Snackbar

Snackbars (also known as toasts) are used for brief notifications of processes that have been or will be performed.

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

Introduction

The Snackbar component appears temporarily and floats above the UI to provide users with (non-critical) updates on an app's processes. The demo below, inspired by Google Keep, shows a basic Snackbar with a text element and two actions:

OPEN SNACKBAR

Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Button from '@mui/material/Button';
import Snackbar, { SnackbarCloseReason } from '@mui/material/Snackbar';
import IconButton from '@mui/material/IconButton';
import CloseIcon from '@mui/icons-material/Close';

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

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

  const handleClose = (
    event: React.SyntheticEvent | Event,
    reason?: SnackbarCloseReason,
  ) => {
    if (reason === 'clickaway') {
```

```
    return;
  }

  setOpen(false);
};

const action = (
  <React.Fragment>
    <Button color="secondary" size="small" onClick={handleClose}>
      UNDO
    </Button>
    <IconButton
      size="small"
      aria-label="close"
      color="inherit"
      onClick={handleClose}
    >
      <CloseIcon fontSize="small" />
    </IconButton>
  </React.Fragment>
```

Usage

Snackbars differ from Alerts in that Snackbars have a fixed position and a high z-index, so they're intended to break out of the document flow; Alerts, on the other hand, are usually part of the flow—except when they're used as children of a Snackbar.

Snackbars also differ from Dialogs in that Snackbars are not intended to convey *critical* information or block the user from interacting with the rest of the app; Dialogs, by contrast, require input from the user in order to be dismissed.

Basics

Import

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

Copy

Position

Use the `anchorOrigin` prop to control the Snackbar's position on the screen.

TOP-CENTER

TOP-LEFT TOP-RIGHT

BOTTOM-LEFT BOTTOM-RIGHT

BOTTOM-CENTER

Edit in Chat | JS | TS | Collapse code

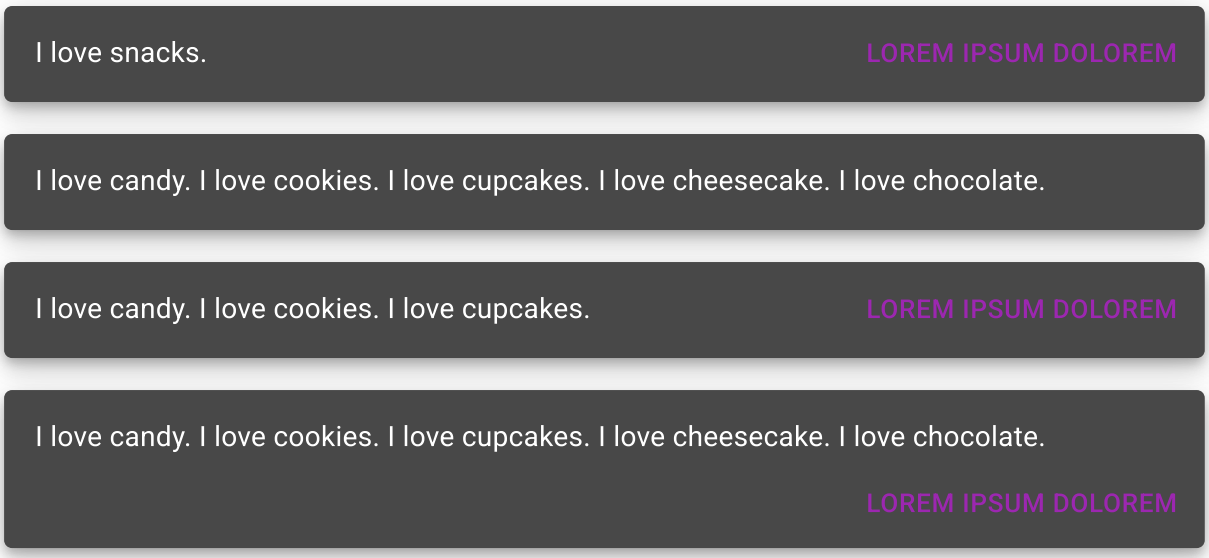
```tsx
import * as React from 'react';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import Button from '@mui/material/Button';
import Snackbar, { SnackbarOrigin } from '@mui/material/Snackbar';

interface State extends SnackbarOrigin {
  open: boolean;
}

export default function PositionedSnackbar() {
  const [state, setState] = React.useState<State>({
    open: false,
    vertical: 'top',
    horizontal: 'center',
  });
  const { vertical, horizontal, open } = state;

  const handleClick = (newState: SnackbarOrigin) => () => {
    setState({ ...newState, open: true });
  };

  const handleClose = () => {
    setState({ ...state, open: false });
  };

  const buttons = (
    <React.Fragment>
      <Box sx={{ display: 'flex', justifyContent: 'center' }}>
        <Button onClick={handleClick({ vertical: 'top', horizontal: 'center' })}>
          Top-Center
        </Button>
      </Box>
      <Grid container sx={{ justifyContent: 'center' }}>
        <Grid size={6}>
          <Button onClick={handleClick({ vertical: 'top', horizontal: 'left' })}>
            Top-Left
          </Button>
```

Content

```tsx
import SnackbarContent from '@mui/material/SnackbarContent';
```

Copy

Use the Snackbar Content component to add text and actions to the Snackbar.



Edit in Chat | JS | TS | Hide code

```tsx
import Button from '@mui/material/Button';
import Stack from '@mui/material/Stack';
import SnackbarContent from '@mui/material/SnackbarContent';

const action = (
  <Button color="secondary" size="small">
    lorem ipsum dolorem
  </Button>
);

export default function LongTextSnackbar() {
  return (
    <Stack spacing={2} sx={{ maxWidth: 600 }}>
      <SnackbarContent message="I love snacks." action={action} />
      <SnackbarContent
        message={
          'I love candy. I love cookies. I love cupcakes. \
          I love cheesecake. I love chocolate.'
        }
      />
      <SnackbarContent
        message="I love candy. I love cookies. I love cupcakes."
        action={action}
      />
      <SnackbarContent
        message={
          'I love candy. I love cookies. I love cupcakes. \
          I love cheesecake. I love chocolate.'
        }
        action={action}
      />
    </Stack>
  );
}
```

Automatic dismiss

Use the `autoHideDuration` prop to automatically trigger the Snackbar's `onClose` function after a set period of time (in milliseconds).

Make sure to provide sufficient time for the user to process the information displayed on it.

OPEN SNACKBAR

Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Button from '@mui/material/Button';
import Snackbar, { SnackbarCloseReason } from '@mui/material/Snackbar';

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

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

  const handleClose = (
    event: React.SyntheticEvent | Event,
    reason?: SnackbarCloseReason,
  ) => {
    if (reason === 'clickaway') {
      return;
    }

    setOpen(false);
  };

  return (
    <div>
      <Button onClick={handleClick}>Open Snackbar</Button>
      <Snackbar
        open={open}
        autoHideDuration={5000}
        onClose={handleClose}
        message="This Snackbar will be dismissed in 5 seconds."
      />
    </div>
  );
}
```

Transitions

You can use the `TransitionComponent` prop to change the transition of the Snackbar from Grow (the default) to others such as Slide.

GROW TRANSITION FADE TRANSITION SLIDE TRANSITION

Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Button from '@mui/material/Button';
import Snackbar from '@mui/material/Snackbar';
import Fade from '@mui/material/Fade';
import Slide, { SlideProps } from '@mui/material/Slide';
import Grow, { GrowProps } from '@mui/material/Grow';
import { TransitionProps } from '@mui/material/transitions';

function SlideTransition(props: SlideProps) {
  return <Slide {...props} direction="up" />;
}

function GrowTransition(props: GrowProps) {
  return <Grow {...props} />;
}

export default function TransitionsSnackbar() {
  const [state, setState] = React.useState<{
    open: boolean;
    Transition: React.ComponentType<
      TransitionProps & {
        children: React.ReactElement<any, any>;
      }
    >;
  }>({
    open: false,
    Transition: Fade,
  });

  const handleClick =
    (
      Transition: React.ComponentType<
        TransitionProps & {
          children: React.ReactElement<any, any>;
        }
      >,
    ) =>
    () => {
```

Customization

Preventing default click away event

If you would like to prevent the default onClickAway behavior, you can set the event's `defaultMuiPrevented` property to `true`:

```
<Snackbar
  slotProps={{
    clickAwayListener: {
      onClickAway: (event) => {
        // Prevent's default 'onClickAway' behavior.
        event.defaultMuiPrevented = true;
      },
    },
  }}
/>
```

Copy

Use with Alerts

Use an Alert inside a Snackbar for messages that communicate a certain severity.

OPEN SNACKBAR

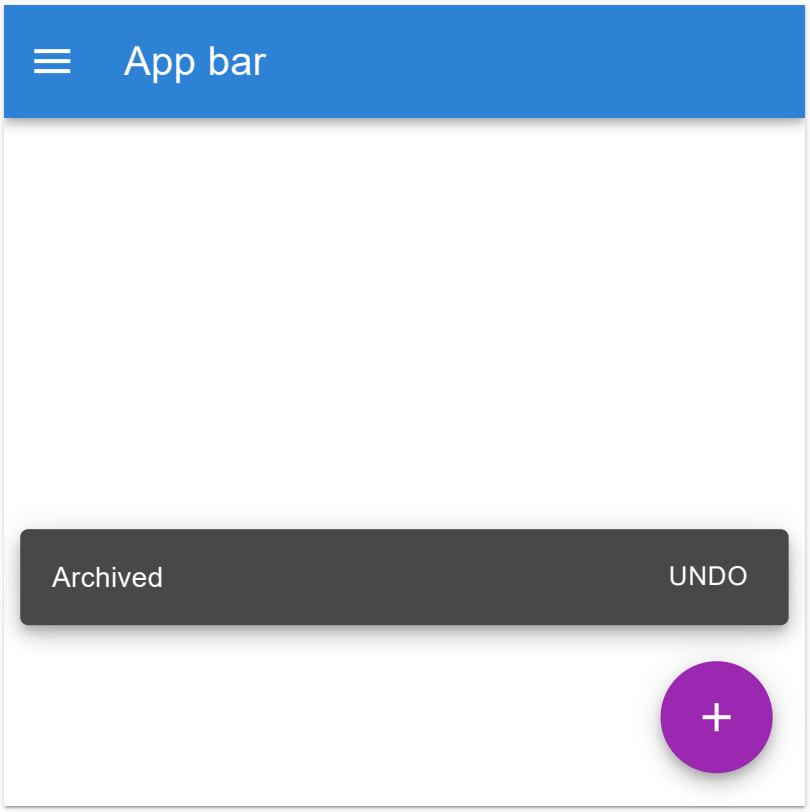
Edit in Chat | JS | TS | Collapse code

```
import * as React from 'react';
import Button from '@mui/material/Button';
import Snackbar, { SnackbarCloseReason } from '@mui/material/Snackbar';
import Alert from '@mui/material/Alert';

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

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

  const handleClose = (
    event?: React.SyntheticEvent | Event,
    reason?: SnackbarCloseReason,
  ) => {
    if (reason === 'clickaway') {
      return;
    }

    setOpen(false);
  };

  return (
    <div>
      <Button onClick={handleClick}>Open Snackbar</Button>
      <Snackbar open={open} autoHideDuration={6000} onClose={handleClose}>
        <Alert
```

```
          onClose={handleClose}
          severity="success"
          variant="filled"
          sx={{ width: '100%' }}
        >
          This is a success Alert inside a Snackbar!
        </Alert>
      </Snackbar>
    </div>
  );
}
```

Use with Floating Action Buttons

If you're using a Floating Action Button on mobile, Material Design recommends positioning snackbars directly above it, as shown in the demo below:



Edit in Chat | JS | TS | Hide code

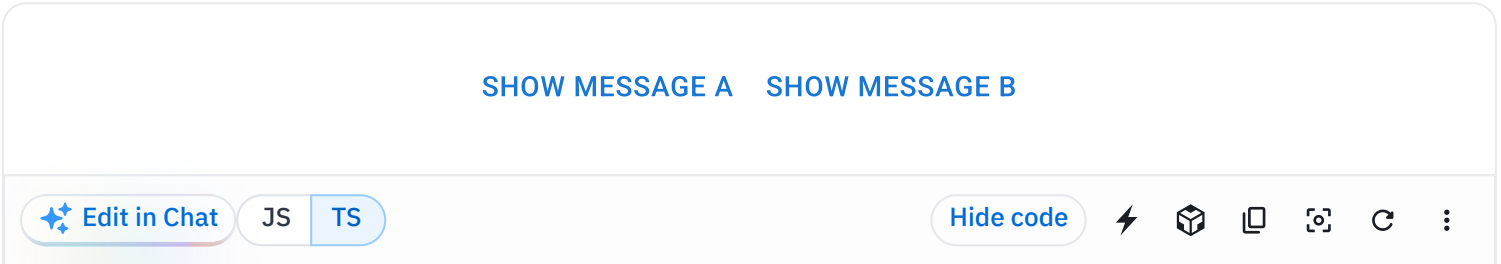
```
import * as React from 'react';
import AppBar from '@mui/material/AppBar';
import CssBaseline from '@mui/material/CssBaseline';
import GlobalStyles from '@mui/material/GlobalStyles';
import Toolbar from '@mui/material/Toolbar';
import IconButton from '@mui/material/IconButton';
import MenuIcon from '@mui/icons-material/Menu';
import Typography from '@mui/material/Typography';
import Button from '@mui/material/Button';
import Fab from '@mui/material/Fab';
import AddIcon from '@mui/icons-material/Add';
import Snackbar from '@mui/material/Snackbar';
```

```tsx
export default function FabIntegrationSnackbar() {
  return (
    <React.Fragment>
      <CssBaseline />
      <GlobalStyles
        styles={(theme) => ({
          body: { backgroundColor: theme.palette.background.paper },
        })}
      />
      <div>
        <AppBar position="static" color="primary">
          <Toolbar>
            <IconButton
              edge="start"
              sx={{ mr: 2 }}
              color="inherit"
              aria-label="menu"
            >
              <MenuIcon />
            </IconButton>
            <Typography variant="h6" color="inherit" component="div">
              App bar
            </Typography>
          </Toolbar>
        </AppBar>
```

Common examples

Consecutive Snackbars

This demo shows how to display multiple Snackbars without stacking them by using a consecutive animation.



Edit in Chat | JS | TS | Hide code

```tsx
import * as React from 'react';
import Button from '@mui/material/Button';
import Snackbar, { SnackbarCloseReason } from '@mui/material/Snackbar';
import IconButton from '@mui/material/IconButton';
import CloseIcon from '@mui/icons-material/Close';

export interface SnackbarMessage {
  message: string;
  key: number;
}
```

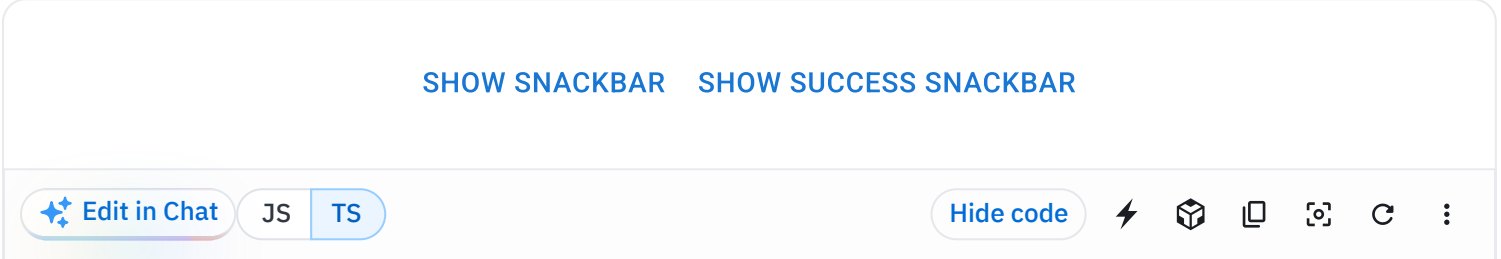
```tsx
export default function ConsecutiveSnackbars() {
  const [snackPack, setSnackPack] = React.useState<readonly SnackbarMessage[]>([]);
  const [open, setOpen] = React.useState(false);
  const [messageInfo, setMessageInfo] = React.useState<SnackbarMessage | undefined>(
    undefined,
  );

  React.useEffect(() => {
    if (snackPack.length && !messageInfo) {
      // Set a new snack when we don't have an active one
      setMessageInfo({ ...snackPack[0] });
      setSnackPack((prev) => prev.slice(1));
      setOpen(true);
    } else if (snackPack.length && messageInfo && open) {
      // Close an active snack when a new one is added
      setOpen(false);
    }
  }, [snackPack, messageInfo, open]);

  const handleClick = (message: string) => () => {
    setSnackPack((prev) => [...prev, { message, key: new Date().getTime() }]);
  };

  const handleClose = (
    event: React.SyntheticEvent | Event,
    reason?: SnackbarCloseReason,
  ) => {
```

Supplementary components

notistack

Star 4.1k downloads 3.8M/month

With an imperative API, notistack lets you vertically stack multiple Snackbars without having to handle their open and close states. Even though this is discouraged in the Material Design guidelines, it is still a common pattern.



Edit in Chat JS TS Hide code

```tsx
import * as React from 'react';
import Button from '@mui/material/Button';
import { SnackbarProvider, VariantType, useSnackbar } from 'notistack';

function MyApp() {
  const { enqueueSnackbar } = useSnackbar();
```

```
  const handleClick = () => {
    enqueueSnackbar('I love snacks.');
  };

  const handleClickVariant = (variant: VariantType) => () => {
    // variant could be success, error, warning, info, or default
    enqueueSnackbar('This is a success message!', { variant });
  };

  return (
    <React.Fragment>
      <Button onClick={handleClick}>Show snackbar</Button>
      <Button onClick={handleClickVariant('success')}>Show success snackbar</Button>
    </React.Fragment>
  );
}

export default function IntegrationNotistack() {
  return (
    <SnackbarProvider maxSnack={3}>
      <MyApp />
    </SnackbarProvider>
  );
}
```

⚠ Note that notistack prevents Snackbars from being closed by pressing `Escape`.

Accessibility

The user should be able to dismiss Snackbars by pressing `Escape`. If there are multiple instances appearing at the same time and you want `Escape` to dismiss only the oldest one that's currently open, call `event.preventDefault` in the `onClose` prop.

Copy

```
export default function MyComponent() {
  const [open, setOpen] = React.useState(true);

  return (
    <React.Fragment>
      <Snackbar
        open={open}
        onClose={(event, reason) => {
          // `reason === 'escapeKeyDown'` if `Escape` was pressed
          setOpen(false);
          // call `event.preventDefault` to only close one Snackbar at a time.
        }}
      />
      <Snackbar open={open} onClose={() => setOpen(false)} />
    </React.Fragment>
```

```
  );
}
```

Anatomy

The Snackbar component is composed of a root `<div>` that houses interior elements like the Snackbar Content and other optional components (such as buttons or decorators).

```
<div role="presentation" class="MuiSnackbar-root">
  <div class="MuiPaper-root MuiSnackbarContent-root" role="alert">
    <div class="MuiSnackbarContent-message">
      <!-- Snackbar content goes here -->
    </div>
  </div>
</div>
```

Copy

API

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

- `<Snackbar />`
- `<SnackbarContent />`

Skeleton

Accordion

MUI • Blog ↗ • Store ↗