Edit this page

Button

Buttons allow users to take actions, and make choices, with a single tap.

Buttons communicate actions that users can take. They are typically placed throughout your UI, in places like:

- Modal windows
- Forms
- Cards
- Toolbars

Feedback WAI-ARIA Bundle size Material Design Figma Adobe Sketch

Basic button

The `Button` comes with three variants: text (default), contained, and outlined.

TEXT CONTAINED OUTLINED

```
<Button variant="text">Text</Button>
<Button variant="contained">Contained</Button>
<Button variant="outlined">Outlined</Button>
```

Text button

Text buttons are typically used for less-pronounced actions, including those located: in dialogs, in cards. In cards, text buttons help maintain an emphasis on card content.

PRIMARY LINK

```
<Button>Primary</Button>
<Button disabled>Disabled</Button>
<Button href="#text-buttons">Link</Button>
```

Contained button

Contained buttons are high-emphasis, distinguished by their use of elevation and fill. They contain actions that are primary to your app.

CONTAINED LINK

```
<Button variant="contained">Contained</Button>
<Button variant="contained" disabled>
  Disabled
</Button>
<Button variant="contained" href="#contained-buttons">
  Link
</Button>
```

You can remove the elevation with the `disableElevation` prop.

DISABLE ELEVATION

```
<Button variant="contained" disableElevation>
  Disable elevation
</Button>
```

Outlined button

Outlined buttons are medium-emphasis buttons. They contain actions that are important but aren't the primary action in an app.

Outlined buttons are also a lower emphasis alternative to contained buttons, or a higher emphasis alternative to text buttons.


```
<Button variant="outlined">Primary</Button>
<Button variant="outlined" disabled>
  Disabled
</Button>
<Button variant="outlined" href="#outlined-buttons">
  Link
</Button>
```

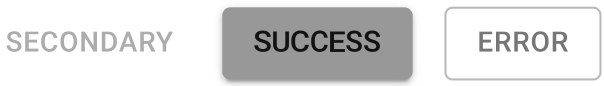
Handling clicks

All components accept an `onClick` handler that is applied to the root DOM element.

```
<Button
  onClick={() => {
    alert('clicked');
  }}
>
  Click me
</Button>
```

Note that the documentation avoids mentioning native props (there are a lot) in the API section of the components.

Color

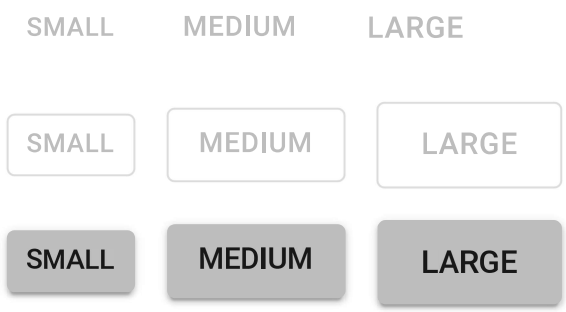


```
<Button color="secondary">Secondary</Button>
<Button variant="contained" color="success">
  Success
</Button>
<Button variant="outlined" color="error">
  Error
</Button>
```

In addition to using the default button colors, you can add custom ones, or disable any you don't need. See the Adding new colors example for more info.

Sizes

For larger or smaller buttons, use the `size` prop.



Upload button


```
<Button variant="contained" component="label">
  Upload
  <input hidden accept="image/*" multiple type="file" />
</Button>
<IconButton color="primary" aria-label="upload picture" component="label">
  <input hidden accept="image/*" type="file" />
  <PhotoCamera />
</IconButton>
```

Buttons with icons and label

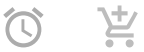
Sometimes you might want to have icons for certain buttons to enhance the UX of the application as we recognize logos more easily than plain text. For example, if you have a delete button you can label it with a dustbin icon.


```
<Button variant="outlined" startIcon={<DeleteIcon />}>
  Delete
</Button>
<Button variant="contained" endIcon={<SendIcon />}>
  Send
</Button>
```

Icon button

Icon buttons are commonly found in app bars and toolbars.

Icons are also appropriate for toggle buttons that allow a single choice to be selected or deselected, such as adding or removing a star to an item.



```
<IconButton aria-label="delete">
  <DeleteIcon />
</IconButton>
<IconButton aria-label="delete" disabled color="primary">
  <DeleteIcon />
</IconButton>
<IconButton color="secondary" aria-label="add an alarm">
  <AlarmIcon />
</IconButton>
<IconButton color="primary" aria-label="add to shopping cart">
  <AddShoppingCartIcon />
</IconButton>
```

Sizes

For larger or smaller icon buttons, use the `size` prop.

```
<IconButton aria-label="delete" size="small">
  <DeleteIcon fontSize="inherit" />
</IconButton>
<IconButton aria-label="delete" size="small">
  <DeleteIcon fontSize="small" />
</IconButton>
<IconButton aria-label="delete" size="large">
  <DeleteIcon />
</IconButton>
<IconButton aria-label="delete" size="large">
  <DeleteIcon fontSize="inherit" />
</IconButton>
```

Colors

Use `color` prop to apply theme color palette to component.

```
<IconButton aria-label="fingerprint" color="secondary">
  <Fingerprint />
</IconButton>
<IconButton aria-label="fingerprint" color="success">
  <Fingerprint />
</IconButton>
```

Customization

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

If you are looking for inspiration, you can check MUI Treasury's customization examples.

Loading button

The loading buttons can show loading state and disable interactions.

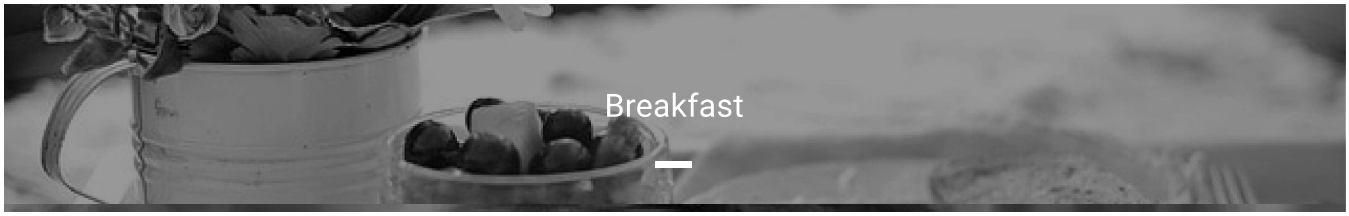
```
<LoadingButton loading variant="outlined">
  Submit
</LoadingButton>
<LoadingButton loading loadingIndicator="Loading…" variant="outlined">
  Fetch data
</LoadingButton>
<LoadingButton
  loading
  loadingPosition="start"
  startIcon={<SaveIcon />}
  variant="outlined"
>
  Save
</LoadingButton>
```

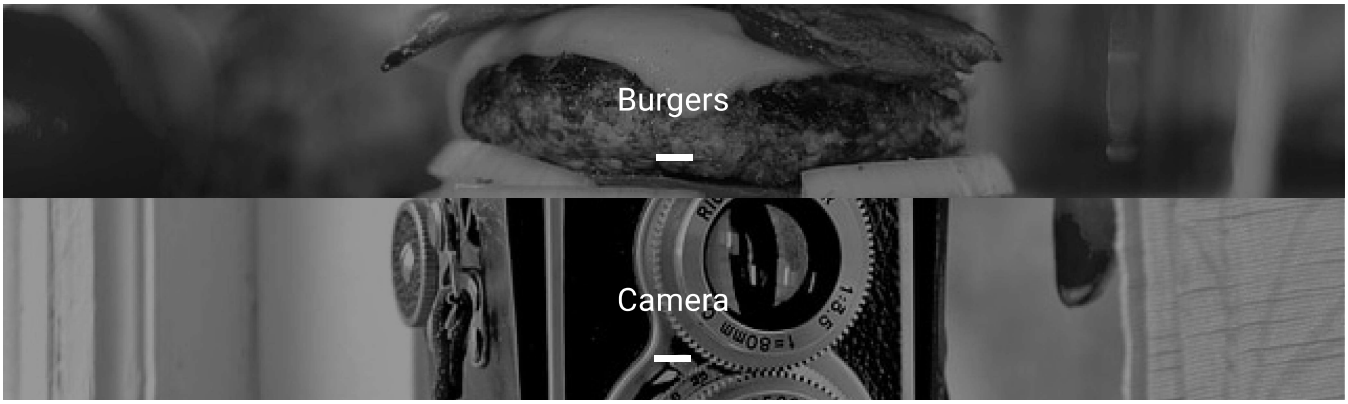
Toggle the loading switch to see the transition between the different states.

Loading

Complex button

The Text Buttons, Contained Buttons, Floating Action Buttons and Icon Buttons are built on top of the same component: the `ButtonBase`. You can take advantage of this lower-level component to build custom interactions.





Third-party routing library

One frequent use case is to perform navigation on the client only, without an HTTP round-trip to the server. The `ButtonBase` component provides the `component` prop to handle this use case. Here is a more detailed guide.

Limitations

Cursor not-allowed

The ButtonBase component sets `pointer-events: none;` on disabled buttons, which prevents the appearance of a disabled cursor.

If you wish to use `not-allowed`, you have two options:

1. **CSS only**. You can remove the pointer-events style on the disabled state of the `<button>` element:

```
.MuiButtonBase-root:disabled {
  cursor: not-allowed;
  pointer-events: auto;
}
```

However:

- You should add `pointer-events: none;` back when you need to display tooltips on disabled elements.
- The cursor won't change if you render something other than a button element, for instance, a link `<a>` element.

2. **DOM change**. You can wrap the button:

```
<span style={{ cursor: 'not-allowed' }}>
  <Button component={Link} disabled>
    disabled
  </Button>
</span>
```

This has the advantage of supporting any element, for instance, a link `<a>` element.

Unstyled

The component also comes with an unstyled version. It's ideal for doing heavy customizations and minimizing bundle size.

API

- `<Button />`
- `<ButtonBase />`
- `<IconButton />`
- `<LoadingButton />`

‹ Autocomplete Button Group ›