Dialog

Dialogs inform users about a task and can contain critical information, require decisions, or involve multiple tasks.

A Dialog is a type of modal window that appears in front of app content to provide critical information or ask for a decision. Dialogs disable all app functionality when they appear, and remain on screen until confirmed, dismissed, or a required action has been taken.

Dialogs are purposefully interruptive, so they should be used sparingly.

View as Markdown Feedback Bundle size Source WAI-ARIA Material Design Figma Sketch

Introduction

Dialogs are implemented using a collection of related components:

- Dialog: the parent component that renders the modal.
- Dialog Title: a wrapper used for the title of a Dialog.
- Dialog Actions: an optional container for a Dialog's Buttons.
- Dialog Content: an optional container for displaying the Dialog's content.
- Dialog Content Text: a wrapper for text inside of `<DialogContent />`.
- Slide: optional Transition used to slide the Dialog in from the edge of the screen.

Selected: user02@gmail.com

OPEN SIMPLE DIALOG

Edit in Chat JS TS Collapse code

```
import * as React from 'react';
import Button from '@mui/material/Button';
import Avatar from '@mui/material/Avatar';
import List from '@mui/material/List';
import ListItem from '@mui/material/ListItem';
import ListItemAvatar from '@mui/material/ListItemAvatar';
import ListItemButton from '@mui/material/ListItemButton';
import ListItemText from '@mui/material/ListItemText';
import DialogTitle from '@mui/material/DialogTitle';
import Dialog from '@mui/material/Dialog';
import PersonIcon from '@mui/icons-material/Person';
import AddIcon from '@mui/icons-material/Add';
import Typography from '@mui/material/Typography';
import { blue } from '@mui/material/colors';

const emails = ['username@gmail.com', 'user02@gmail.com'];

export interface SimpleDialogProps {
  open: boolean;
  selectedValue: string;
  onClose: (value: string) => void;
}

function SimpleDialog(props: SimpleDialogProps) {
  const { onClose, selectedValue, open } = props;

  const handleClose = () => {
    onClose(selectedValue);
  };

  const handleListItemClick = (value: string) => {
    onClose(value);
  };

  return (
    <Dialog onClose={handleClose} open={open}>
```

Basics

```
import Dialog from '@mui/material/Dialog';
import DialogTitle from '@mui/material/DialogTitle';
```

Alerts

Alerts are urgent interruptions, requiring acknowledgement, that inform the user about a situation.

Most alerts don't need titles. They summarize a decision in a sentence or two by either:

- Asking a question (for example "Delete this conversation?")
- Making a statement related to the action buttons

Use title bar alerts only for high-risk situations, such as the potential loss of connectivity. Users should be able to understand the choices based on the title and button text alone.

If a title is required:

- Use a clear question or statement with an explanation in the content area, such as "Erase USB storage?".
- Avoid apologies, ambiguity, or questions, such as "Warning!" or "Are you sure?"

OPEN ALERT DIALOG

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Button from '@mui/material/Button';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';

export default function AlertDialog() {
  const [open, setOpen] = React.useState(false);

  const handleClickOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Button variant="outlined" onClick={handleClickOpen}>
        Open alert dialog
      </Button>
      <Dialog
        open={open}
        onClose={handleClose}
        aria-labelledby="alert-dialog-title"
        aria-describedby="alert-dialog-description"
      >
        <DialogTitle id="alert-dialog-title">
          {"Use Google's location service?"}
        </DialogTitle>
        <DialogContent>
          <DialogContentText id="alert-dialog-description">
            Let Google help apps determine location. This means sending anonymous
            location data to Google, even when no apps are running.
          </DialogContentText>
```

Transitions

You can also swap out the transition, the next example uses `Slide`.

SLIDE IN ALERT DIALOG

Edit in Chat JS TS Hide code

```tsx
import * as React from 'react';
import Button from '@mui/material/Button';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';
import Slide from '@mui/material/Slide';
import { TransitionProps } from '@mui/material/transitions';

const Transition = React.forwardRef(function Transition(
  props: TransitionProps & {
    children: React.ReactElement<any, any>;
  },
  ref: React.Ref<unknown>,
) {
  return <Slide direction="up" ref={ref} {...props} />;
});

export default function AlertDialogSlide() {
  const [open, setOpen] = React.useState(false);

  const handleClickOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Button variant="outlined" onClick={handleClickOpen}>
        Slide in alert dialog
      </Button>
      <Dialog
        open={open}
        slots={{
```

Form dialogs

Form dialogs allow users to fill out form fields within a dialog. For example, if your site prompts for potential subscribers to fill in their email address, they can fill out the email field and touch 'Submit'.

OPEN FORM DIALOG

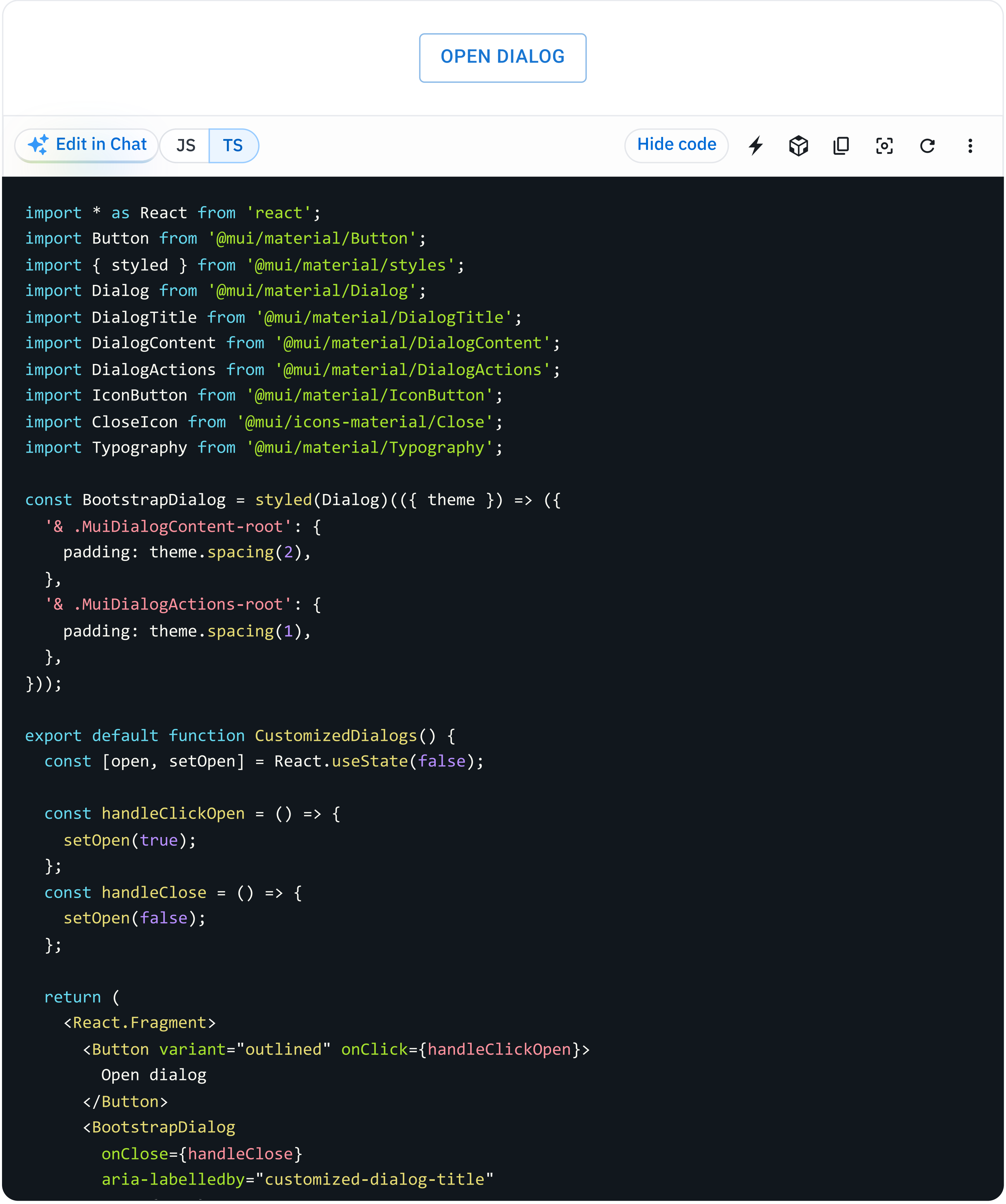
Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Button from '@mui/material/Button';
import TextField from '@mui/material/TextField';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';

export default function FormDialog() {
  const [open, setOpen] = React.useState(false);

  const handleClickOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  const handleSubmit = (event: React.FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    const formData = new FormData(event.currentTarget);
    const formJson = Object.fromEntries((formData as any).entries());
    const email = formJson.email;
    console.log(email);
    handleClose();
  };

  return (
    <React.Fragment>
      <Button variant="outlined" onClick={handleClickOpen}>
        Open form dialog
      </Button>
      <Dialog open={open} onClose={handleClose}>
        <DialogTitle>Subscribe</DialogTitle>
        <DialogContent>
          <DialogContentText>
```

Customization

Here is an example of customizing the component. You can learn more about this in the overrides documentation page.

The dialog has a close button added to aid usability.



```
import * as React from 'react';
import Button from '@mui/material/Button';
import { styled } from '@mui/material/styles';
import Dialog from '@mui/material/Dialog';
import DialogTitle from '@mui/material/DialogTitle';
import DialogContent from '@mui/material/DialogContent';
import DialogActions from '@mui/material/DialogActions';
import IconButton from '@mui/material/IconButton';
import CloseIcon from '@mui/icons-material/Close';
import Typography from '@mui/material/Typography';

const BootstrapDialog = styled(Dialog)(({ theme }) => ({
  '& .MuiDialogContent-root': {
    padding: theme.spacing(2),
  },
  '& .MuiDialogActions-root': {
    padding: theme.spacing(1),
  },
}));

export default function CustomizedDialogs() {
  const [open, setOpen] = React.useState(false);

  const handleClickOpen = () => {
    setOpen(true);
  };
  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Button variant="outlined" onClick={handleClickOpen}>
        Open dialog
      </Button>
      <BootstrapDialog
        onClose={handleClose}
        aria-labelledby="customized-dialog-title"
```

Full-screen dialogs

OPEN FULL-SCREEN DIALOG

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Button from '@mui/material/Button';
import Dialog from '@mui/material/Dialog';
import ListItemText from '@mui/material/ListItemText';
import ListItemButton from '@mui/material/ListItemButton';
import List from '@mui/material/List';
import Divider from '@mui/material/Divider';
import AppBar from '@mui/material/AppBar';
import Toolbar from '@mui/material/Toolbar';
import IconButton from '@mui/material/IconButton';
import Typography from '@mui/material/Typography';
import CloseIcon from '@mui/icons-material/Close';
import Slide from '@mui/material/Slide';
import { TransitionProps } from '@mui/material/transitions';

const Transition = React.forwardRef(function Transition(
  props: TransitionProps & {
    children: React.ReactElement<unknown>;
  },
  ref: React.Ref<unknown>,
) {
  return <Slide direction="up" ref={ref} {...props} />;
});

export default function FullScreenDialog() {
  const [open, setOpen] = React.useState(false);

  const handleClickOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Button variant="outlined" onClick={handleClickOpen}>
```

Optional sizes

You can set a dialog maximum width by using the `maxWidth` enumerable in combination with the `fullWidth` boolean. When the `fullWidth` prop is true, the dialog will adapt based on the `maxWidth` value.

OPEN MAX-WIDTH DIALOG

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Button from '@mui/material/Button';
import Dialog, { DialogProps } from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';
import FormControl from '@mui/material/FormControl';
import FormControlLabel from '@mui/material/FormControlLabel';
import InputLabel from '@mui/material/InputLabel';
import MenuItem from '@mui/material/MenuItem';
import Select, { SelectChangeEvent } from '@mui/material/Select';
import Switch from '@mui/material/Switch';

export default function MaxWidthDialog() {
  const [open, setOpen] = React.useState(false);
  const [fullWidth, setFullWidth] = React.useState(true);
  const [maxWidth, setMaxWidth] = React.useState<DialogProps['maxWidth']>('sm');

  const handleClickOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  const handleMaxWidthChange = (event: SelectChangeEvent<typeof maxWidth>) => {
    setMaxWidth(
      // @ts-expect-error autofill of arbitrary value is not handled.
      event.target.value,
    );
  };

  const handleFullWidthChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setFullWidth(event.target.checked);
  };
```

Responsive full-screen

You may make a dialog responsively full screen using `useMediaQuery`.

```
import useMediaQuery from '@mui/material/useMediaQuery';

function MyComponent() {
  const theme = useTheme();
  const fullScreen = useMediaQuery(theme.breakpoints.down('md'));

  return <Dialog fullScreen={fullScreen} />;
}
```

OPEN RESPONSIVE DIALOG

Edit in Chat JS TS Hide code

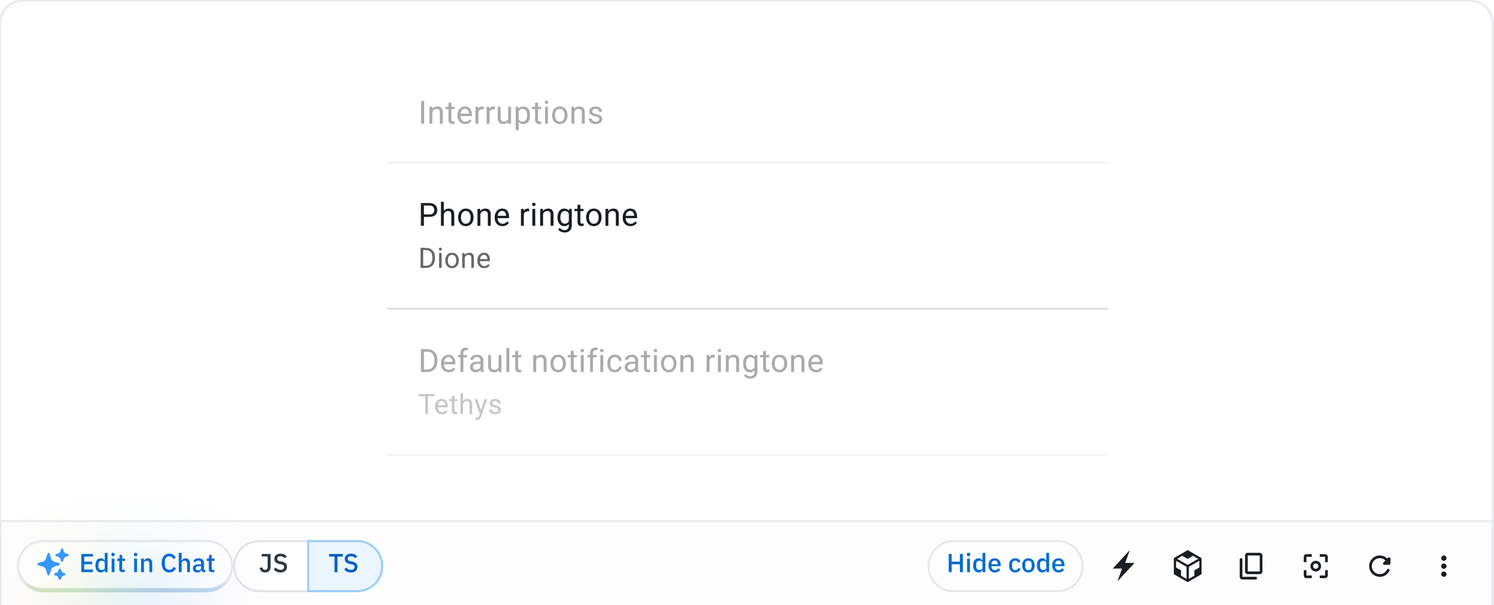
```
import * as React from 'react';
import Button from '@mui/material/Button';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';
import useMediaQuery from '@mui/material/useMediaQuery';
import { useTheme } from '@mui/material/styles';

export default function ResponsiveDialog() {
  const [open, setOpen] = React.useState(false);
  const theme = useTheme();
  const fullScreen = useMediaQuery(theme.breakpoints.down('md'));

  const handleClickOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Button variant="outlined" onClick={handleClickOpen}>
        Open responsive dialog
      </Button>
      <Dialog
        fullScreen={fullScreen}
        open={open}
        onClose={handleClose}
        aria-labelledby="responsive-dialog-title"
      >
        <DialogTitle id="responsive-dialog-title">
```

```
          {"Use Google's location service?"}
        </DialogTitle>
        <DialogContent>
```

Confirmation dialogs

Confirmation dialogs require users to explicitly confirm their choice before an option is committed. For example, users can listen to multiple ringtones but only make a final selection upon touching "OK".

Touching "Cancel" in a confirmation dialog, cancels the action, discards any changes, and closes the dialog.



Interruptions

Phone ringtone
Dione

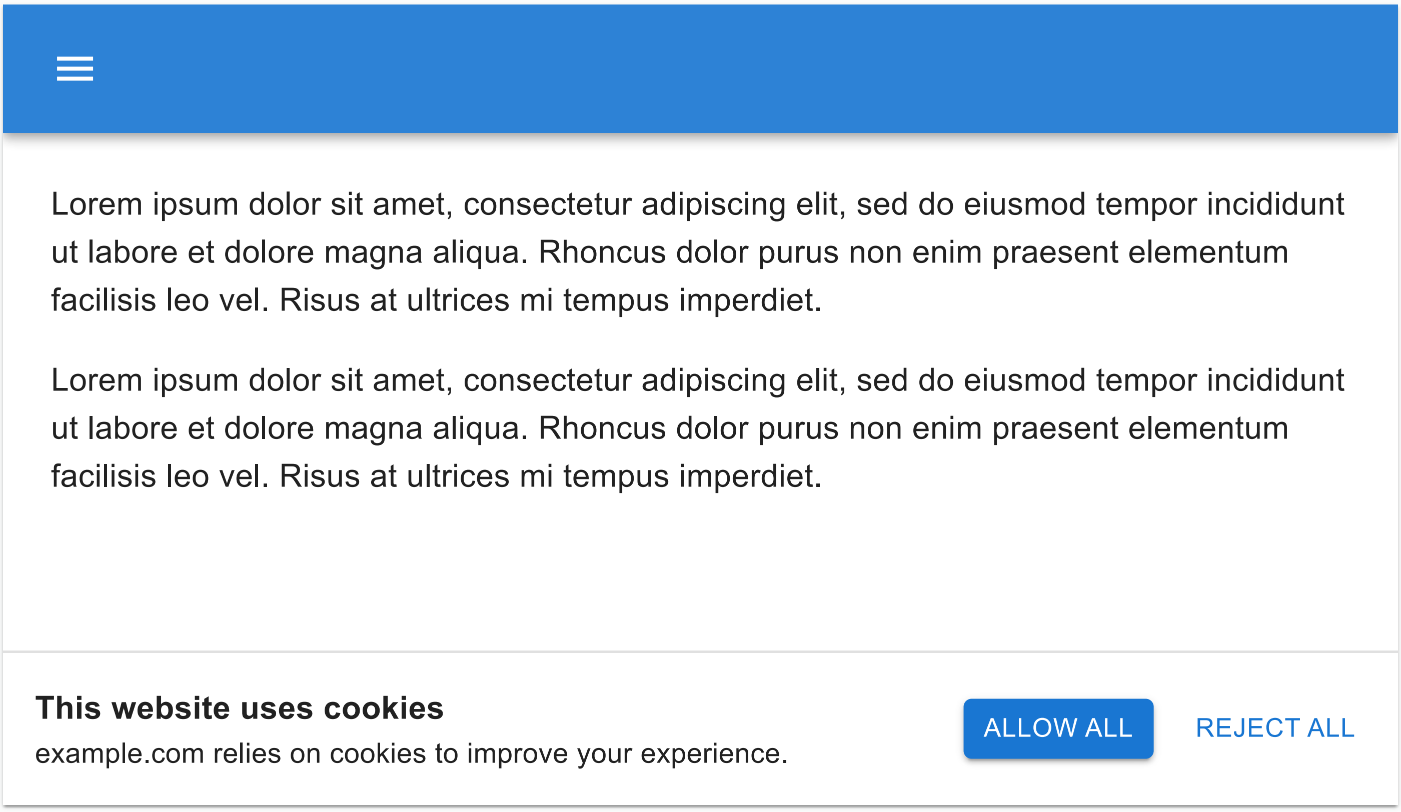
Default notification ringtone
Tethys

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Button from '@mui/material/Button';
import List from '@mui/material/List';
import ListItemButton from '@mui/material/ListItemButton';
import ListItemText from '@mui/material/ListItemText';
import DialogTitle from '@mui/material/DialogTitle';
import DialogContent from '@mui/material/DialogContent';
import DialogActions from '@mui/material/DialogActions';
import Dialog from '@mui/material/Dialog';
import RadioGroup from '@mui/material/RadioGroup';
import Radio from '@mui/material/Radio';
import FormControlLabel from '@mui/material/FormControlLabel';

const options = [
  'None',
  'Atria',
  'Callisto',
  'Dione',
  'Ganymede',
  'Hangouts Call',
  'Luna',
  'Oberon',
  'Phobos',
  'Pyxis',
  'Sedna',
  'Titania',
  'Triton',
  'Umbriel',
];

export interface ConfirmationDialogRawProps {
  id: string;
  keepMounted: boolean;
  value: string;
  open: boolean;
  onClose: (value?: string) => void;
```

Non-modal dialog

Dialogs can also be non-modal, meaning they don't interrupt user interaction behind it. Visit [the Nielsen Norman Group article](#) for more in-depth guidance about modal vs. non-modal dialog usage.

The demo below shows a persistent cookie banner, a common non-modal dialog use case.



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Rhoncus dolor purus non enim praesent elementum facilisis leo vel. Risus at ultrices mi tempus imperdiet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Rhoncus dolor purus non enim praesent elementum facilisis leo vel. Risus at ultrices mi tempus imperdiet.

This website uses cookies

example.com relies on cookies to improve your experience.

ALLOW ALL REJECT ALL

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Stack from '@mui/material/Stack';
import TrapFocus from '@mui/material/Unstable_TrapFocus';
import CssBaseline from '@mui/material/CssBaseline';
import AppBar from '@mui/material/AppBar';
import Toolbar from '@mui/material/Toolbar';
import Container from '@mui/material/Container';
import IconButton from '@mui/material/IconButton';
import MenuIcon from '@mui/icons-material/Menu';
import Paper from '@mui/material/Paper';
import Fade from '@mui/material/Fade';
import Button from '@mui/material/Button';
import Box from '@mui/material/Box';
import Typography from '@mui/material/Typography';

export default function CookiesBanner() {
  const [bannerOpen, setBannerOpen] = React.useState(true);

  const closeBanner = () => {
    setBannerOpen(false);
  };

  return (
    <React.Fragment>
      <CssBaseline />
      <AppBar position="fixed" component="nav">
        <Toolbar>
          <IconButton size="large" edge="start" color="inherit" aria-label="menu">
            <MenuIcon />
          </IconButton>
        </Toolbar>
      </AppBar>
      <Container component="main" sx={{ pt: 3 }}>
        <Toolbar />
        <Typography sx={{ marginBottom: 2 }}>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
          tempor incididunt ut labore et dolore magna aliqua. Rhoncus dolor purus non
          enim praesent elementum facilisis leo vel. Risus at ultrices mi tempus
```

Draggable dialog

You can create a draggable dialog by using [react-draggable](). To do so, you can pass the imported `Draggable` component as the `PaperComponent` of the `Dialog` component. This will make the entire dialog draggable.

OPEN DRAGGABLE DIALOG

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Button from '@mui/material/Button';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';
import Paper, { PaperProps } from '@mui/material/Paper';
import Draggable from 'react-draggable';

function PaperComponent(props: PaperProps) {
  const nodeRef = React.useRef<HTMLDivElement>(null);
  return (
    <Draggable
      nodeRef={nodeRef as React.RefObject<HTMLDivElement>}
      handle="#draggable-dialog-title"
      cancel={'[class*="MuiDialogContent-root"]'}
    >
      <Paper {...props} ref={nodeRef} />
    </Draggable>
  );
}

export default function DraggableDialog() {
  const [open, setOpen] = React.useState(false);

  const handleClickOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  return (
    <React.Fragment>
      <Button variant="outlined" onClick={handleClickOpen}>
        Open draggable dialog
```

Scrolling long content

When dialogs become too long for the user's viewport or device, they scroll.

- `scroll=paper` the content of the dialog scrolls within the paper element.
- `scroll=body` the content of the dialog scrolls within the body element.

Try the demo below to see what we mean:

SCROLL=PAPER SCROLL=BODY

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Button from '@mui/material/Button';
import Dialog, { DialogProps } from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';

export default function ScrollDialog() {
  const [open, setOpen] = React.useState(false);
  const [scroll, setScroll] = React.useState<DialogProps['scroll']>('paper');

  const handleClickOpen = (scrollType: DialogProps['scroll']) => () => {
    setOpen(true);
    setScroll(scrollType);
  };

  const handleClose = () => {
    setOpen(false);
  };

  const descriptionElementRef = React.useRef<HTMLElement>(null);
  React.useEffect(() => {
    if (open) {
      const { current: descriptionElement } = descriptionElementRef;
      if (descriptionElement !== null) {
        descriptionElement.focus();
      }
    }
  }, [open]);

  return (
    <React.Fragment>
      <Button onClick={handleClickOpen('paper')}>scroll=paper</Button>
      <Button onClick={handleClickOpen('body')}>scroll=body</Button>
      <Dialog
```

Performance

Follow the Modal performance section.

Limitations

Follow the Modal limitations section.

Supplementary projects

For more advanced use cases you might be able to take advantage of:

material-ui-confirm

Star 368 downloads 298k/month

The package `material-ui-confirm` provides dialogs for confirming user actions without writing boilerplate code.

Accessibility

Follow the Modal accessibility section.

API

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

- `<Dialog />`
- `<DialogActions />`
- `<DialogContent />`
- `<DialogContentText />`
- `<DialogTitle />`
- `<Slide />`

Edit this page

Was this page helpful?

‹ Backdrop

Progress ›