Search... Ctrl+K

Button

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

Check out the latest remote job listings from the leading job board for designers, developers, and creative pros.

ads via Carbon

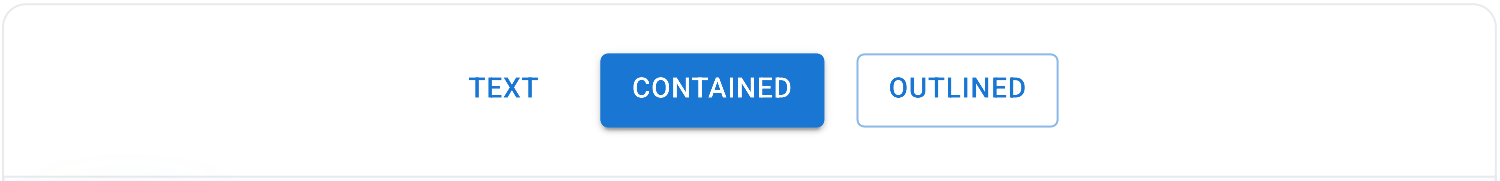
Buttons communicate actions that users can take. They are typically placed throughout your UI, in places like:

- Modal windows
- Forms
- Cards
- Toolbars

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

Basic button

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



Edit in Chat JS TS Collapse code

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

export default function BasicButtons() {
  return (
    <Stack spacing={2} direction="row">
      <Button variant="text">Text</Button>
      <Button variant="contained">Contained</Button>
      <Button variant="outlined">Outlined</Button>
    </Stack>
  );
}
```

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 DISABLED LINK

Edit in Chat JS TS Collapse code

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

export default function TextButtons() {
  return (
    <Stack direction="row" spacing={2}>
      <Button>Primary</Button>
      <Button disabled>Disabled</Button>
      <Button href="#text-buttons">Link</Button>
    </Stack>
  );
}
```

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 DISABLED LINK

Edit in Chat JS TS Collapse code

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

export default function ContainedButtons() {
  return (
    <Stack direction="row" spacing={2}>
      <Button variant="contained">Contained</Button>
      <Button variant="contained" disabled>
```

```
        Disabled
      </Button>
      <Button variant="contained" href="#contained-buttons">
        Link
      </Button>
    </Stack>
  );
}
```

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

Edit in Chat JS TS Collapse code

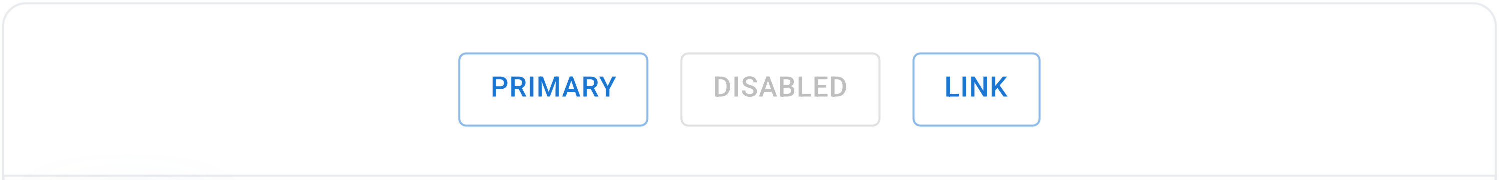
```
import Button from '@mui/material/Button';

export default function DisableElevation() {
  return (
    <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.



Edit in Chat JS TS Collapse code

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

export default function OutlinedButtons() {
  return (
    <Stack direction="row" spacing={2}>
      <Button variant="outlined">Primary</Button>
      <Button variant="outlined" disabled>
        Disabled
      </Button>
      <Button variant="outlined" href="#outlined-buttons">
        Link
      </Button>
    </Stack>
  );
}
```

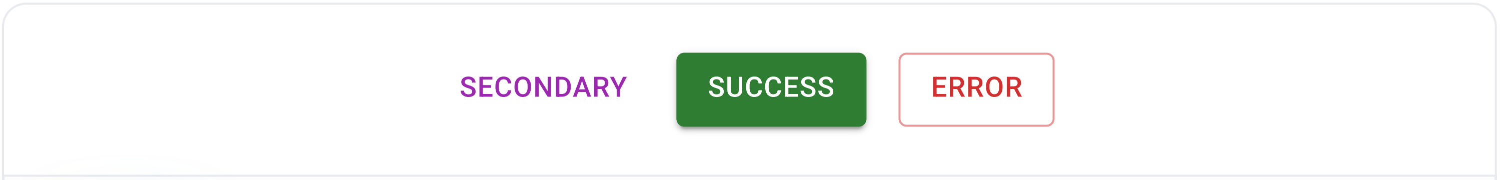
Handling clicks

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



Edit in Chat JS TS Collapse code

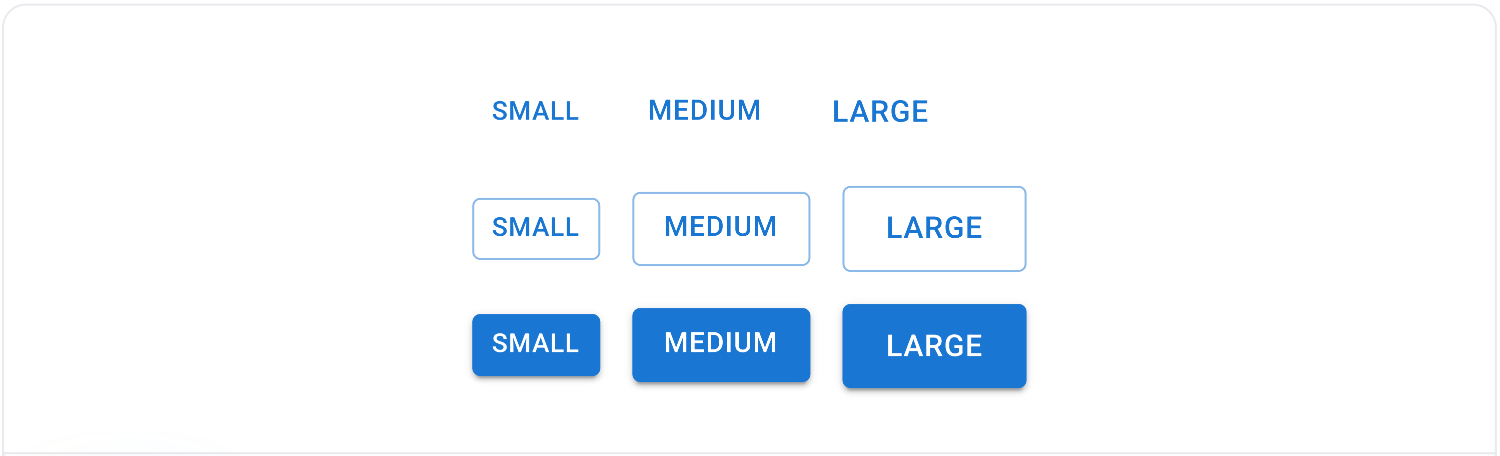
```
import Stack from '@mui/material/Stack';
import Button from '@mui/material/Button';

export default function ColorButtons() {
  return (
    <Stack direction="row" spacing={2}>
      <Button color="secondary">Secondary</Button>
      <Button variant="contained" color="success">
        Success
      </Button>
      <Button variant="outlined" color="error">
        Error
      </Button>
    </Stack>
  );
}
```

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](#) examples for more info.

Sizes

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



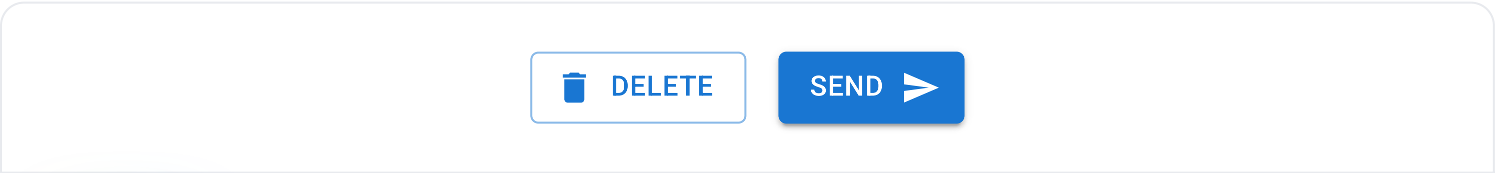
Edit in Chat JS TS Hide code

```
import Box from '@mui/material/Box';
import Button from '@mui/material/Button';

export default function ButtonSizes() {
  return (
    <Box sx={{ '& button': { m: 1 } }}>
      <div>
        <Button size="small">Small</Button>
        <Button size="medium">Medium</Button>
        <Button size="large">Large</Button>
      </div>
      <div>
        <Button variant="outlined" size="small">
          Small
        </Button>
        <Button variant="outlined" size="medium">
          Medium
        </Button>
        <Button variant="outlined" size="large">
          Large
        </Button>
      </div>
      <div>
        <Button variant="contained" size="small">
          Small
        </Button>
        <Button variant="contained" size="medium">
          Medium
        </Button>
        <Button variant="contained" size="large">
          Large
        </Button>
      </div>
    </Box>
  );
}
```

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.



Edit in Chat JS TS Collapse code

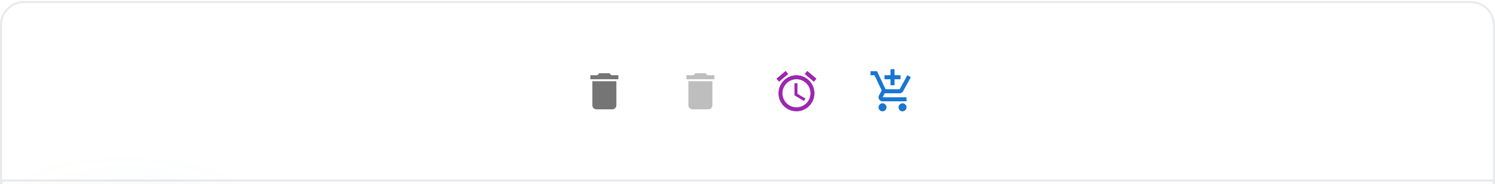
```
import Button from '@mui/material/Button';
import DeleteIcon from '@mui/icons-material/Delete';
import SendIcon from '@mui/icons-material/Send';
import Stack from '@mui/material/Stack';

export default function IconLabelButtons() {
  return (
    <Stack direction="row" spacing={2}>
      <Button variant="outlined" startIcon={<DeleteIcon />}>
        Delete
      </Button>
      <Button variant="contained" endIcon={<SendIcon />}>
        Send
      </Button>
    </Stack>
  );
}
```

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.



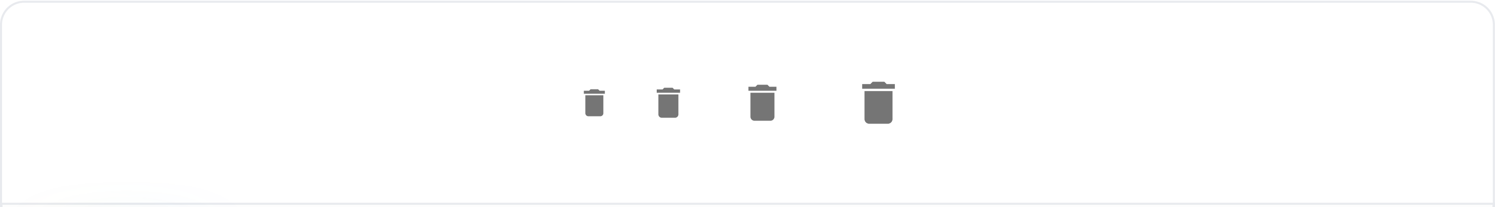
Edit in Chat JS TS Collapse code

```
import IconButton from '@mui/material/IconButton';
import Stack from '@mui/material/Stack';
import DeleteIcon from '@mui/icons-material/Delete';
import AlarmIcon from '@mui/icons-material/Alarm';
import AddShoppingCartIcon from '@mui/icons-material/AddShoppingCart';

export default function IconButtons() {
  return (
    <Stack direction="row" spacing={1}>
      <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>
    </Stack>
  );
}
```

Sizes

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



Edit in Chat JS TS Collapse code

```
import Stack from '@mui/material/Stack';
import IconButton from '@mui/material/IconButton';
import DeleteIcon from '@mui/icons-material/Delete';

export default function IconButtonSizes() {
  return (
    <Stack direction="row" spacing={1} sx={{ alignItems: 'center' }}>
      <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>
    </Stack>
  );
}
```

Colors

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

```
import Stack from '@mui/material/Stack';
import IconButton from '@mui/material/IconButton';
import Fingerprint from '@mui/icons-material/Fingerprint';

export default function IconButtonColors() {
  return (
    <Stack direction="row" spacing={1}>
      <IconButton aria-label="fingerprint" color="secondary">
        <Fingerprint />
      </IconButton>
      <IconButton aria-label="fingerprint" color="success">
        <Fingerprint />
      </IconButton>
    </Stack>
  );
}
```

Loading

Starting from v6.4.0, use `loading` prop to set icon buttons in a loading state and disable interactions.

Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Tooltip from '@mui/material/Tooltip';
import IconButton from '@mui/material/IconButton';
import ShoppingCartIcon from '@mui/icons-material/ShoppingCart';

export default function LoadingIconButton() {
  const [loading, setLoading] = React.useState(false);
  React.useEffect(() => {
    const timeout = setTimeout(() => {
      setLoading(false);
    }, 2000);
    return () => clearTimeout(timeout);
  });
  return (
    <Tooltip title="Click to see loading">
      <IconButton onClick={() => setLoading(true)} loading={loading}>
        <ShoppingCartIcon />
      </IconButton>
    </Tooltip>
  );
}
```

Badge

You can use the `Badge` component to add a badge to an `IconButton`.

2

Edit in Chat JS TS Collapse code

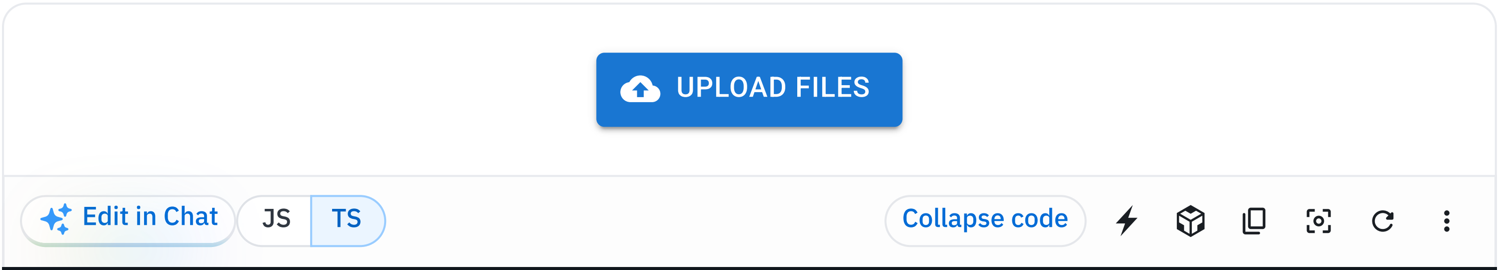
```tsx
import { styled } from '@mui/material/styles';
import IconButton from '@mui/material/IconButton';
import Badge, { badgeClasses } from '@mui/material/Badge';
import ShoppingCartIcon from '@mui/icons-material/ShoppingCartOutlined';

const CartBadge = styled(Badge)`
  & .${badgeClasses.badge} {
    top: -12px;
    right: -6px;
  }
`;
```

```
export default function IconButtonWithBadge() {
  return (
    <IconButton>
      <ShoppingCartIcon fontSize="small" />
      <CartBadge badgeContent={2} color="primary" overlap="circular" />
    </IconButton>
  );
}
```

File upload

To create a file upload button, turn the button into a label using `component="label"` and then create a visually-hidden input with type `file`.



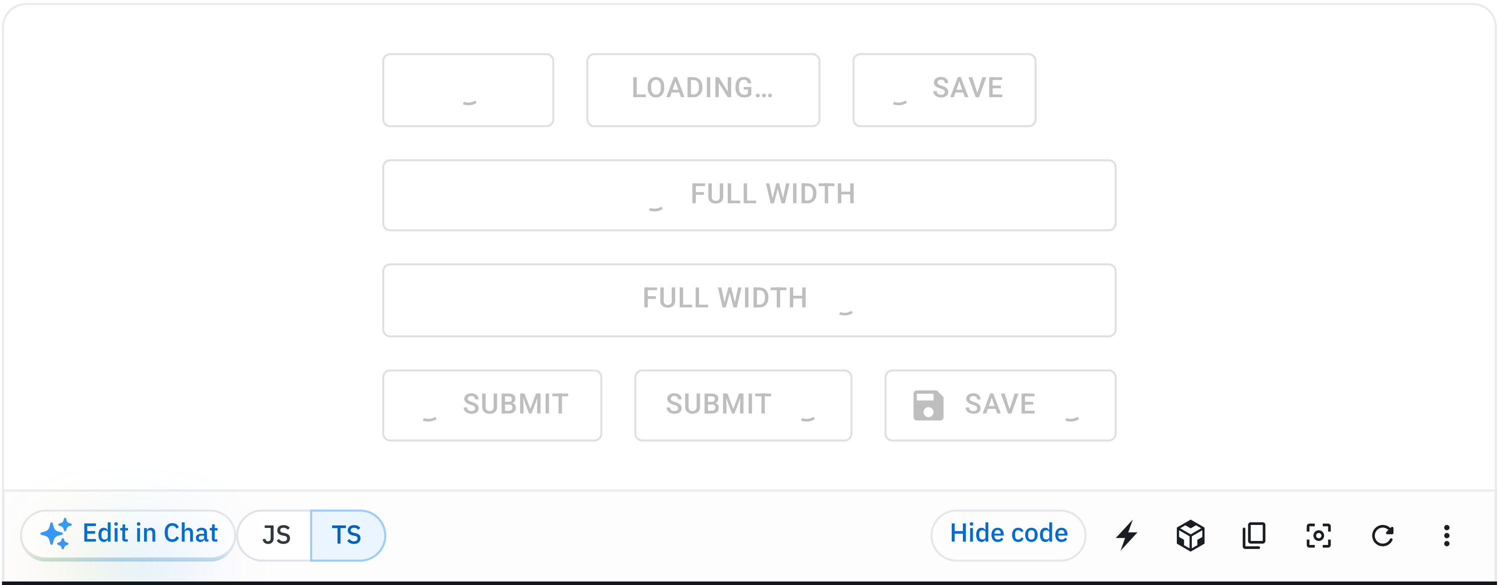
```
import { styled } from '@mui/material/styles';
import Button from '@mui/material/Button';
import CloudUploadIcon from '@mui/icons-material/CloudUpload';

const VisuallyHiddenInput = styled('input')({
  clip: 'rect(0 0 0 0)',
  clipPath: 'inset(50%)',
  height: 1,
  overflow: 'hidden',
  position: 'absolute',
  bottom: 0,
  left: 0,
  whiteSpace: 'nowrap',
  width: 1,
});

export default function InputFileUpload() {
  return (
    <Button
      component="label"
      role={undefined}
      variant="contained"
      tabIndex={-1}
      startIcon={<CloudUploadIcon />}
    >
      Upload files
      <VisuallyHiddenInput
        type="file"
        onChange={(event) => console.log(event.target.files)}
        multiple
      />
    </Button>
  );
}
```

Loading

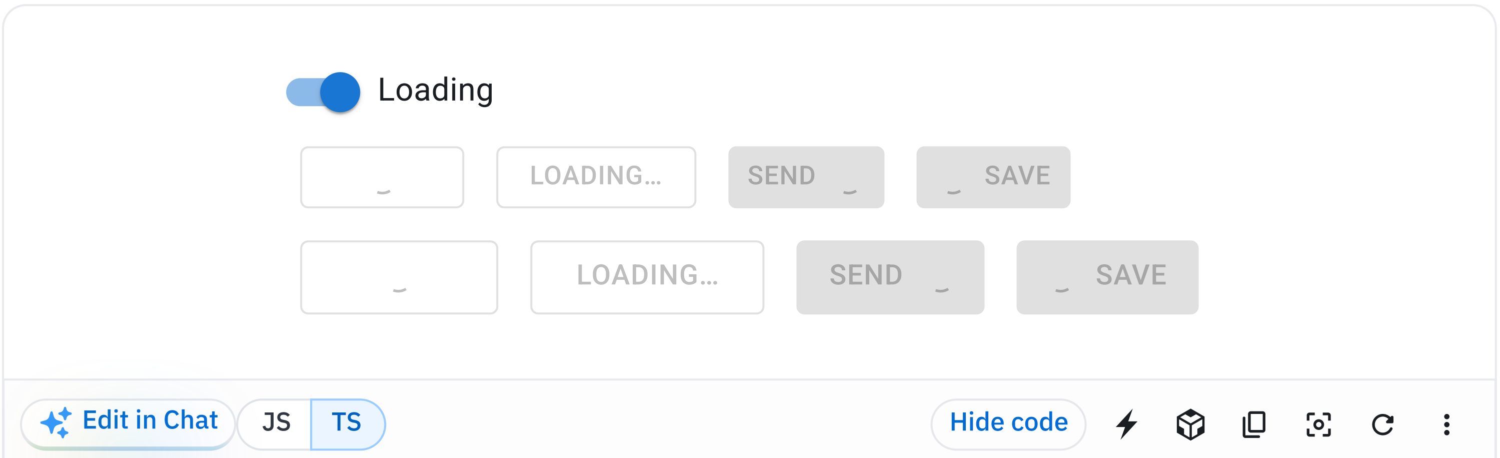
Starting from v6.4.0, use the `loading` prop to set buttons in a loading state and disable interactions.



```
import Button from '@mui/material/Button';
import SaveIcon from '@mui/icons-material/Save';
import Stack from '@mui/material/Stack';

export default function LoadingButtons() {
  return (
    <Stack spacing={2}>
      <Stack direction="row" spacing={2}>
        <Button loading variant="outlined">
          Submit
        </Button>
        <Button loading loadingIndicator="Loading…" variant="outlined">
          Fetch data
        </Button>
        <Button
          loading
          loadingPosition="start"
          startIcon={<SaveIcon />}
          variant="outlined"
        >
          Save
        </Button>
      </Stack>
      <Button
        fullWidth
        loading
        loadingPosition="start"
        startIcon={<SaveIcon />}
        variant="outlined"
      >
        Full width
      </Button>
      <Button
        fullWidth
        loading
        loadingPosition="end"
        endIcon={<SaveIcon />}
        variant="outlined"
```

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



```
import * as React from 'react';
import Button from '@mui/material/Button';
import Box from '@mui/material/Box';
import FormControlLabel from '@mui/material/FormControlLabel';
import Switch from '@mui/material/Switch';
import SaveIcon from '@mui/icons-material/Save';
import SendIcon from '@mui/icons-material/Send';

export default function LoadingButtonsTransition() {
  const [loading, setLoading] = React.useState(true);
  function handleClick() {
    setLoading(true);
  }

  return (
    <div>
      <FormControlLabel
        sx={{ display: 'block' }}
        control={
          <Switch
            checked={loading}
            onChange={() => setLoading(!loading)}
            name="loading"
            color="primary"
          />
        }
        label="Loading"
      />
      <Box sx={{ '& > button': { m: 1 } }}>
        <Button
          size="small"
          onClick={handleClick}
          loading={loading}
          variant="outlined"
          disabled
        >
          Disabled
        </Button>
```

⚠ When the `loading` prop is set to `boolean`, the loading wrapper is always present in the DOM to prevent a Google Translation Crash.

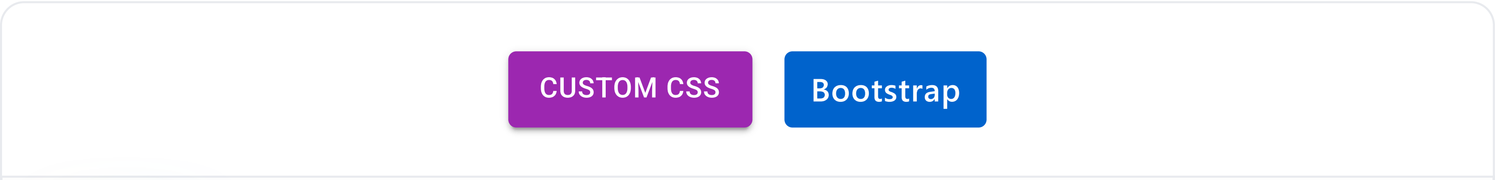
The `loading` value should always be `null` or `boolean`. The pattern below is not recommended as it can cause the Google Translation crash:

```
<Button {...(isFetching && { loading: true })}> // ❌ Don't do this
```

Copy

Customization

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



Edit in Chat JS TS Hide code

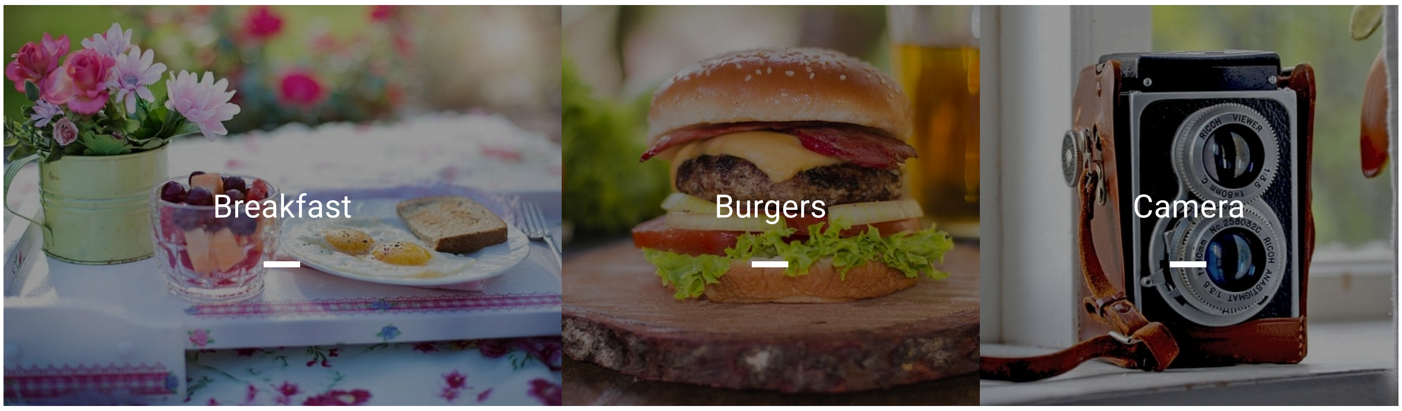
```
import { styled } from '@mui/material/styles';
import Button, { ButtonProps } from '@mui/material/Button';
import Stack from '@mui/material/Stack';
import { purple } from '@mui/material/colors';

const BootstrapButton = styled(Button)({
  boxShadow: 'none',
  textTransform: 'none',
  fontSize: 16,
  padding: '6px 12px',
  border: '1px solid',
  lineHeight: 1.5,
  backgroundColor: '#0063cc',
  borderColor: '#0063cc',
  fontFamily: [
    '-apple-system',
    'BlinkMacSystemFont',
    '"Segoe UI"',
    'Roboto',
    '"Helvetica Neue"',
    'Arial',
    'sans-serif',
    '"Apple Color Emoji"',
    '"Segoe UI Emoji"',
    '"Segoe UI Symbol"',
  ].join(','),
  '&:hover': {
    backgroundColor: '#0069d9',
    borderColor: '#0062cc',
    boxShadow: 'none',
  },
  '&:active': {
    boxShadow: 'none',
    backgroundColor: '#0062cc',
    borderColor: '#005cbf',
  },
  '&:focus': {
    boxShadow: '0 0 0 0.2rem rgba(0,123,255,.5)',
```

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

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.



Edit in Chat JS TS Hide code

```
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import ButtonBase from '@mui/material/ButtonBase';
import Typography from '@mui/material/Typography';

const images = [
  {
    url: '/static/images/buttons/breakfast.jpg',
    title: 'Breakfast',
    width: '40%',
  },
  {
    url: '/static/images/buttons/burgers.jpg',
    title: 'Burgers',
    width: '30%',
  },
  {
    url: '/static/images/buttons/camera.jpg',
    title: 'Camera',
    width: '30%',
  },
];

const ImageButton = styled(ButtonBase)(({ theme }) => ({
  position: 'relative',
  height: 200,
  [theme.breakpoints.down('sm')]: {
    width: '100% !important', // Overrides inline-style
    height: 100,
  },
  '&:hover, &.Mui-focusVisible': {
    zIndex: 1,
    '& .MuiImageBackdrop-root': {
      opacity: 0.15,
    },
    '& .MuiImageMarked-root': {
      opacity: 0,
    },
```

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:

```css
.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:

```jsx
<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.

API

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

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