Snackbar

Snackbars provide brief notifications. The component is also known as a toast.

Snackbars inform users of a process that an app has performed or will perform. They appear temporarily, towards the bottom of the screen. They shouldn't interrupt the user experience, and they don't require user input to disappear.

Snackbars contain a single line of text directly related to the operation performed. They may contain a text action, but no icons. You can use them to display notifications.

{{"component": "modules/components/ComponentLinkHeader.js"}}

Frequency: Only one snackbar may be displayed at a time.

Simple snackbars

A basic snackbar that aims to reproduce Google Keep's snackbar behavior.

{{"demo": "SimpleSnackbar.js"}}

Customization

Here are some examples of customizing the component. You can learn more about this in the overrides documentation page.

{{"demo": "CustomizedSnackbars.js"}}

Positioned snackbars

In wide layouts, snackbars can be left-aligned or center-aligned if they are consistently placed on the same spot at the bottom of the screen, however there may be circumstances where the placement of the snackbar needs to be more flexible. You can control the position of the snackbar by specifying the `anchorOrigin` prop.

{{"demo": "PositionedSnackbar.js"}}

Message Length

Some snackbars with varying message length.

{{"demo": "LongTextSnackbar.js"}}

Transitions

Consecutive Snackbars

When multiple snackbar updates are necessary, they should appear one at a time.

{{"demo": "ConsecutiveSnackbars.js"}}

Snackbars and floating action buttons (FABs)

Snackbars should appear above FABs (on mobile).

{{"demo": "FabIntegrationSnackbar.js", "iframe": true, "maxWidth": 400}}

Change transition

Grow is the default transition but you can use a different one.

{{"demo": "TransitionsSnackbar.js"}}

Control Slide direction

You can change the direction of the Slide transition.

Example of making the slide transition to the left:

```
import Slide from '@material-ui/core/Slide';

function TransitionLeft(props) {
  return <Slide {...props} direction="left" />;
}

export default function MyComponent() {
  return <Snackbar TransitionComponent={TransitionLeft} />;
}
```

Other examples:

{{"demo": "DirectionSnackbar.js"}}

Complementary projects

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

notistack

This example demonstrates how to use notistack. notistack has an **imperative API** that makes it easy to display snackbars, without having to handle their open/close state. It also enables you to **stack** them on top of one another (although this is discouraged by the Material Design guidelines).

{{"demo": "IntegrationNotistack.js", "defaultCodeOpen": false}}

Accessibility

(WAI-ARIA: https://www.w3.org/TR/wai-aria-1.1/#alert)

By default, the snackbar won't auto-hide. However, if you decide to use the `autoHideDuration` prop, it's recommended to give the user sufficient time to respond.

When open, **every** `Snackbar` will be dismissed if `Escape` is pressed. Unless you don't handle `onClose` with the `"escapeKeyDown"` reason. If you want to limit this behavior to only dismiss the oldest currently open Snackbar call `event.preventDefault` in `onClose`.

```jsx
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>
  );
}
```