Snackbar（消息条）

消息条提供简短的通知信息。 该组件也被称为 toast。

消息条将应用程序已执行或即将执行的进程通知给用户。 它们会从屏幕底部短暂地出现。 它们不应该打断用户体验，也不需要用户输入就能消失。

消息条包含了一行直接与所执行操作相关的文本。 它们可能包含一些文本操作，但并不会展示图标。 您也可以用他们展示通知。

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

我们规定一次只能显示一个消息条。

简单的消息条

一个简单的消息条旨在重现谷歌 Keep 消息条的行为。

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

自定义的消息条

你可以参考以下一些例子来自定义组件。 您可以在 重写文档页面 中了解更多有关此内容的信息。

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

定位的消息条

在宽大的布局中，如果消息条始终放在屏幕底部的同一位置，那么可以左对齐或中间对齐，但在某些情况下，消息条的位置可能需要更加灵活。 通过指定 `anchorOrigin` 的属性，您可以控制消息条的位置。

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

消息的长度

有些消息条会有不同的长度。

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

过渡动画

连续的消息条

当需要显示多个消息条的时候，一次应该只显示一个。

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

消息条（Snackbars）和悬浮按钮（FABs）

消息条应当显示在悬浮按钮的上方（这是在移动设备上）。

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

更改过渡动画

Grow 是默认的过渡动画，但你可以使用不同的过渡动画。

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

控制滑动的方向

你可以修改 Slide 过渡的方向 。

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} />;
}
```

对于更高级的用例，您可以利用：

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

补充项目

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

notistack

以下例子演示了如何使用 notistack。 notistack 有一个 **imperative API** 可以轻松地显示一串消息条，且无需处理其打开/关闭状态。 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}}

无障碍设计

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

默认情况下，消息条不会自动隐藏。 但是，如果您决定使用 `autoHideDuration` 属性，我们建议给用户提供 足够的时间 来响应。

当消息条打开时，如果Escape 键被按下，**每个** `Snackbar` 将会消失。 Unless you don't handle `onClose` with the `"escapeKeyDown"` reason. 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` .

```
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>
  );
}
        }}
      />
      <Snackbar open={open} onClose={() => setOpen(false)} />
    </React.Fragment>
  );
}
```