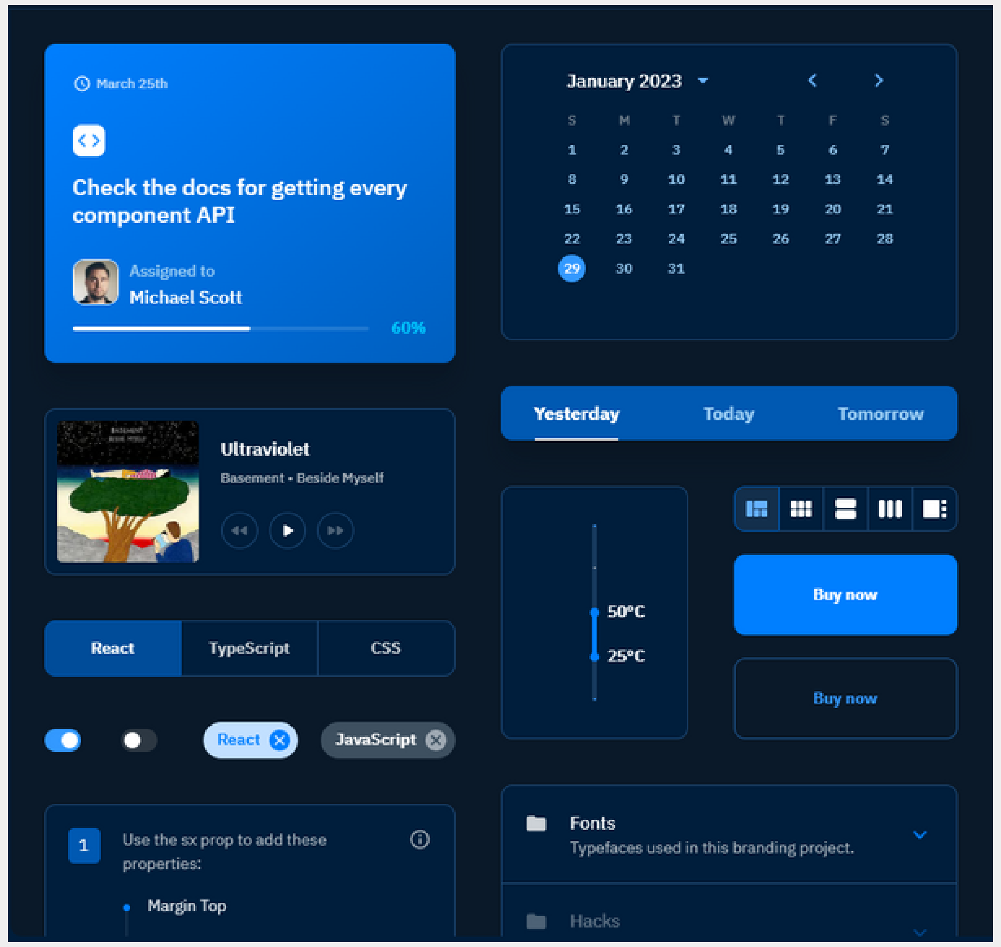
What is UI Framework



A set of **ready-to-use** react UI components.

Why UI Framework?

1. It saves time! Building the entire UI library from scratch take huge effort and time.
2. Design Consistency Using the same UI components allows designers to increase consistency while minimizing errors and rework
3. Easy to Scale
4. Easy to Maintenance

Famous UI Frameworks

Ant Design

MUI

ChakraUI

Semantic UI

React Bootstrap

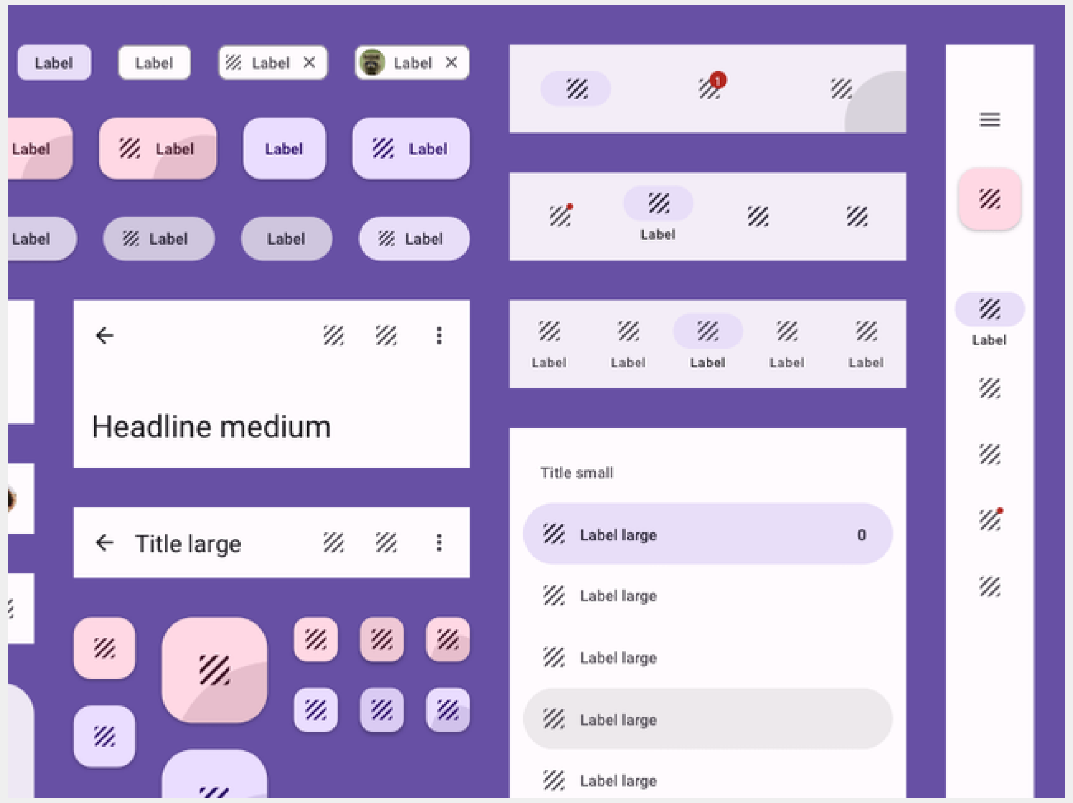
Evergreen UI

MUI Framework

MUI (Material UI) is a massive library of UI components designers and developers can use to build React applications. MUI based on Material Design from Google

https://mui.com/material-ui/getting-started/overview/

What is Material Design?



Material is a design system created by Google to help teams build high-quality digital experiences for Android, iOS, Flutter, and the web.

Material Design is inspired by the physical world and its textures, including how they reflect light and cast shadows. Material surfaces reimagine the mediums of paper and ink.

https://m2.material.io/design/introduction

Installation

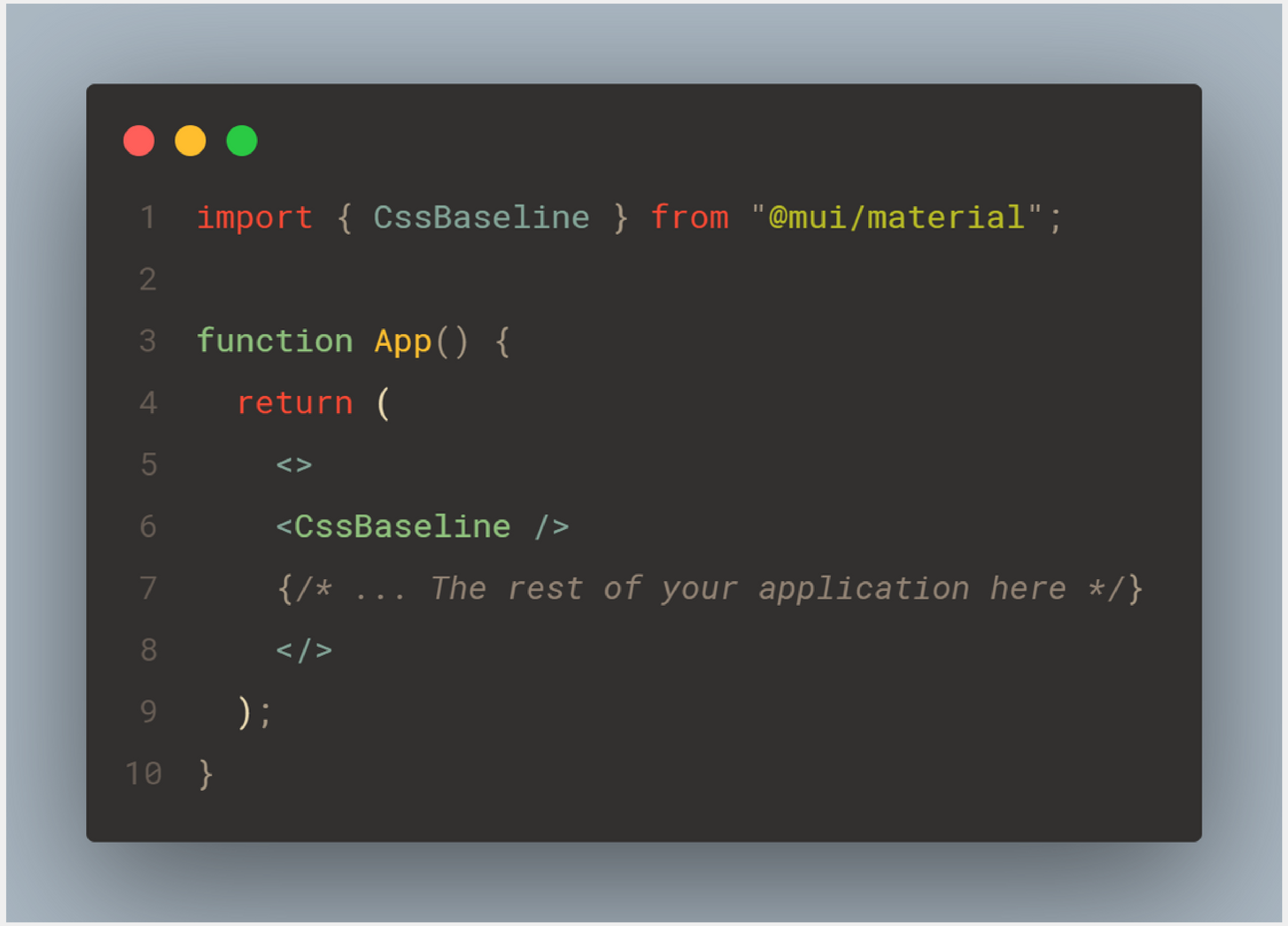
```
npm install @mui/material @emotion/react @emotion/styled
```

```
npm install @mui/icons-material
```

(for ready-to-use SVG Icon components)

https://mui.com/material-ui/getting-started/installation/

Setup: Reset style



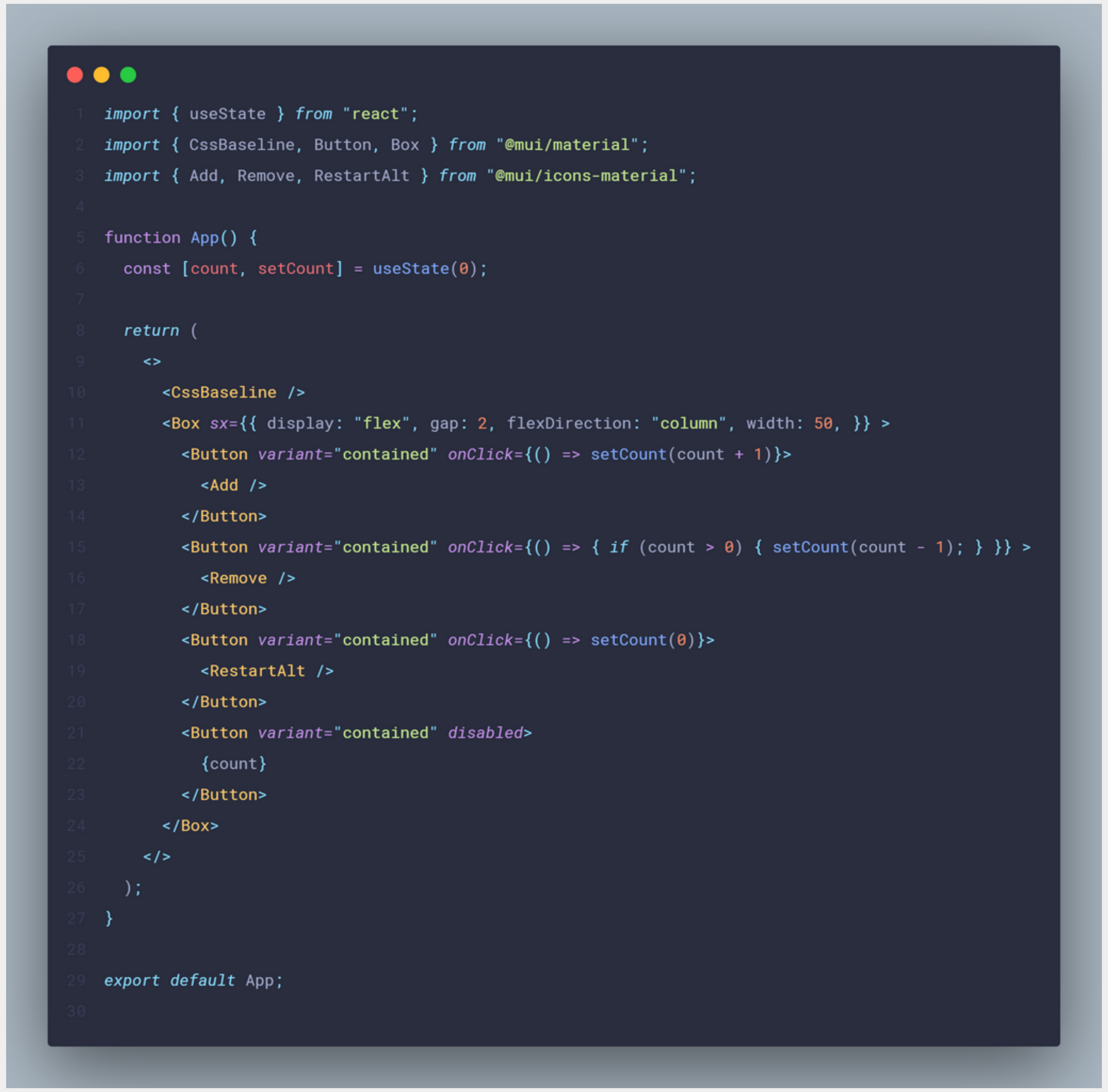
```
import { CssBaseline } from "@mui/material";

function App() {
  return (
    <>
    <CssBaseline />
    {/* ... The rest of your application here */}
    </>
  );
}
```

https://mui.com/material-ui/react-css-baseline/#approach

https://meyerweb.com/eric/tools/css/reset/

Quick Usage

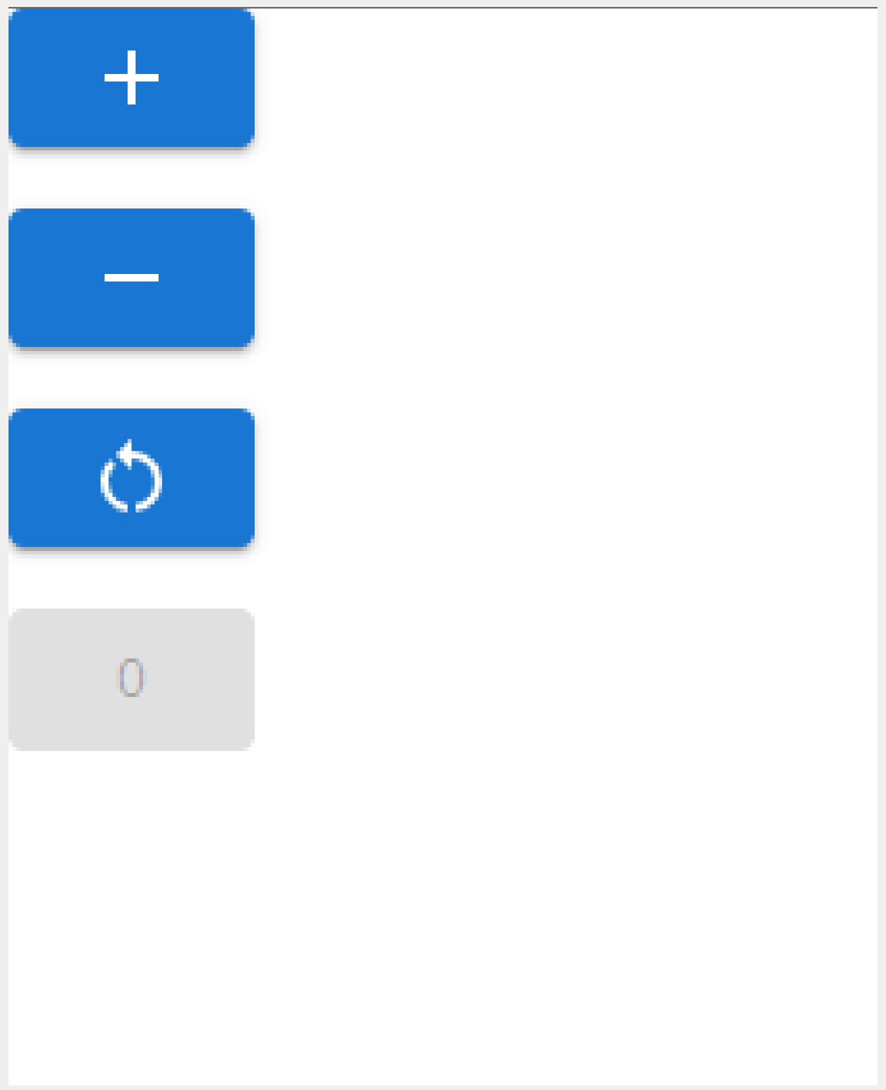


```
import { useState } from "react";
import { CssBaseline, Button, Box } from "@mui/material";
import { Add, Remove, RestartAlt } from "@mui/icons-material";

function App() {
  const [count, setCount] = useState(0);

  return (
    <>
      <CssBaseline />
      <Box sx={{ display: "flex", gap: 2, flexDirection: "column", width: 50, }} >
        <Button variant="contained" onClick={() => setCount(count + 1)}>
          <Add />
        </Button>
        <Button variant="contained" onClick={() => { if (count > 0) { setCount(count - 1); } }} >
          <Remove />
        </Button>
        <Button variant="contained" onClick={() => setCount(0)}>
          <RestartAlt />
        </Button>
        <Button variant="contained" disabled>
          {count}
        </Button>
      </Box>
    </>
  );
}

export default App;
```



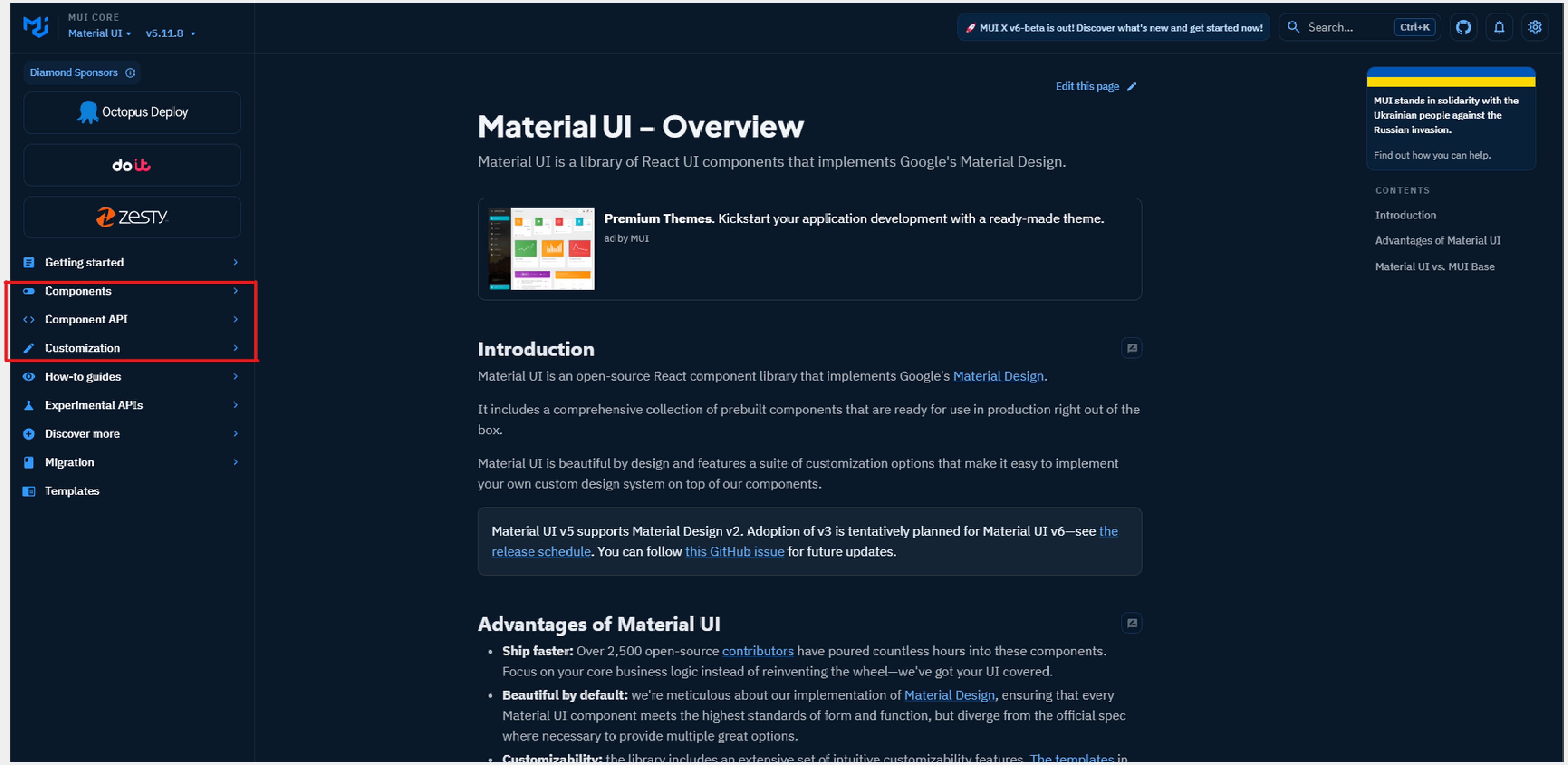
Code

```jsx
import { useState } from "react";
import { CssBaseline, Button, Box } from "@mui/material";
import { Add, Remove, RestartAlt } from "@mui/icons-material";

function App() {
  const [count, setCount] = useState(0);
  return (
    <>
      <CssBaseline />
      <Box sx={{ display: "flex", gap: 2, flexDirection: "column", width: 50, }} >
        <Button variant="contained" onClick={() => setCount(count + 1)}>
          <Add />
        </Button>
        <Button variant="contained" onClick={() => { if (count > 0) { setCount(count - 1); } }} >
          <Remove />
        </Button>
        <Button variant="contained" onClick={() => setCount(0)}>
          <RestartAlt />
        </Button>
        <Button variant="contained" disabled>
          {count}
        </Button>
      </Box>
    </>
  );
}

export default App;
```

MUI Document Overview



MUI stands in solidarity with the Ukrainian people against the Russian invasion.

Find out how you can help.

Material UI – Overview

Material UI is a library of React UI components that implements Google's Material Design.

Premium Themes. Kickstart your application development with a ready-made theme.
ad by MUI

Introduction

Material UI is an open-source React component library that implements Google's Material Design.

It includes a comprehensive collection of prebuilt components that are ready for use in production right out of the box.

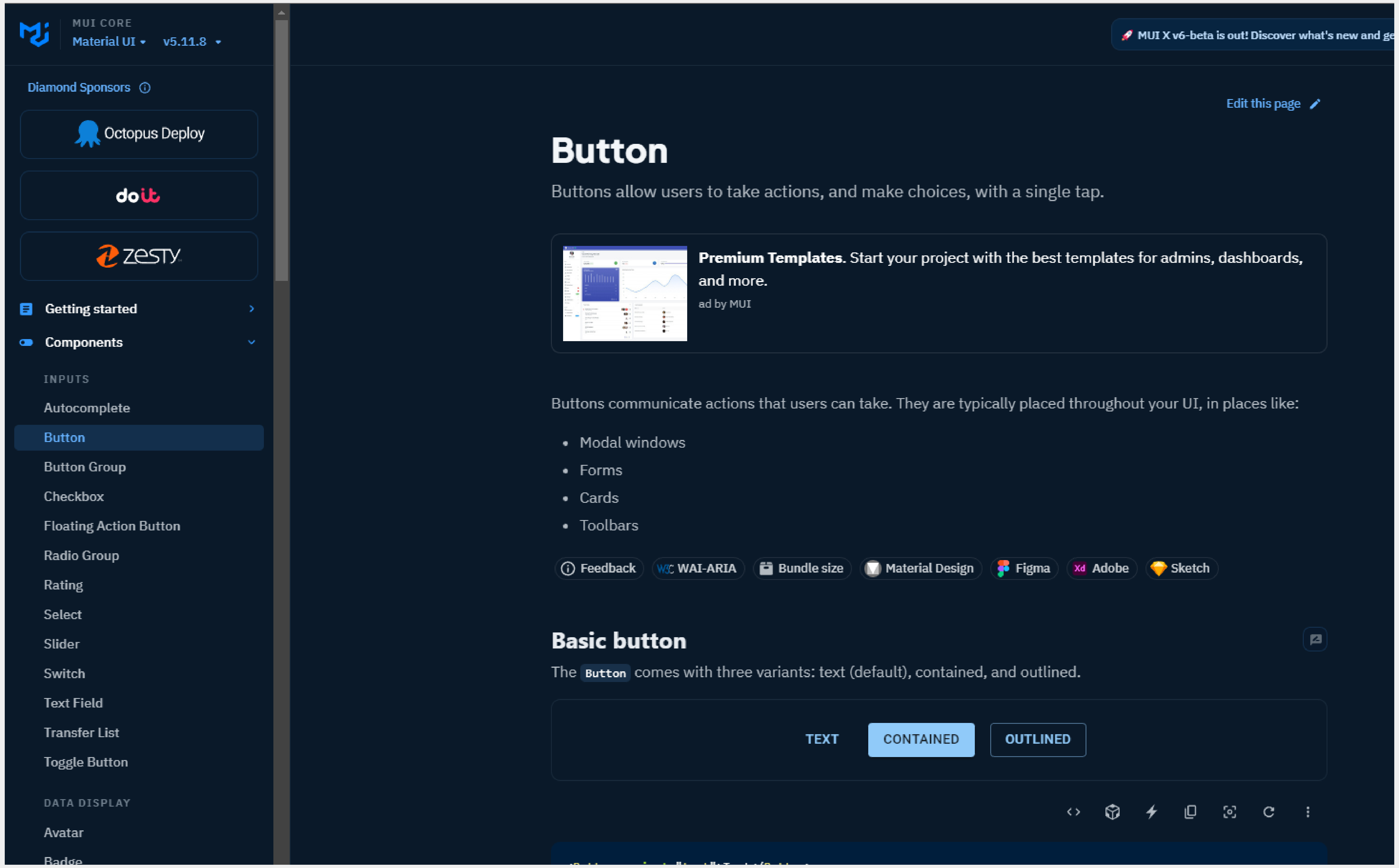
Material UI is beautiful by design and features a suite of customization options that make it easy to implement your own custom design system on top of our components.

Material UI v5 supports Material Design v2. Adoption of v3 is tentatively planned for Material UI v6—see the release schedule. You can follow this GitHub issue for future updates.

Advantages of Material UI

- **Ship faster:** Over 2,500 open-source contributors have poured countless hours into these components. Focus on your core business logic instead of reinventing the wheel—we've got your UI covered.
- **Beautiful by default:** we're meticulous about our implementation of Material Design, ensuring that every Material UI component meets the highest standards of form and function, but diverge from the official spec where necessary to provide multiple great options.
- **Customizability:** the library includes an extensive set of intuitive customizability features. The templates in

MUI Document Overview



MUI CORE
Material UI ▾ v5.11.8 ▾

Diamond Sponsors

Octopus Deploy
doit
zesty

Getting started
Components

INPUTS
- Autocomplete
- Button
- Button Group
- Checkbox
- Floating Action Button
- Radio Group
- Rating
- Select
- Slider
- Switch
- Text Field
- Transfer List
- Toggle Button

DATA DISPLAY
- Avatar
- Badge

MUI X v6-beta is out! Discover what's new and ge

Edit this page

Button

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

Premium Templates. Start your project with the best templates for admins, dashboards, and more.
ad by MUI

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

- Modal windows
- Forms
- Cards
- Toolbars

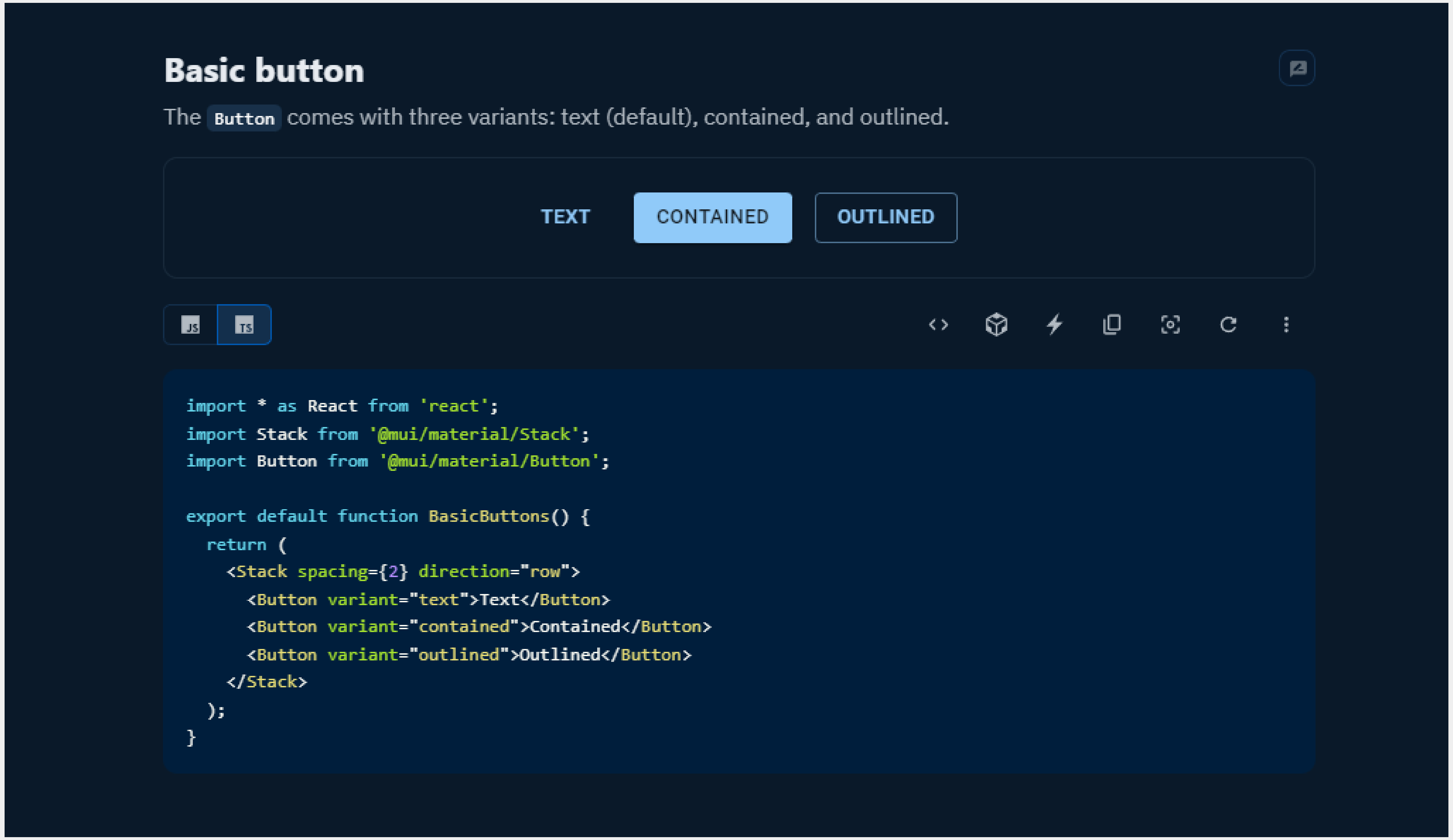
Feedback · WAI-ARIA · Bundle size · Material Design · Figma · Adobe · Sketch

Basic button

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

TEXT CONTAINED OUTLINED

MUI Document Overview



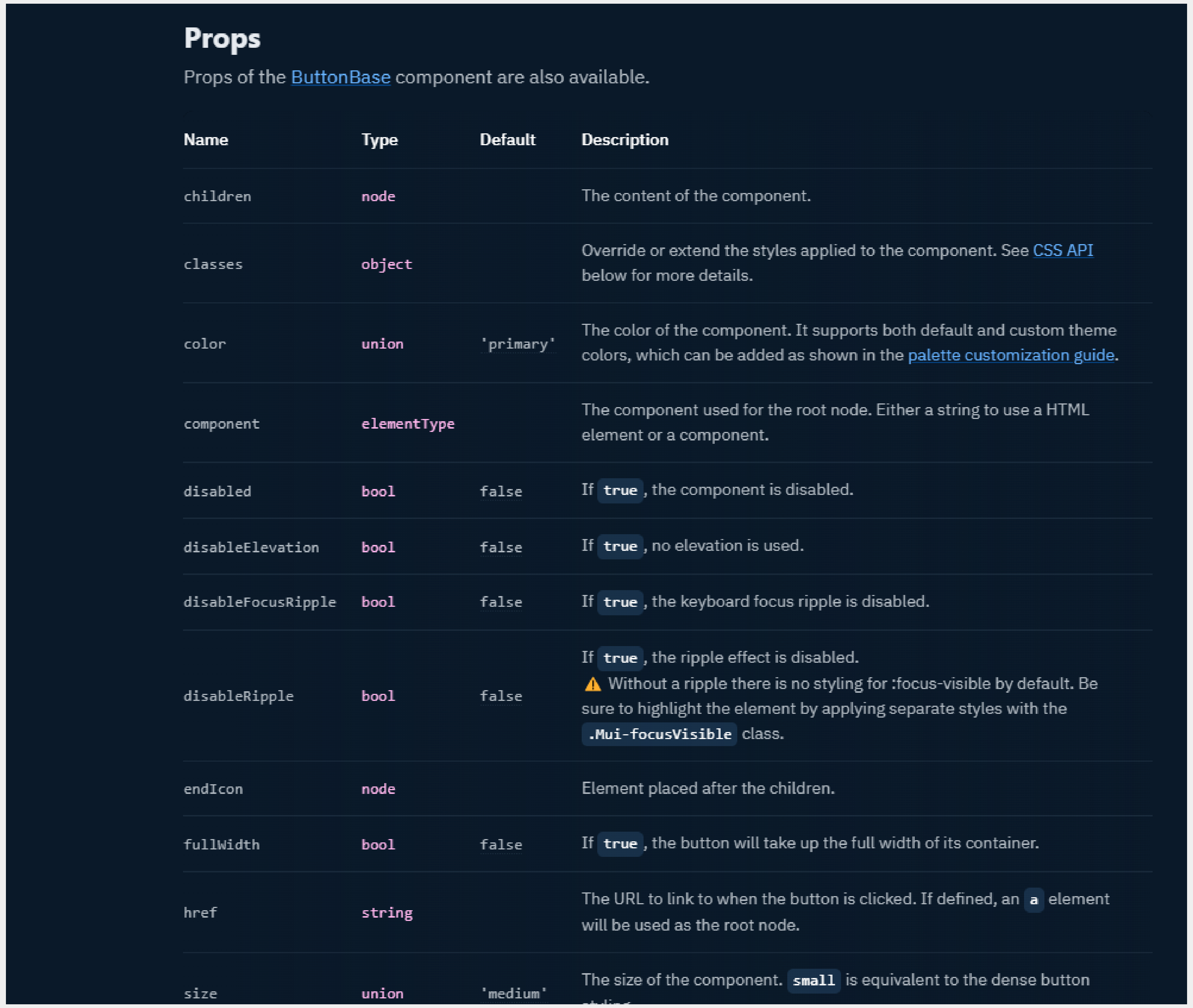
Basic button

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

```
import * as React from 'react';
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>
  );
}
```

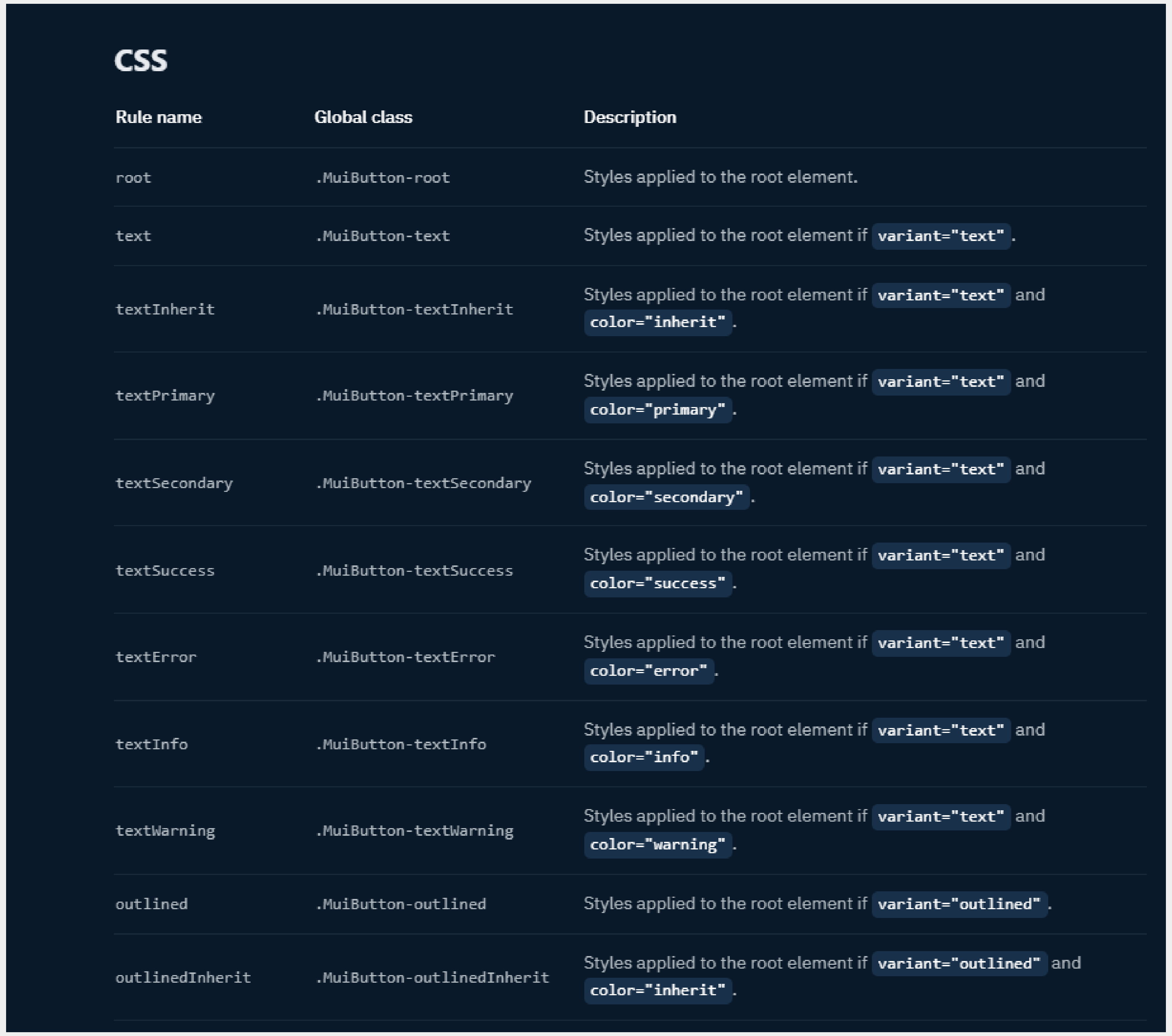
MUI Document Overview



Props

Props of the ButtonBase component are also available.

Name	Type	Default	Description
`children`	`node`		The content of the component.
`classes`	`object`		Override or extend the styles applied to the component. See CSS API below for more details.
`color`	`union`	`'primary'`	The color of the component. It supports both default and custom theme colors, which can be added as shown in the palette customization guide.
`component`	`elementType`		The component used for the root node. Either a string to use a HTML element or a component.
`disabled`	`bool`	`false`	If `true`, the component is disabled.
`disableElevation`	`bool`	`false`	If `true`, no elevation is used.
`disableFocusRipple`	`bool`	`false`	If `true`, the keyboard focus ripple is disabled.
`disableRipple`	`bool`	`false`	If `true`, the ripple effect is disabled. ⚠️ Without a ripple there is no styling for :focus-visible by default. Be sure to highlight the element by applying separate styles with the `.Mui-focusVisible` class.
`endIcon`	`node`		Element placed after the children.
`fullWidth`	`bool`	`false`	If `true`, the button will take up the full width of its container.
`href`	`string`		The URL to link to when the button is clicked. If defined, an `a` element will be used as the root node.
`size`	`union`	`'medium'`	The size of the component. `small` is equivalent to the dense button



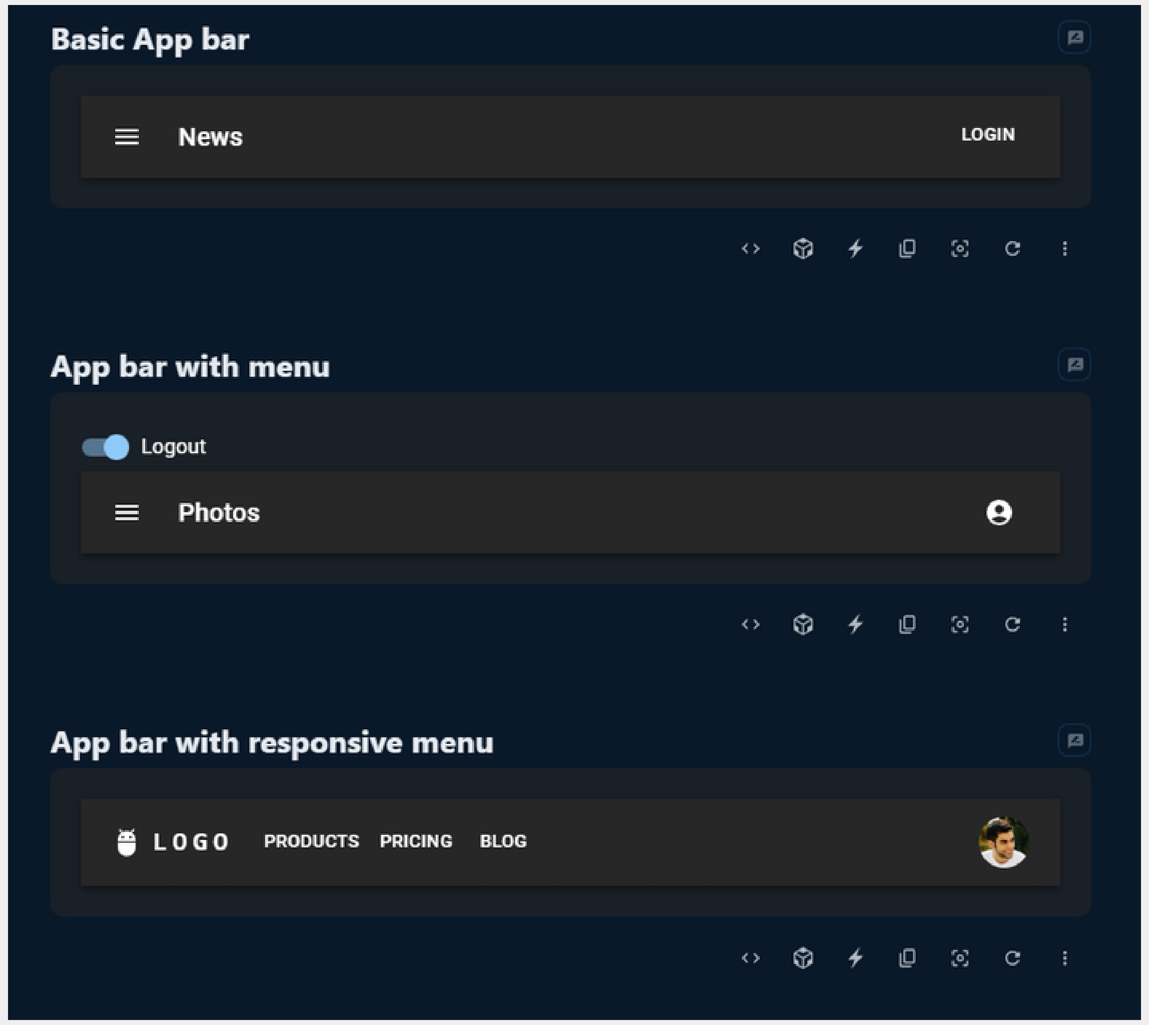
CSS

Rule name	Global class	Description
`root`	`.MuiButton-root`	Styles applied to the root element.
`text`	`.MuiButton-text`	Styles applied to the root element if `variant="text"`.
`textInherit`	`.MuiButton-textInherit`	Styles applied to the root element if `variant="text"` and `color="inherit"`.
`textPrimary`	`.MuiButton-textPrimary`	Styles applied to the root element if `variant="text"` and `color="primary"`.
`textSecondary`	`.MuiButton-textSecondary`	Styles applied to the root element if `variant="text"` and `color="secondary"`.
`textSuccess`	`.MuiButton-textSuccess`	Styles applied to the root element if `variant="text"` and `color="success"`.
`textError`	`.MuiButton-textError`	Styles applied to the root element if `variant="text"` and `color="error"`.
`textInfo`	`.MuiButton-textInfo`	Styles applied to the root element if `variant="text"` and `color="info"`.
`textWarning`	`.MuiButton-textWarning`	Styles applied to the root element if `variant="text"` and `color="warning"`.
`outlined`	`.MuiButton-outlined`	Styles applied to the root element if `variant="outlined"`.
`outlinedInherit`	`.MuiButton-outlinedInherit`	Styles applied to the root element if `variant="outlined"` and `color="inherit"`.

MUI Components

- Inputs
- Data Display
- Feedback
- Surfaces
- Navigation
- Layout
- Utils

Surface Components


Basic App bar

News LOGIN

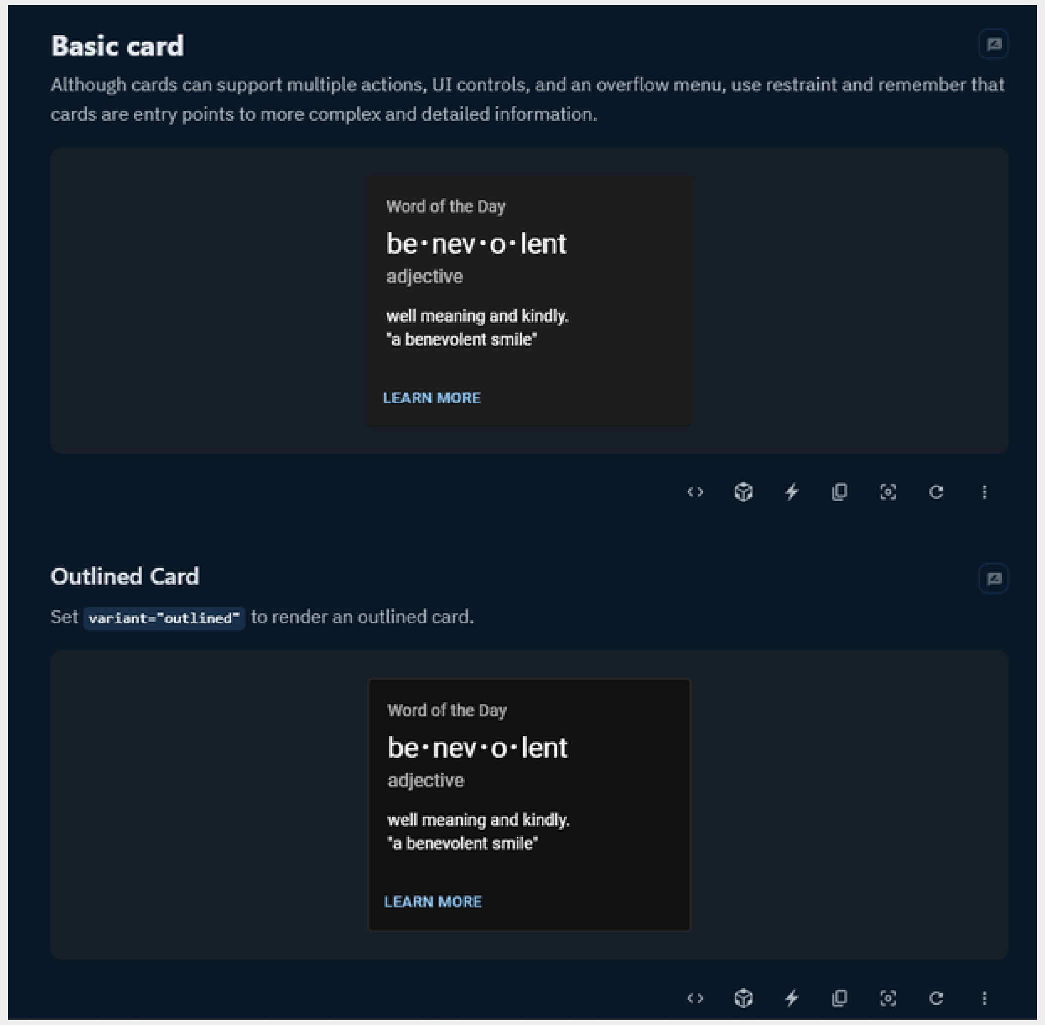
App bar with menu

Logout

Photos

App bar with responsive menu

LOGO PRODUCTS PRICING BLOG


Basic card

Although cards can support multiple actions, UI controls, and an overflow menu, use restraint and remember that cards are entry points to more complex and detailed information.

Word of the Day
be•nev•o•lent
adjective
well meaning and kindly.
"a benevolent smile"
LEARN MORE

Outlined Card

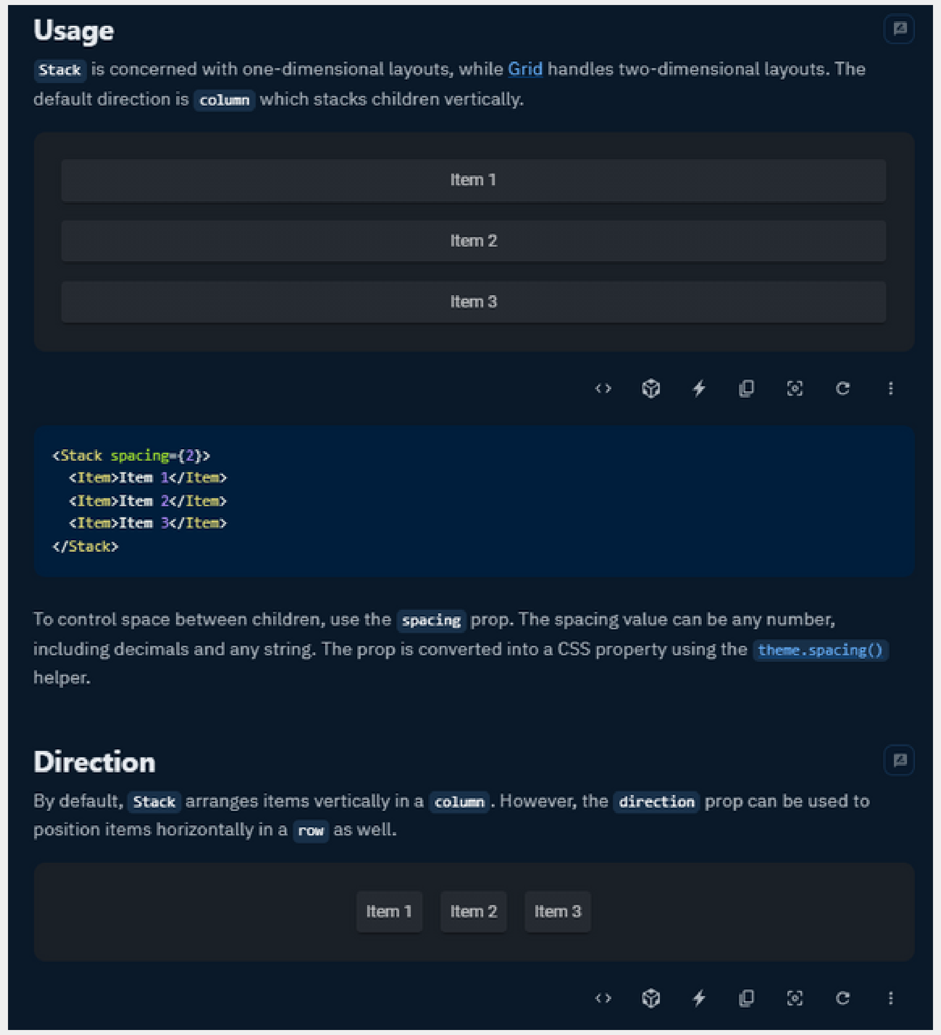
Set `variant="outlined"` to render an outlined card.

Word of the Day
be•nev•o•lent
adjective
well meaning and kindly.
"a benevolent smile"
LEARN MORE

- Accordion
- App Bar
- Card
- Paper

https://m2.material.io/design/environment/surfaces.html#material-environment

Layout Components



Usage

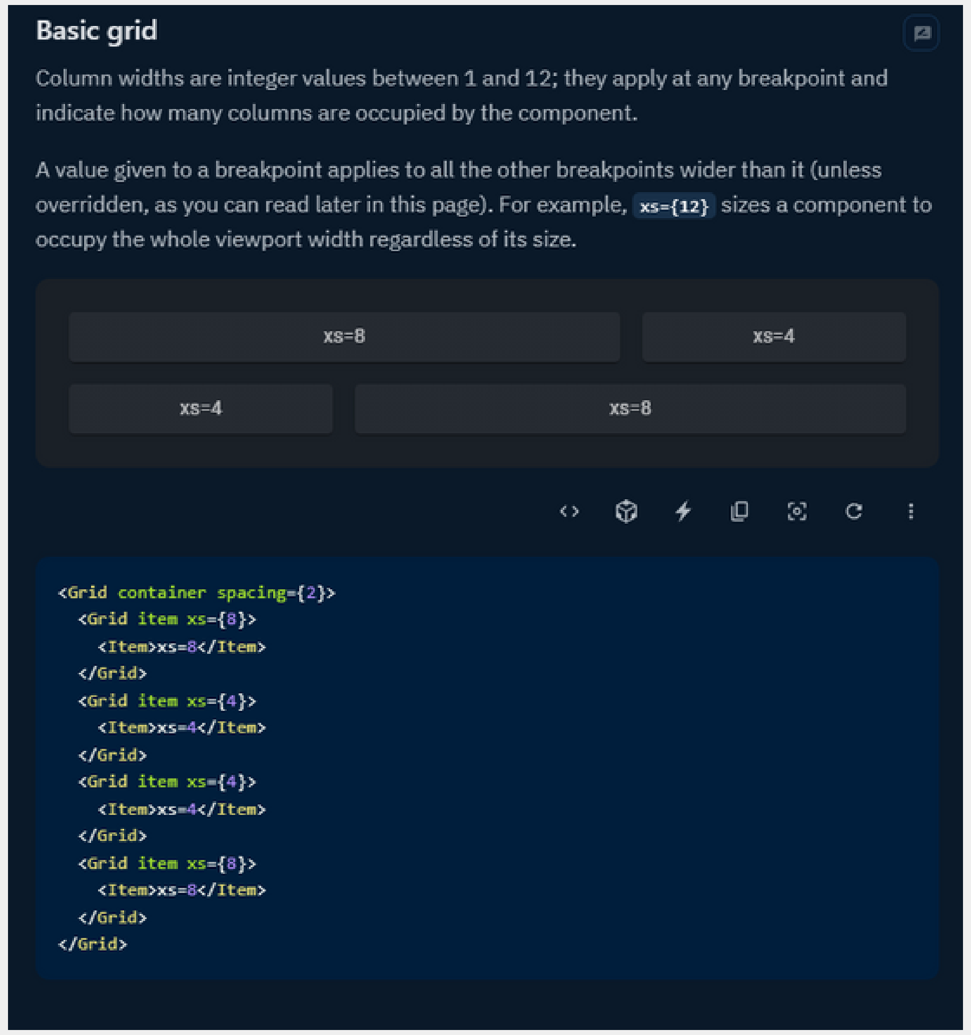
`Stack` is concerned with one-dimensional layouts, while Grid handles two-dimensional layouts. The default direction is `column` which stacks children vertically.

```
<Stack spacing={2}>
  <Item>Item 1</Item>
  <Item>Item 2</Item>
  <Item>Item 3</Item>
</Stack>
```

To control space between children, use the `spacing` prop. The spacing value can be any number, including decimals and any string. The prop is converted into a CSS property using the `theme.spacing()` helper.

Direction

By default, `Stack` arranges items vertically in a `column`. However, the `direction` prop can be used to position items horizontally in a `row` as well.



Basic grid

Column widths are integer values between 1 and 12; they apply at any breakpoint and indicate how many columns are occupied by the component.

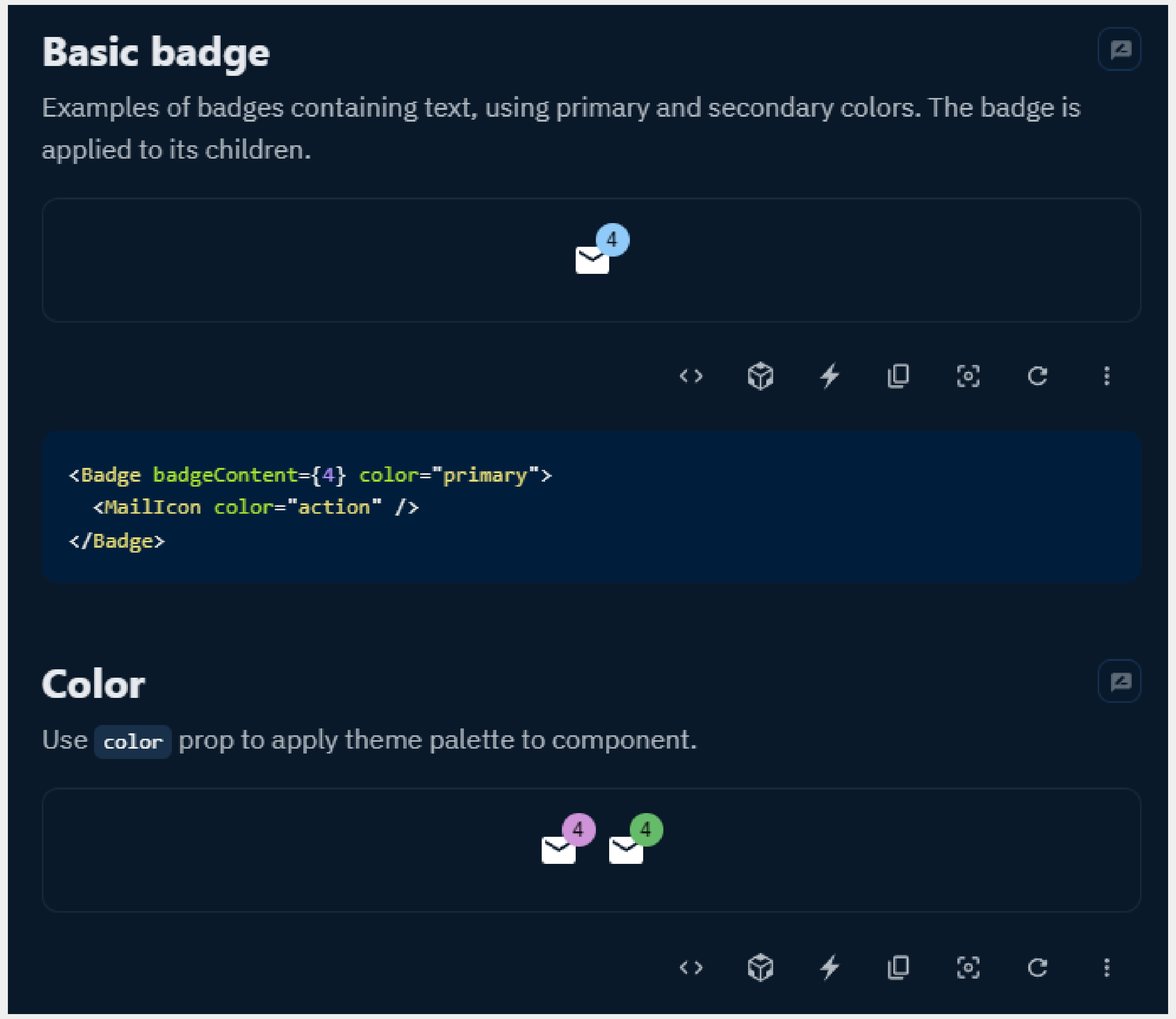
A value given to a breakpoint applies to all the other breakpoints wider than it (unless overridden, as you can read later in this page). For example, `xs={12}` sizes a component to occupy the whole viewport width regardless of its size.

```
<Grid container spacing={2}>
  <Grid item xs={8}>
    <Item>xs=8</Item>
  </Grid>
  <Grid item xs={4}>
    <Item>xs=4</Item>
  </Grid>
  <Grid item xs={4}>
    <Item>xs=4</Item>
  </Grid>
  <Grid item xs={8}>
    <Item>xs=8</Item>
  </Grid>
</Grid>
```

- Box
- Container
- Grid
- Stack
- Image List

https://m2.material.io/design/layout/understanding-layout.html#layout-anatomy

Data Display Components



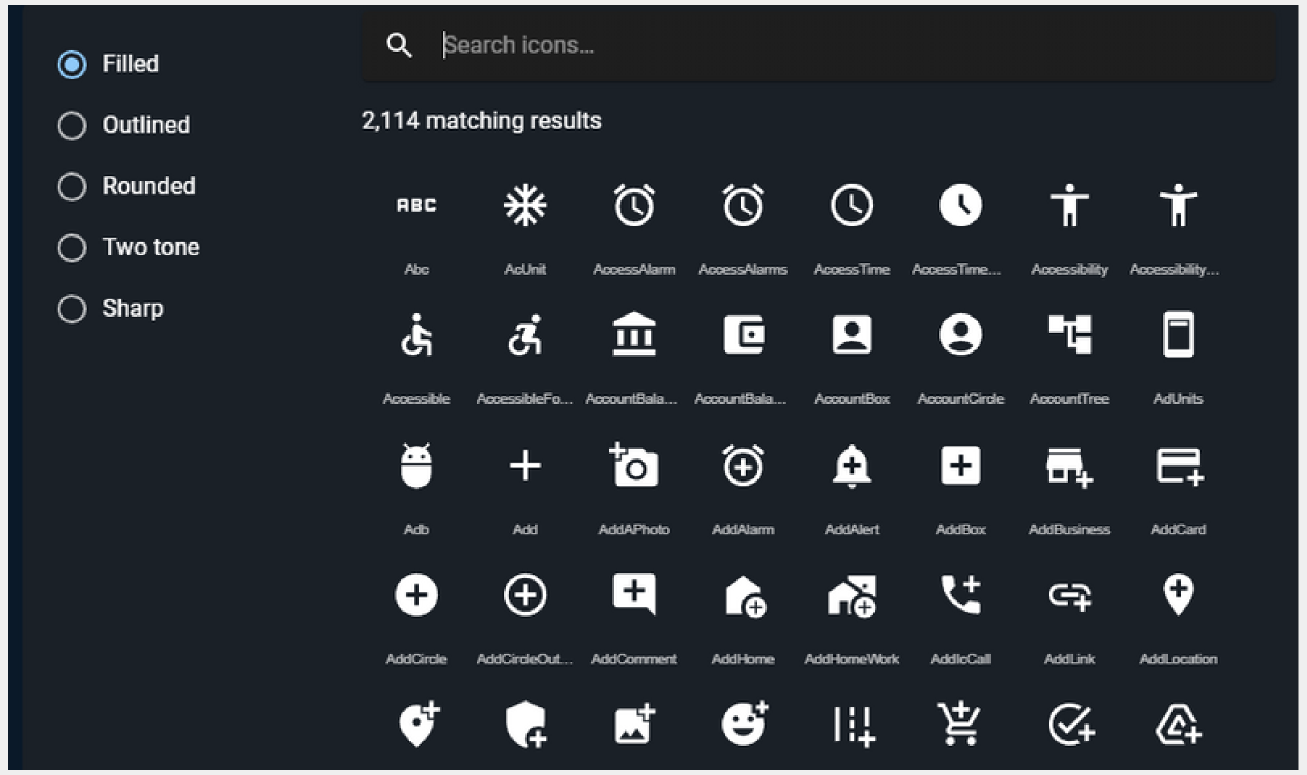
Basic badge

Examples of badges containing text, using primary and secondary colors. The badge is applied to its children.

```
<Badge badgeContent={4} color="primary">
  <MailIcon color="action" />
</Badge>
```

Color

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



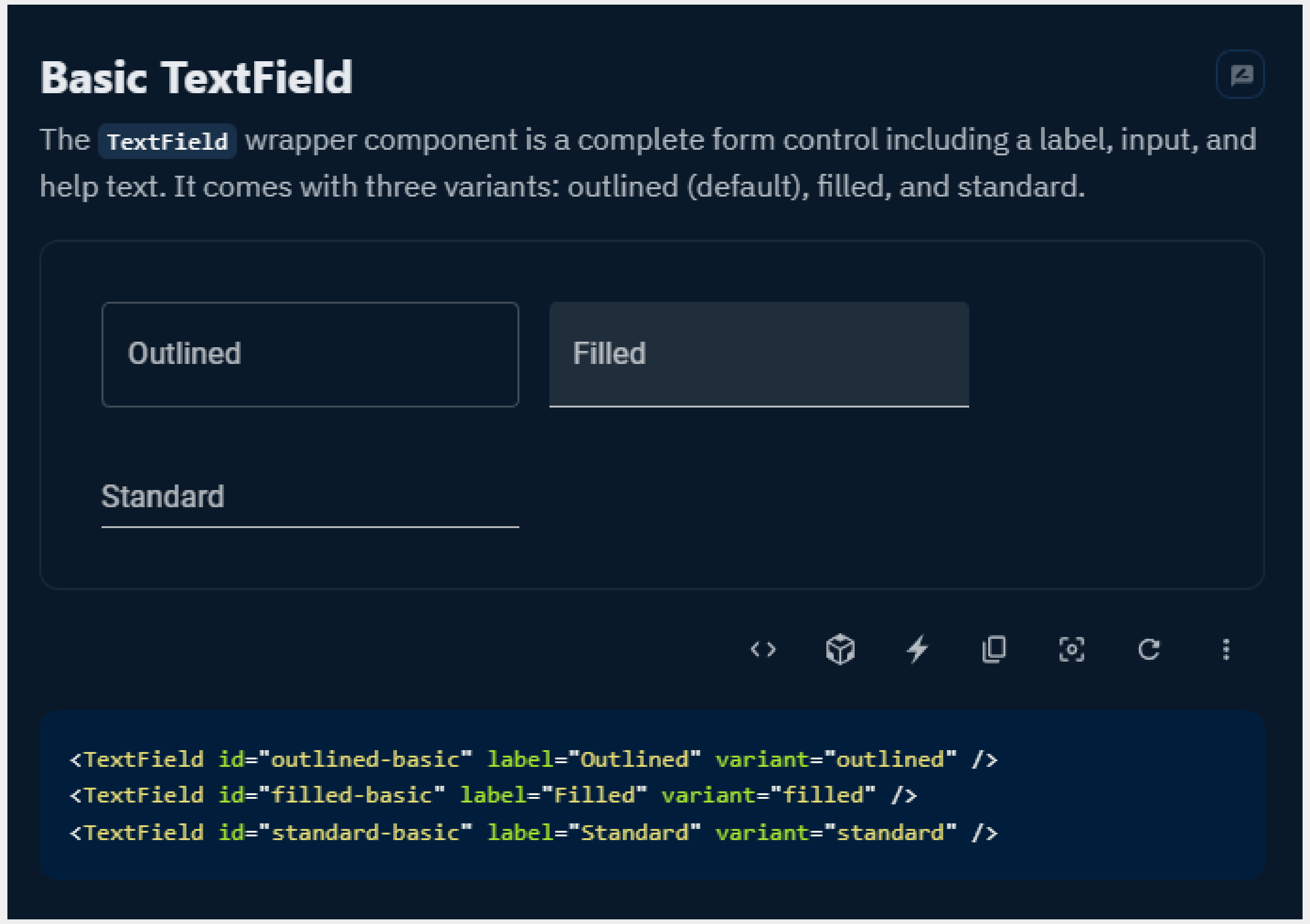
[✓ Filled] Outlined Rounded Two tone Sharp

Search icons...

2,114 matching results

- Avatar
- Badge
- Chip
- Divider
- Icons
- Material Icons
- List
- Table
- Tooltip
- Typography

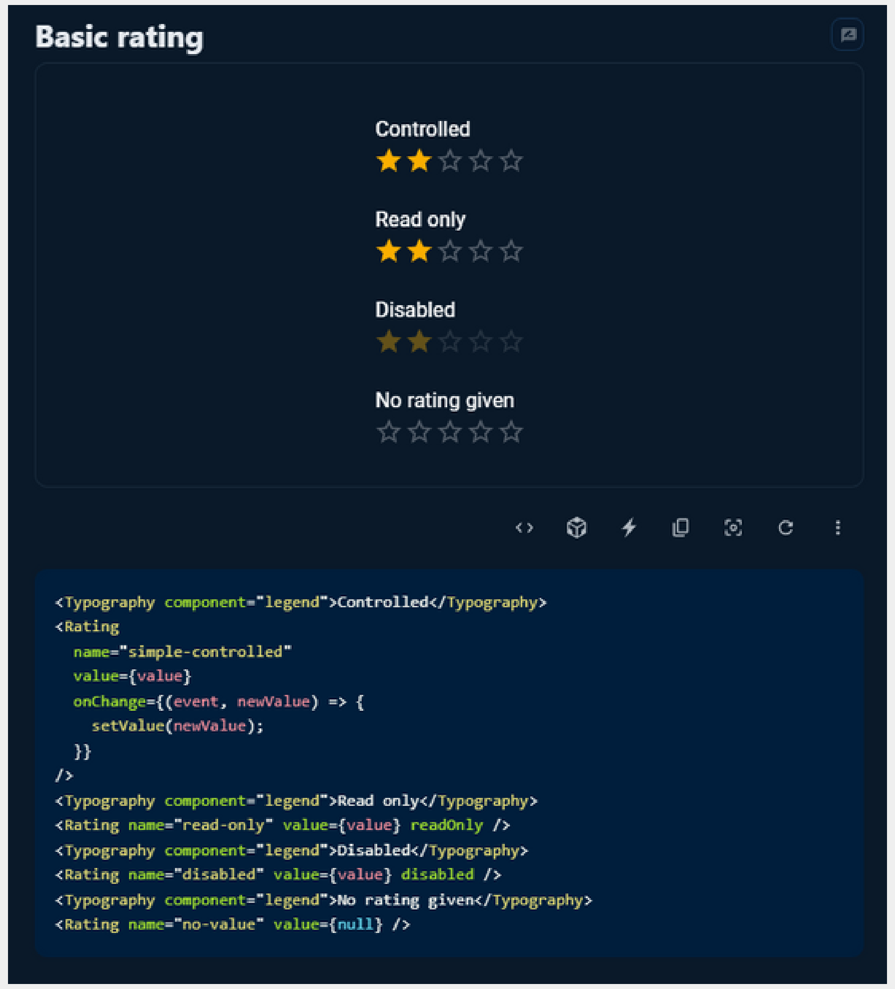
Inputs Components



Basic TextField

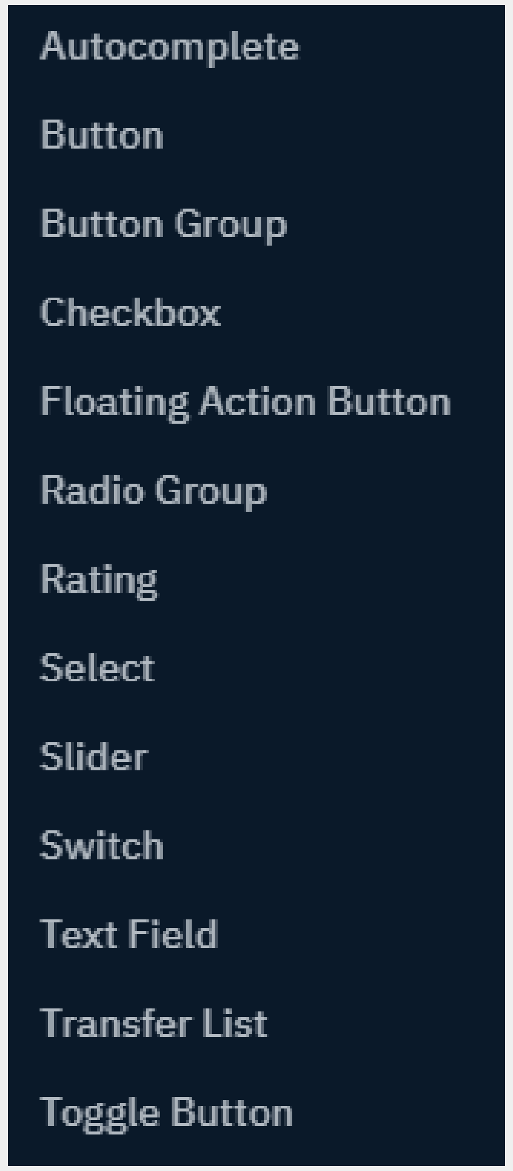
The `TextField` wrapper component is a complete form control including a label, input, and help text. It comes with three variants: outlined (default), filled, and standard.

```
<TextField id="outlined-basic" label="Outlined" variant="outlined" />
<TextField id="filled-basic" label="Filled" variant="filled" />
<TextField id="standard-basic" label="Standard" variant="standard" />
```



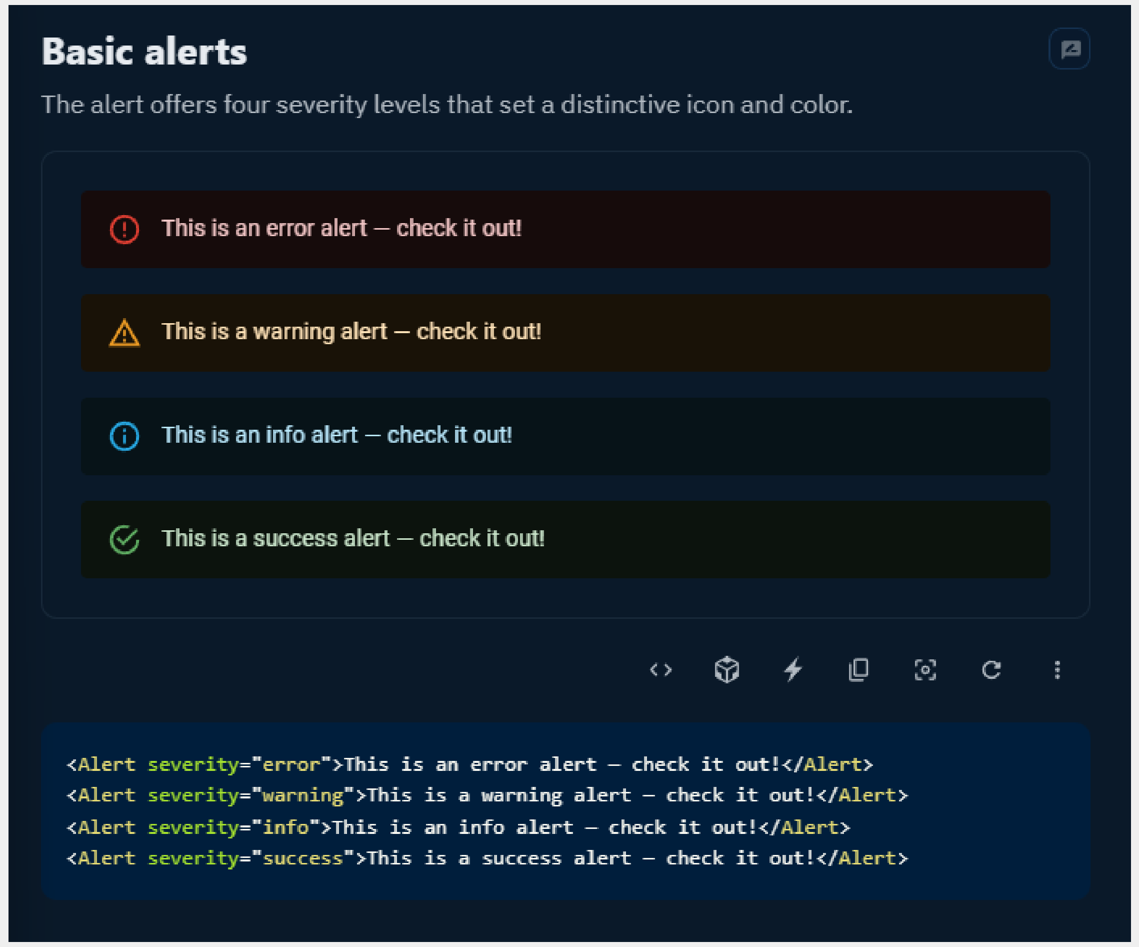
Basic rating

```
<Typography component="legend">Controlled</Typography>
<Rating
  name="simple-controlled"
  value={value}
  onChange={(event, newValue) => {
    setValue(newValue);
  }}
/>
<Typography component="legend">Read only</Typography>
<Rating name="read-only" value={value} readOnly />
<Typography component="legend">Disabled</Typography>
<Rating name="disabled" value={value} disabled />
<Typography component="legend">No rating given</Typography>
<Rating name="no-value" value={null} />
```



- Autocomplete
- Button
- Button Group
- Checkbox
- Floating Action Button
- Radio Group
- Rating
- Select
- Slider
- Switch
- Text Field
- Transfer List
- Toggle Button

Feedback Components


Basic alerts

The alert offers four severity levels that set a distinctive icon and color.

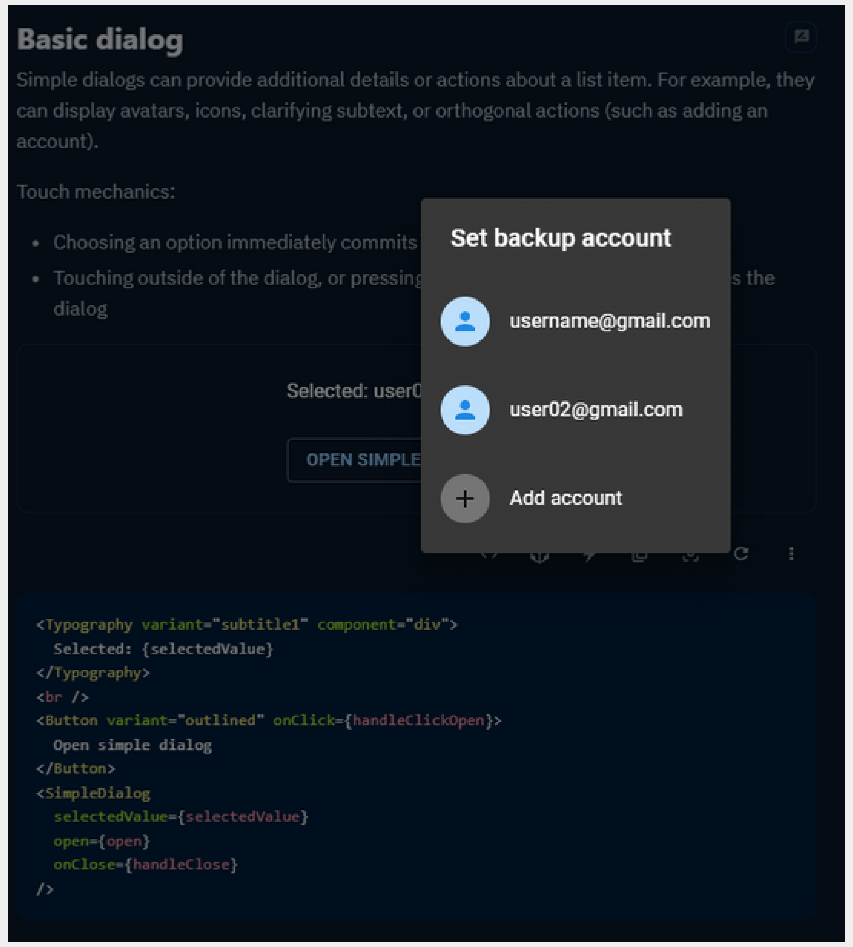
This is an error alert — check it out!

This is a warning alert — check it out!

This is an info alert — check it out!

This is a success alert — check it out!

```
<Alert severity="error">This is an error alert — check it out!</Alert>
<Alert severity="warning">This is a warning alert — check it out!</Alert>
<Alert severity="info">This is an info alert — check it out!</Alert>
<Alert severity="success">This is a success alert — check it out!</Alert>
```


Basic dialog

Simple dialogs can provide additional details or actions about a list item. For example, they can display avatars, icons, clarifying subtext, or orthogonal actions (such as adding an account).

Touch mechanics:

- Choosing an option immediately commits
- Touching outside of the dialog, or pressing s the dialog

Selected: user0

```
<Typography variant="subtitle1" component="div">
  Selected: {selectedValue}
</Typography>
<br />
<Button variant="outlined" onClick={handleClickOpen}>
  Open simple dialog
</Button>
<SimpleDialog
  selectedValue={selectedValue}
  open={open}
  onClose={handleClose}
/>
```

Alert

Backdrop

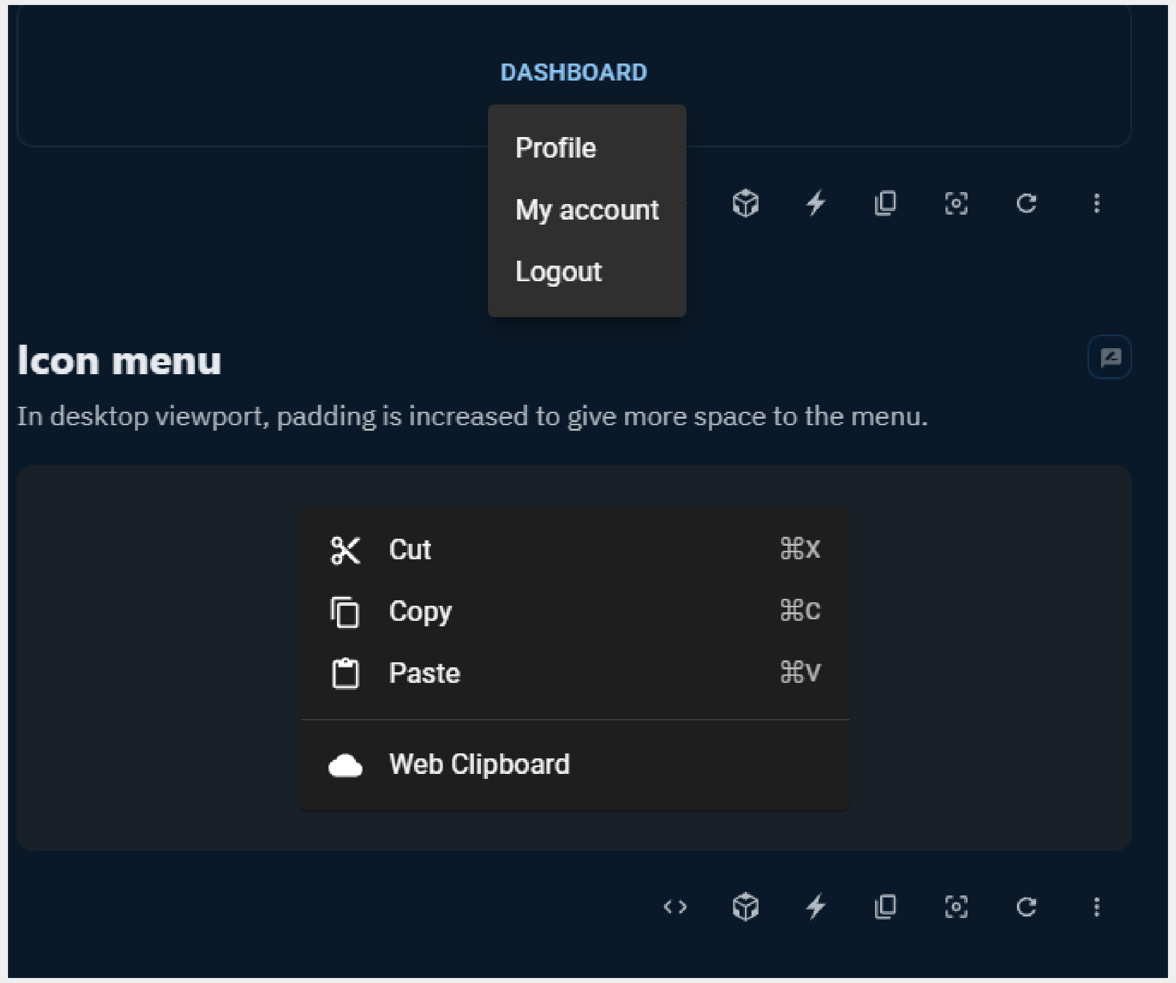
Dialog

Progress

Skeleton

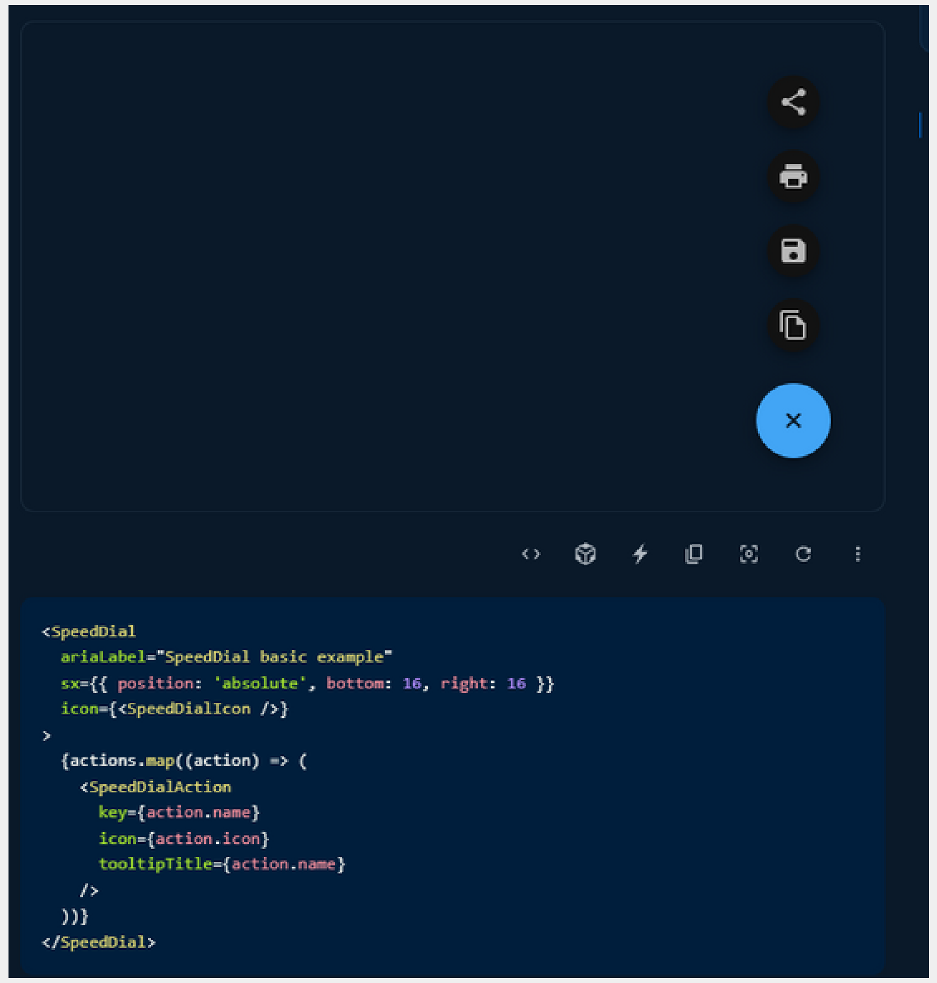
Snackbar

Navigation Components

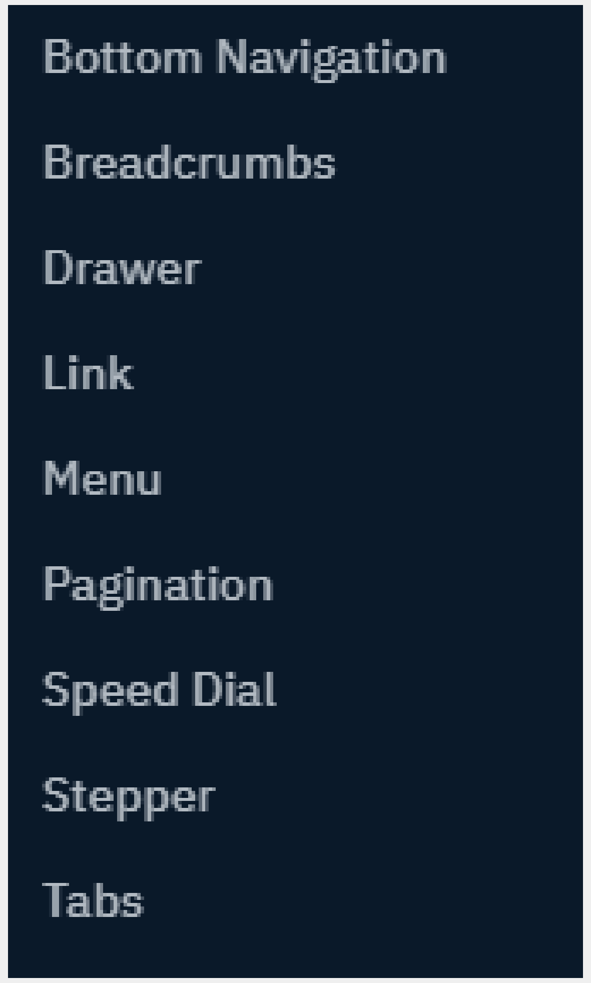


Icon menu

In desktop viewport, padding is increased to give more space to the menu.

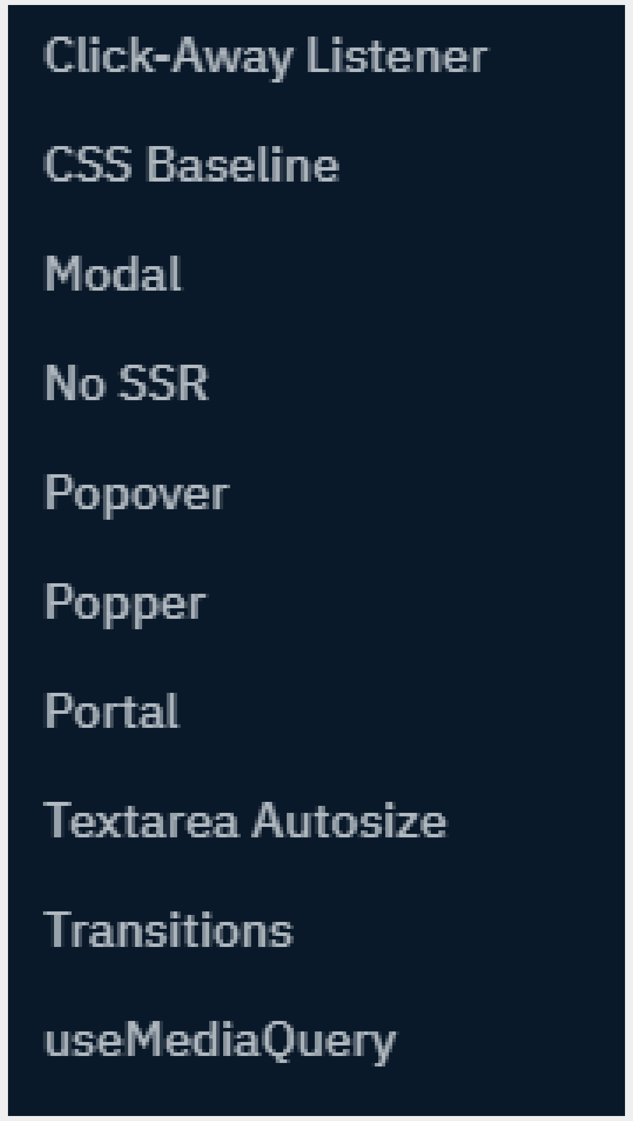


```
<SpeedDial
  ariaLabel="SpeedDial basic example"
  sx={{ position: 'absolute', bottom: 16, right: 16 }}
  icon={<SpeedDialIcon />}
>
  {actions.map((action) => (
    <SpeedDialAction
      key={action.name}
      icon={action.icon}
      tooltipTitle={action.name}
    />
  ))}
</SpeedDial>
```



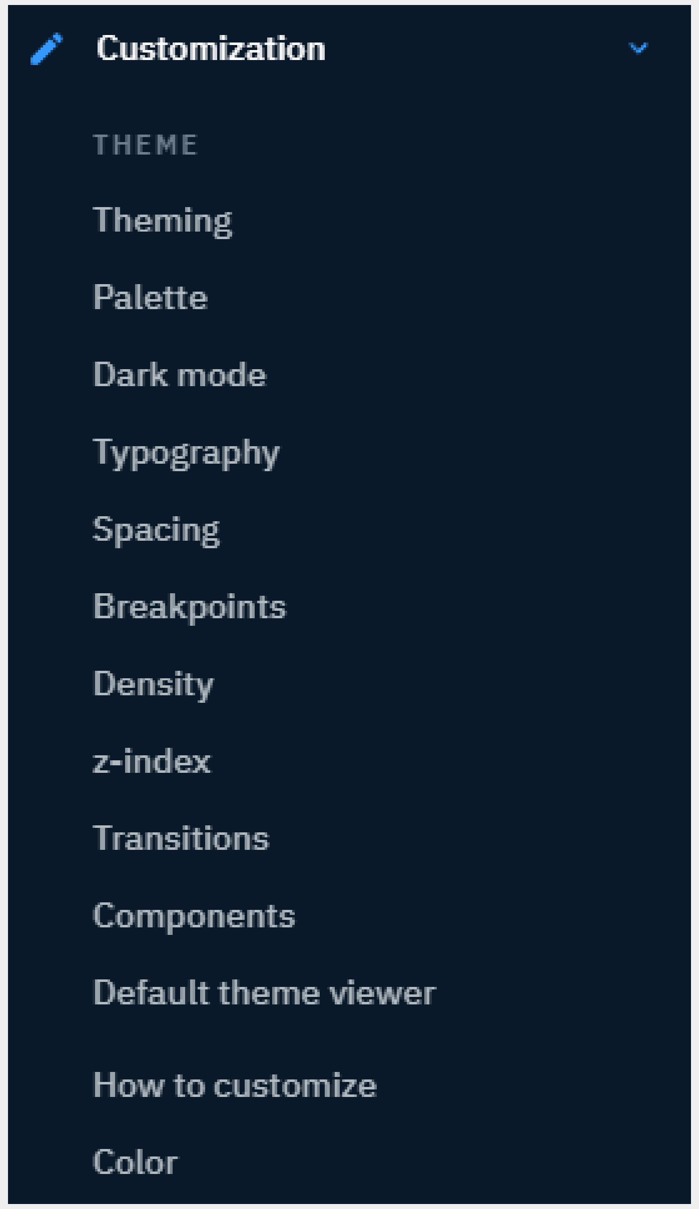
Bottom Navigation
Breadcrumbs
Drawer
Link
Menu
Pagination
Speed Dial
Stepper
Tabs

Utils Components



Click-Away Listener
CSS Baseline
Modal
No SSR
Popover
Popper
Portal
Textarea Autosize
Transitions
useMediaQuery

Component Customization



Customization

THEME

Theming

Palette

Dark mode

Typography

Spacing

Breakpoints

Density

z-index

Transitions

Components

Default theme viewer

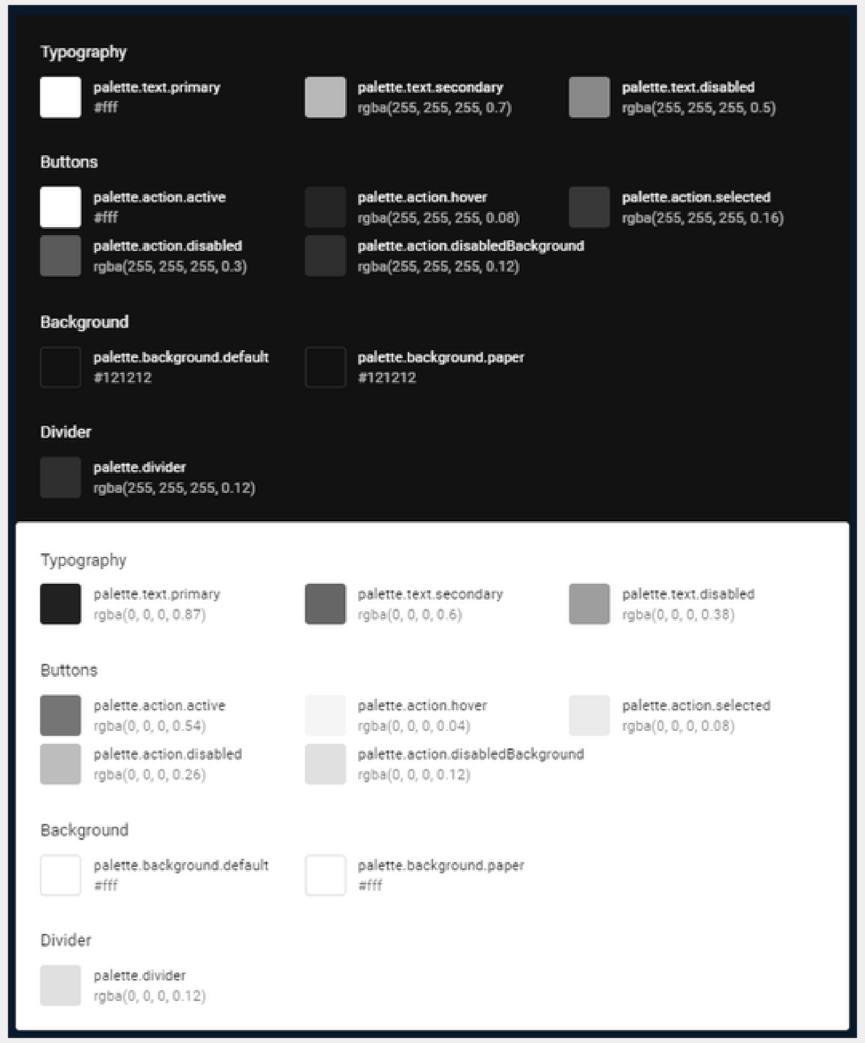
How to customize

Color

Theming

- palette
- typography
- spacing
- breakpoints
- zIndex
- transitions
- components

Dark mode



Typography

palette.text.primary
#fff

palette.text.secondary
rgba(255, 255, 255, 0.7)

palette.text.disabled
rgba(255, 255, 255, 0.5)

Buttons

palette.action.active
#fff

palette.action.hover
rgba(255, 255, 255, 0.08)

palette.action.selected
rgba(255, 255, 255, 0.16)

palette.action.disabled
rgba(255, 255, 255, 0.3)

palette.action.disabledBackground
rgba(255, 255, 255, 0.12)

Background

palette.background.default
#121212

palette.background.paper
#121212

Divider

palette.divider
rgba(255, 255, 255, 0.12)

Typography

palette.text.primary
rgba(0, 0, 0, 0.87)

palette.text.secondary
rgba(0, 0, 0, 0.6)

palette.text.disabled
rgba(0, 0, 0, 0.38)

Buttons

palette.action.active
rgba(0, 0, 0, 0.54)

palette.action.hover
rgba(0, 0, 0, 0.04)

palette.action.selected
rgba(0, 0, 0, 0.08)

palette.action.disabled
rgba(0, 0, 0, 0.26)

palette.action.disabledBackground
rgba(0, 0, 0, 0.12)

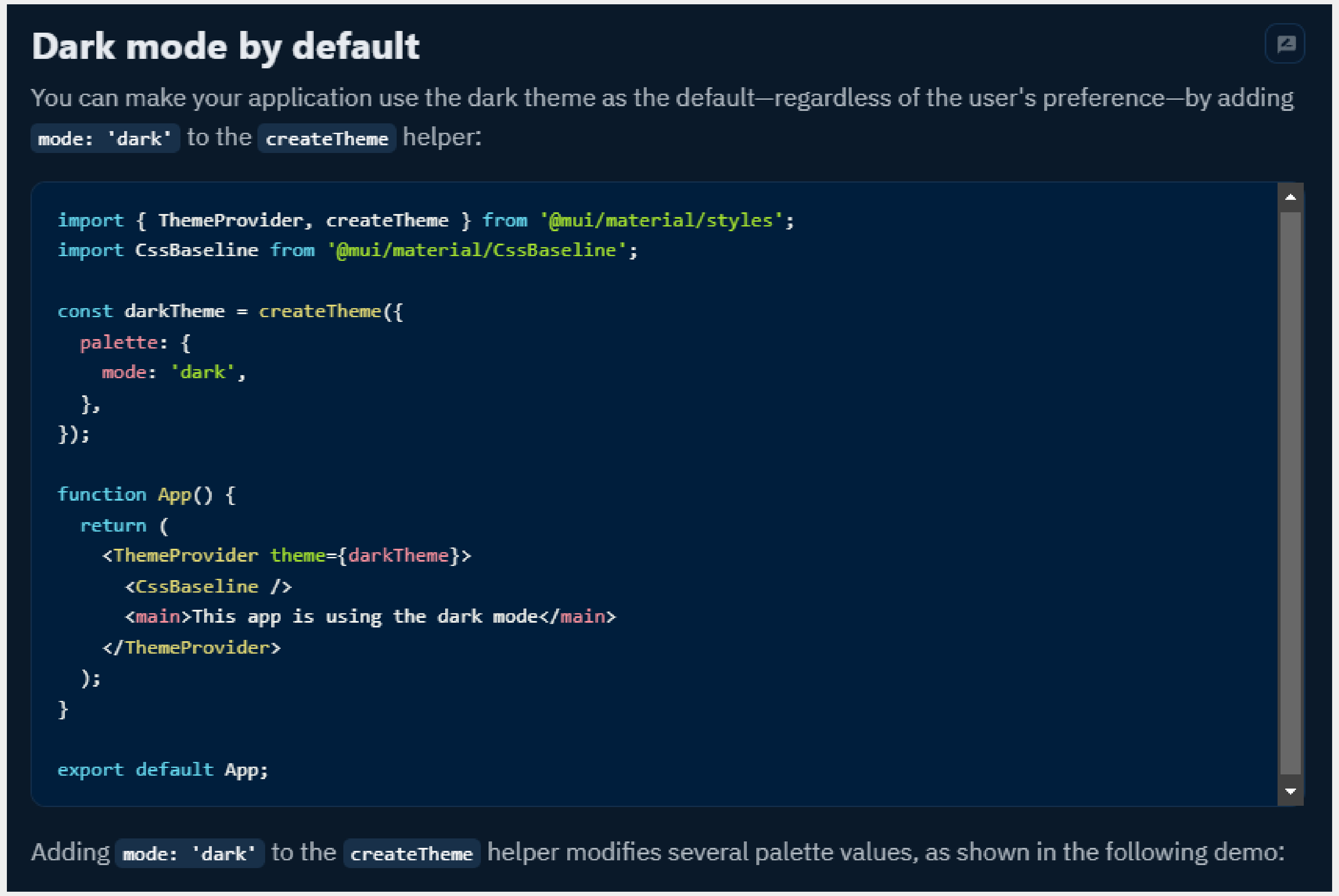
Background

palette.background.default
#fff

palette.background.paper
#fff

Divider

palette.divider
rgba(0, 0, 0, 0.12)



Dark mode by default

You can make your application use the dark theme as the default—regardless of the user's preference—by adding `mode: 'dark'` to the `createTheme` helper:

```jsx
import { ThemeProvider, createTheme } from '@mui/material/styles';
import CssBaseline from '@mui/material/CssBaseline';

const darkTheme = createTheme({
  palette: {
    mode: 'dark',
  },
});

function App() {
  return (
    <ThemeProvider theme={darkTheme}>
      <CssBaseline />
      <main>This app is using the dark mode</main>
    </ThemeProvider>
  );
}

export default App;
```

Adding `mode: 'dark'` to the `createTheme` helper modifies several palette values, as shown in the following demo:

https://mui.com/material-ui/customization/dark-mode/

How to customize

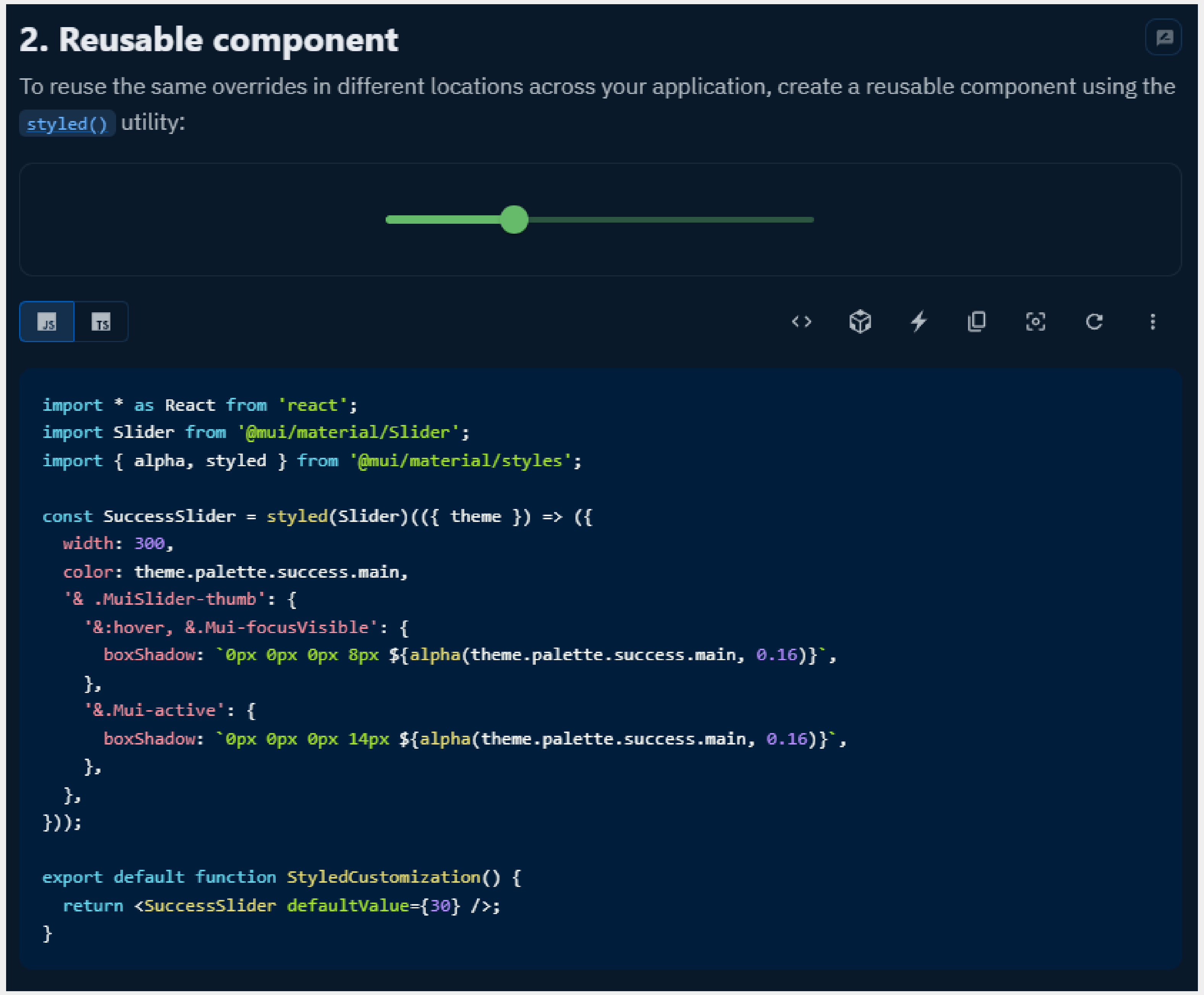
1. One-off customization
2. Reusable component
3. Global theme overrides
4. Global CSS override

1. One-off customization

- sx prop
  - Overriding nested component styles
- Overriding styles with class names
- State classes
  - Custom state classes

styling with class names
https://mui.com/material-ui/guides/interoperability/
Custom state classes
https://mui.com/system/styles/advanced/#class-names

2. Reusable component



2. Reusable component

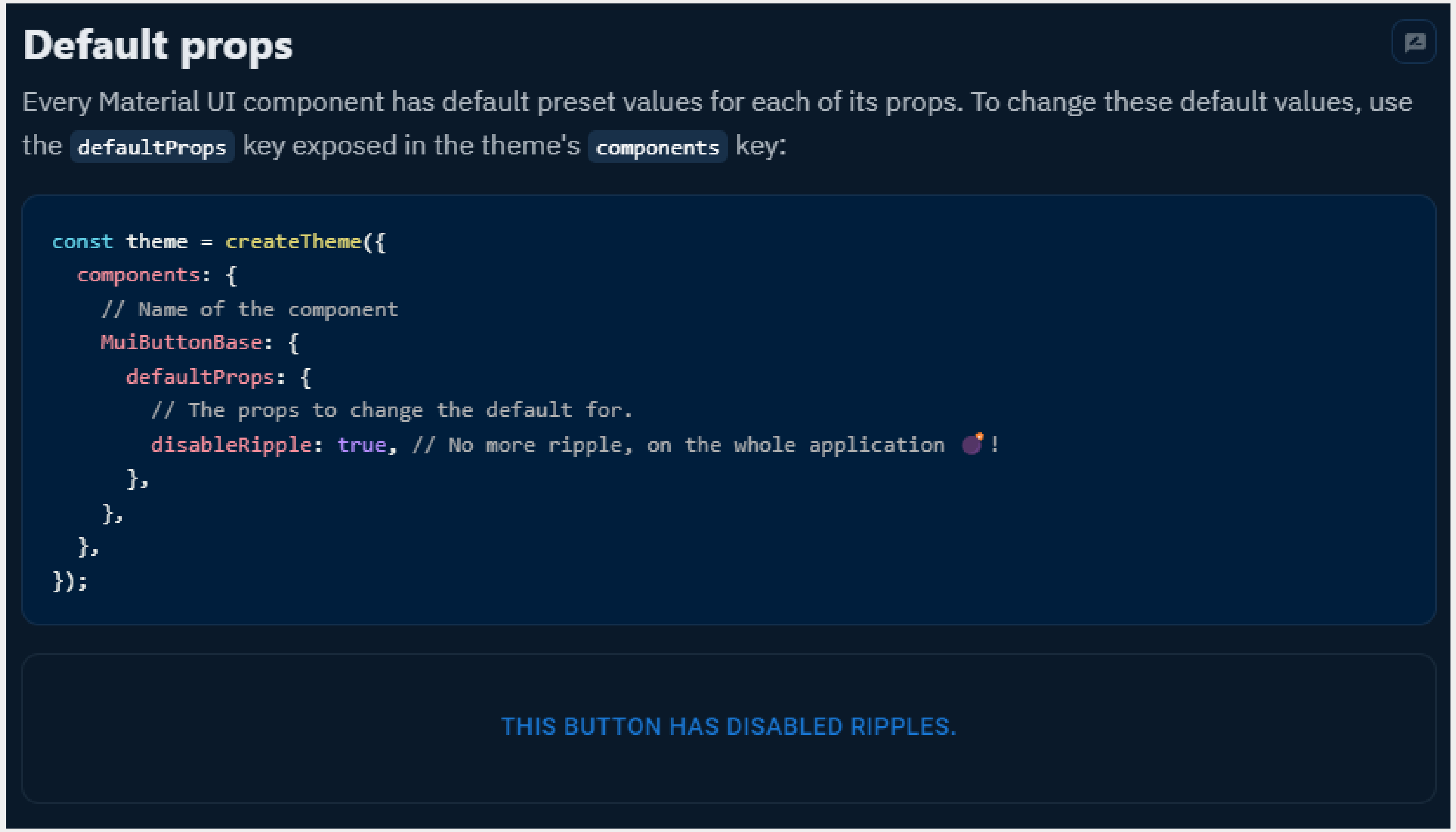
To reuse the same overrides in different locations across your application, create a reusable component using the `styled()` utility:

```
import * as React from 'react';
import Slider from '@mui/material/Slider';
import { alpha, styled } from '@mui/material/styles';

const SuccessSlider = styled(Slider)(({ theme }) => ({
  width: 300,
  color: theme.palette.success.main,
  '& .MuiSlider-thumb': {
    '&:hover, &.Mui-focusVisible': {
      boxShadow: `0px 0px 0px 8px ${alpha(theme.palette.success.main, 0.16)}`,
    },
    '&.Mui-active': {
      boxShadow: `0px 0px 0px 14px ${alpha(theme.palette.success.main, 0.16)}`,
    },
  },
}));

export default function StyledCustomization() {
  return <SuccessSlider defaultValue={30} />;
}
```

3. Global theme overrides

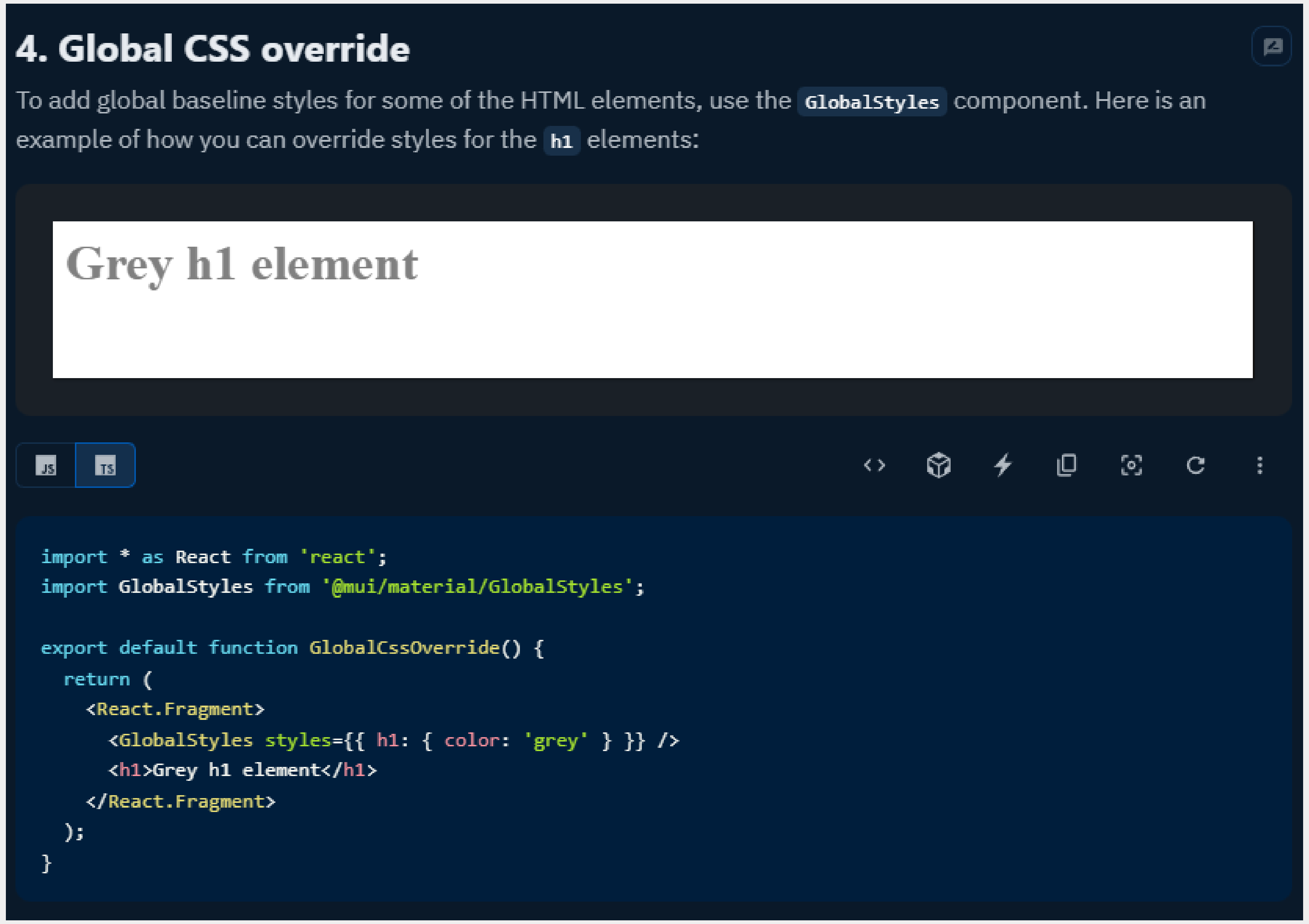


Default props

Every Material UI component has default preset values for each of its props. To change these default values, use the `defaultProps` key exposed in the theme's `components` key:

```
const theme = createTheme({
  components: {
    // Name of the component
    MuiButtonBase: {
      defaultProps: {
        // The props to change the default for.
        disableRipple: true, // No more ripple, on the whole application 💣!
      },
    },
  },
});
```

4. Global CSS override



4. Global CSS override

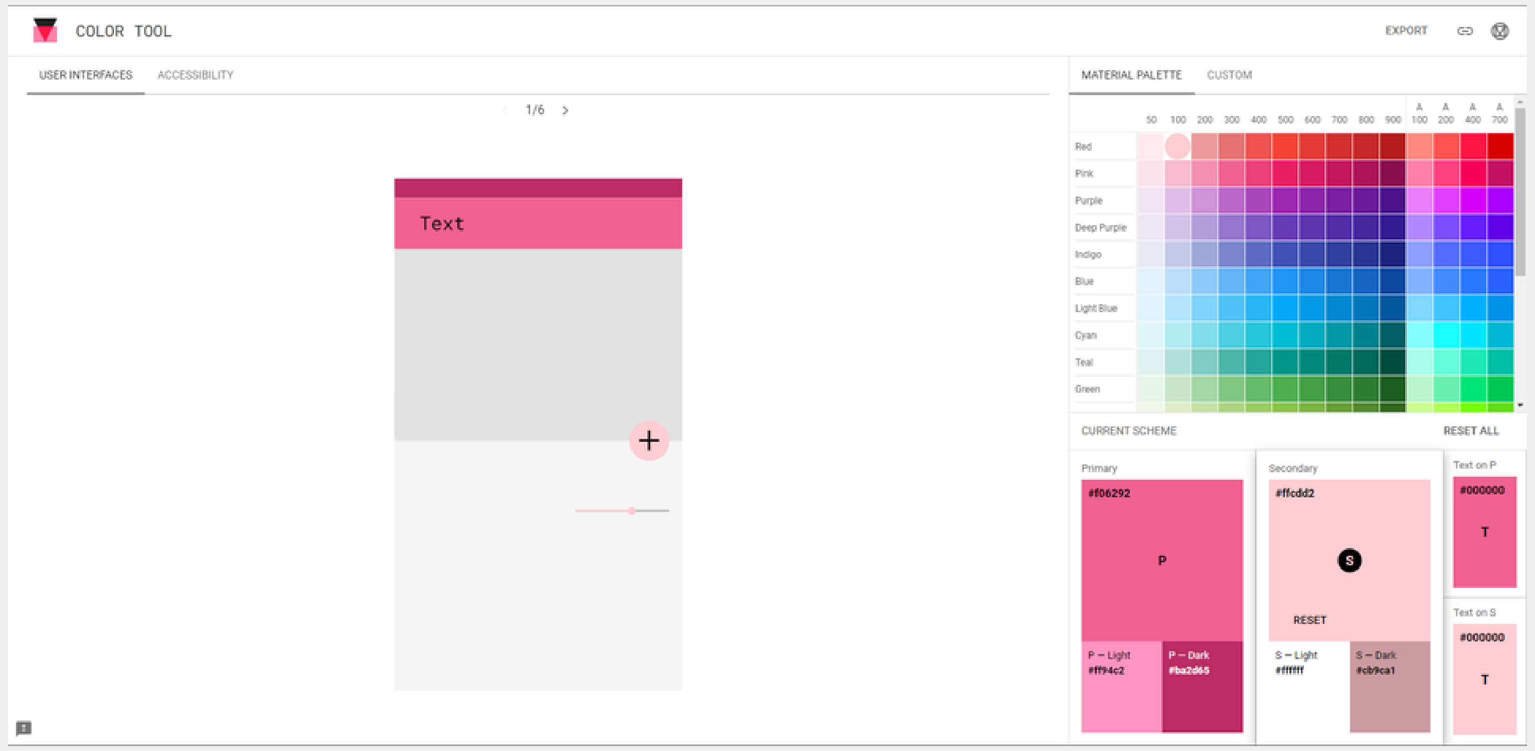
To add global baseline styles for some of the HTML elements, use the `GlobalStyles` component. Here is an example of how you can override styles for the `h1` elements:

Grey h1 element

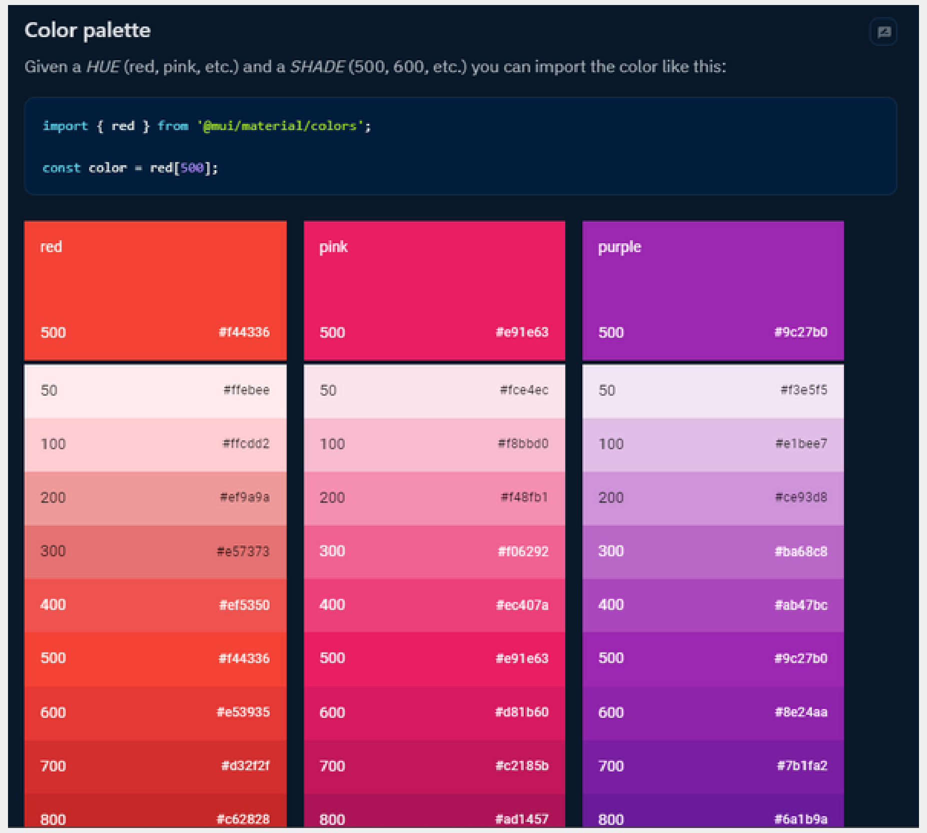
```
import * as React from 'react';
import GlobalStyles from '@mui/material/GlobalStyles';

export default function GlobalCssOverride() {
  return (
    <React.Fragment>
      <GlobalStyles styles={{ h1: { color: 'grey' } }} />
      <h1>Grey h1 element</h1>
    </React.Fragment>
  );
}
```

Color



Primary #f06292
P – Light #ff94c2
P – Dark #ba2d65
Secondary #ffcdd2
S – Light #ffffff
S – Dark #cb9ca1
Text on P #000000
Text on S #000000



Color palette

Given a *HUE* (red, pink, etc.) and a *SHADE* (500, 600, etc.) you can import the color like this:

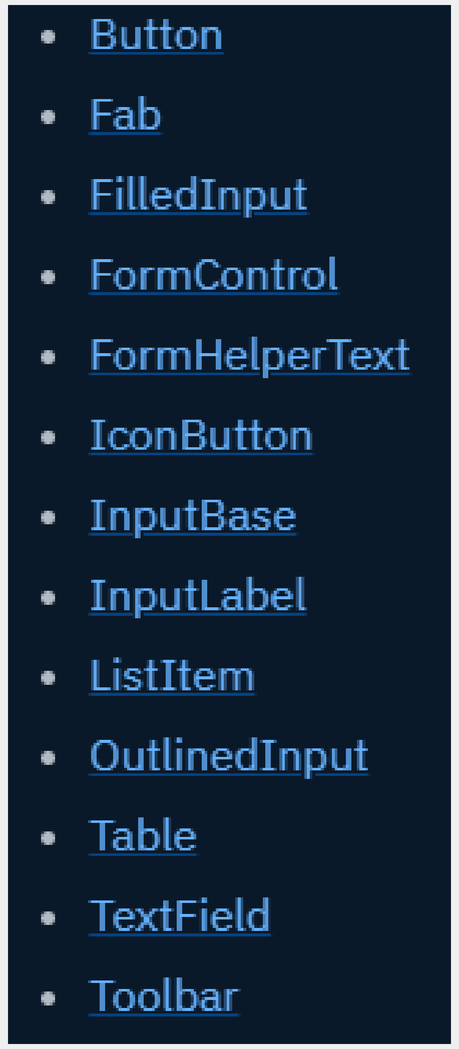
```
import { red } from '@mui/material/colors';

const color = red[500];
```

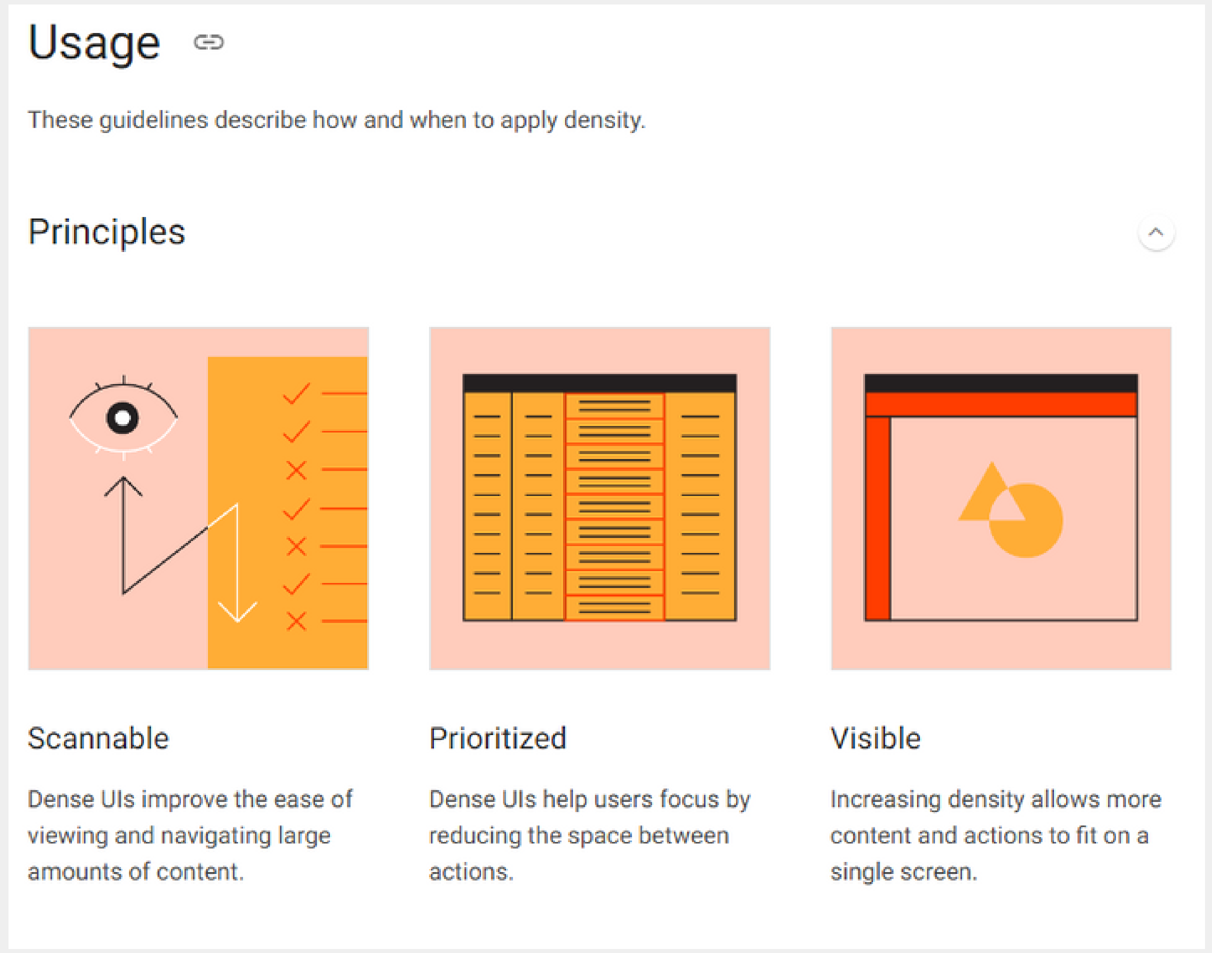
red		pink		purple	
500	#f44336	500	#e91e63	500	#9c27b0
50	#ffebee	50	#fce4ec	50	#f3e5f5
100	#ffcdd2	100	#f8bbd0	100	#e1bee7
200	#ef9a9a	200	#f48fb1	200	#ce93d8
300	#e57373	300	#f06292	300	#ba68c8
400	#ef5350	400	#ec407a	400	#ab47bc
500	#f44336	500	#e91e63	500	#9c27b0
600	#e53935	600	#d81b60	600	#8e24aa
700	#d32f2f	700	#c2185b	700	#7b1fa2
800	#c62828	800	#ad1457	800	#6a1b9a

https://m2.material.io/resources/color/
https://mui.com/material-ui/customization/color/

Density



- Button
- Fab
- FilledInput
- FormControl
- FormHelperText
- IconButton
- InputBase
- InputLabel
- ListItem
- OutlinedInput
- Table
- TextField
- Toolbar



Usage

These guidelines describe how and when to apply density.

Principles

Scannable

Dense UIs improve the ease of viewing and navigating large amounts of content.

Prioritized

Dense UIs help users focus by reducing the space between actions.

Visible

Increasing density allows more content and actions to fit on a single screen.

https://m2.material.io/design/layout/applying-density.html#usage
https://mui.com/material-ui/customization/density/