates a backdrop, for disabling interaction below the modal.
- 🔐 It disables scrolling of the page content while open.
- ♿️ It properly manages focus; moving to the modal content, and keeping it there until the modal is closed.
- ♿️ Adds the appropriate ARIA roles automatically.

View as Markdown Feedback Bundle size Source WAI-ARIA

The term "modal" is sometimes used to mean "dialog", but this is a misnomer. A modal window describes parts of a UI. An element is considered modal if it blocks interaction with the rest of the application.

If you are creating a modal dialog, you probably want to use the Dialog component rather than directly using Modal. Modal is a lower-level construct that is leveraged by the following components:

- Dialog
- Drawer
- Menu
- Popover

Basic modal

OPEN MODAL

Edit in Chat JS TS Collapse code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
import Modal from '@mui/material/Modal';

const style = {
  position: 'absolute',
  top: '50%',
  left: '50%',
  transform: 'translate(-50%, -50%)',
  width: 400,
  bgcolor: 'background.paper',
  border: '2px solid #000',
  boxShadow: 24,
  p: 4,
};

export default function BasicModal() {
  const [open, setOpen] = React.useState(false);
  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);

  return (
    <div>
      <Button onClick={handleOpen}>Open modal</Button>
      <Modal
        open={open}
        onClose={handleClose}
        aria-labelledby="modal-modal-title"
        aria-describedby="modal-modal-description"
      >
        <Box sx={style}>
          <Typography id="modal-modal-title" variant="h6" component="h2">
            Text in a modal
          </Typography>
          <Typography id="modal-modal-description" sx={{ mt: 2 }}>
            Duis mollis, est non commodo luctus, nisi erat porttitor ligula.
```

Notice that you can disable the outline (often blue or gold) with the `outline: 0` CSS property.

Nested modal

Modals can be nested, for example a select within a dialog, but stacking of more than two modals, or any two modals with a backdrop is discouraged.

OPEN MODAL

Edit in Chat JS TS Collapse code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Modal from '@mui/material/Modal';
import Button from '@mui/material/Button';

const style = {
  position: 'absolute',
  top: '50%',
  left: '50%',
  transform: 'translate(-50%, -50%)',
  width: 400,
  bgcolor: 'background.paper',
  border: '2px solid #000',
  boxShadow: 24,
  pt: 2,
  px: 4,
  pb: 3,
};

function ChildModal() {
  const [open, setOpen] = React.useState(false);
  const handleOpen = () => {
    setOpen(true);
  };
  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Button onClick={handleOpen}>Open Child Modal</Button>
      <Modal
        open={open}
        onClose={handleClose}
        aria-labelledby="child-modal-title"
        aria-describedby="child-modal-description"
      >
        <Box sx={{ ...style, width: 200 }}>
```

Transitions

The open/close state of the modal can be animated with a transition component. This component should respect the following conditions:

- Be a direct child descendent of the modal.
- Have an `in` prop. This corresponds to the open/close state.
- Call the `onEnter` callback prop when the enter transition starts.
- Call the `onExited` callback prop when the exit transition is completed. These two callbacks allow the modal to unmount the child content when closed and fully transitioned.

Modal has built-in support for react-transition-group.

OPEN MODAL

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Backdrop from '@mui/material/Backdrop';
import Box from '@mui/material/Box';
import Modal from '@mui/material/Modal';
import Fade from '@mui/material/Fade';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';

const style = {
  position: 'absolute',
  top: '50%',
  left: '50%',
  transform: 'translate(-50%, -50%)',
  width: 400,
  bgcolor: 'background.paper',
  border: '2px solid #000',
  boxShadow: 24,
  p: 4,
};

export default function TransitionsModal() {
  const [open, setOpen] = React.useState(false);
  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);

  return (
    <div>
      <Button onClick={handleOpen}>Open modal</Button>
      <Modal
        aria-labelledby="transition-modal-title"
        aria-describedby="transition-modal-description"
        open={open}
        onClose={handleClose}
        closeAfterTransition
        slots={{ backdrop: Backdrop }}
        slotProps={{
          backdrop: {
```

```
          timeout: 500,
```

Alternatively, you can use react-spring.

OPEN MODAL

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Backdrop from '@mui/material/Backdrop';
import Box from '@mui/material/Box';
import Modal from '@mui/material/Modal';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
import { useSpring, animated } from '@react-spring/web';

interface FadeProps {
  children: React.ReactElement<any>;
  in?: boolean;
  onClick?: any;
  onEnter?: (node: HTMLElement, isAppearing: boolean) => void;
  onExited?: (node: HTMLElement, isAppearing: boolean) => void;
  ownerState?: any;
}

const Fade = React.forwardRef<HTMLDivElement, FadeProps>(function Fade(props, ref) {
  const {
    children,
    in: open,
    onClick,
    onEnter,
    onExited,
    ownerState,
    ...other
  } = props;
  const style = useSpring({
    from: { opacity: 0 },
    to: { opacity: open ? 1 : 0 },
    onStart: () => {
      if (open && onEnter) {
        onEnter(null as any, true);
      }
    },
    onRest: () => {
      if (!open && onExited) {
        onExited(null as any, true);
```

Performance

The content of modal is unmounted when closed. If you need to make the content available to search engines or render expensive component trees inside your modal while optimizing for interaction responsiveness it might be a good idea to change this default behavior by enabling the `keepMounted` prop:

```
<Modal keepMounted />
```

Copy

OPEN MODAL

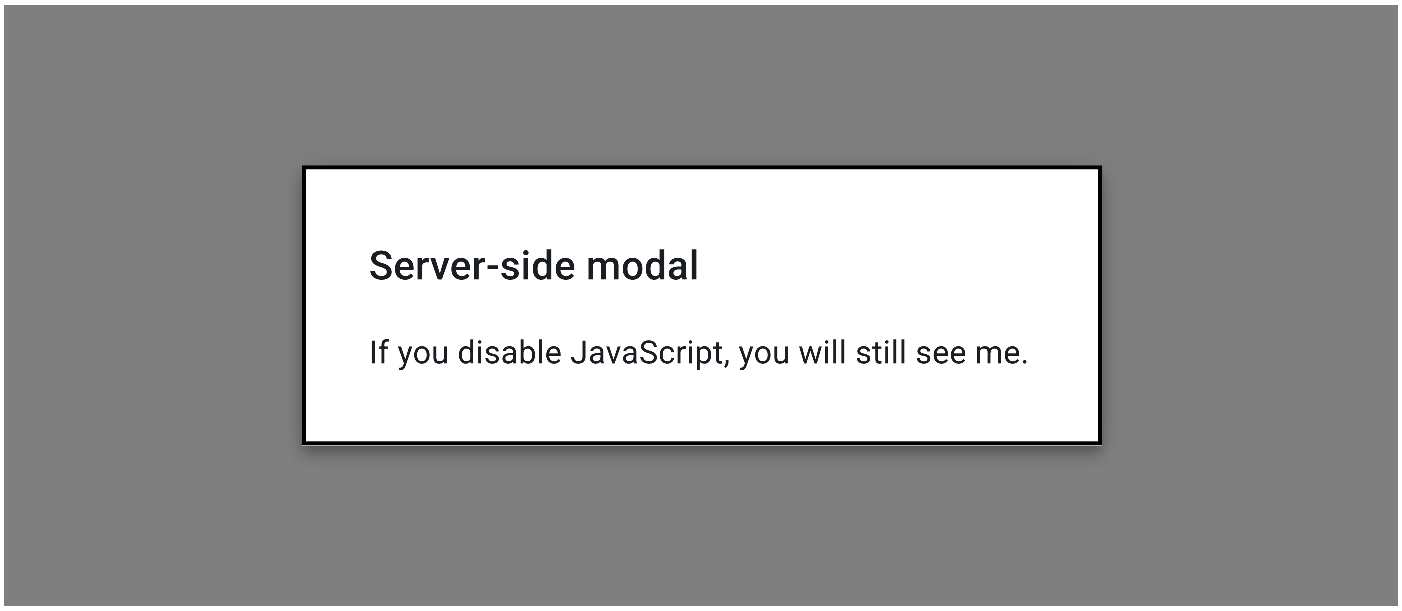
Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Modal from '@mui/material/Modal';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';

const style = {
  position: 'absolute',
  top: '50%',
  left: '50%',
  transform: 'translate(-50%, -50%)',
  width: 400,
  bgcolor: 'background.paper',
  border: '2px solid #000',
  boxShadow: 24,
  p: 4,
};

export default function KeepMountedModal() {
  const [open, setOpen] = React.useState(false);
  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);

  return (
    <div>
      <Button onClick={handleOpen}>Open modal</Button>
      <Modal
        keepMounted
        open={open}
        onClose={handleClose}
        aria-labelledby="keep-mounted-modal-title"
        aria-describedby="keep-mounted-modal-description"
      >
        <Box sx={style}>
          <Typography id="keep-mounted-modal-title" variant="h6" component="h2">
            Text in a modal
          </Typography>
          <Typography id="keep-mounted-modal-description" sx={{ mt: 2 }}>
```

As with any performance optimization, this is not a silver bullet. Be sure to identify bottlenecks first, and then try out these optimization strategies.

Server-side modal

React doesn't support the `createPortal()` API on the server. In order to display the modal, you need to disable the portal feature with the `disablePortal` prop:


Server-side modal

If you disable JavaScript, you will still see me.

```tsx
import * as React from 'react';
import Modal from '@mui/material/Modal';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';

export default function ServerModal() {
  const rootRef = React.useRef<HTMLDivElement>(null);

  return (
    <Box
      sx={{
        height: 300,
        flexGrow: 1,
        minWidth: 300,
        transform: 'translateZ(0)',
      }}
      ref={rootRef}
    >
      <Modal
        disablePortal
        disableEnforceFocus
        disableAutoFocus
        open
        aria-labelledby="server-modal-title"
        aria-describedby="server-modal-description"
        sx={{
          display: 'flex',
          p: 1,
          alignItems: 'center',
          justifyContent: 'center',
        }}
        container={() => rootRef.current!}
      >
        <Box
          sx={(theme) => ({
            position: 'relative',
            width: 400,
```

Limitations

Focus trap

The modal moves the focus back to the body of the component if the focus tries to escape it.

This is done for accessibility purposes. However, it might create issues. In the event the users need to interact with another part of the page, for example with a chatbot window, you can disable the behavior:

```
<Modal disableEnforceFocus />
```

Accessibility

(WAI-ARIA: https://www.w3.org/WAI/ARIA/apg/patterns/dialog-modal/)

- Be sure to add `aria-labelledby="id..."`, referencing the modal title, to the `Modal`. Additionally, you may give a description of your modal with the `aria-describedby="id..."` prop on 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>
  ```

- The WAI-ARIA Authoring Practices can help you set the initial focus on the most relevant element, based on your modal content.
- Keep in mind that a "modal window" overlays on either the primary window or another modal window. Windows under a modal are **inert**. That is, users cannot interact with content outside an active modal window. This might create conflicting behaviors.

API

See the documentation below for a complete reference to all of the props and classes available to the components mentioned here.

- `<Modal />`