Palette

The palette enables you to modify the color of the components to suit your brand.

Palette colors

The theme exposes the following palette colors (accessible under `theme.palette.`):

- *primary* - used to represent primary interface elements for a user. It's the color displayed most frequently across your app's screens and components.
- *secondary* - used to represent secondary interface elements for a user. It provides more ways to accent and distinguish your product. Having it is optional.
- *error* - used to represent interface elements that the user should be made aware of.
- *warning* - used to represent potentially dangerous actions or important messages.
- *info* - used to present information to the user that is neutral and not necessarily important.
- *success* - used to indicate the successful completion of an action that user triggered.

If you want to learn more about color, you can check out the color section.

Default values

You can explore the default values of the palette using the theme explorer or by opening the dev tools console on this page (`window.theme.palette`).

{{"demo": "Intentions.js", "bg": "inline", "hideToolbar": true}}

The default palette uses the shades prefixed with `A` (`A200` , etc.) for the secondary palette color, and the un-prefixed shades for the other palette colors.

Customization

You may override the default palette values by including a palette object as part of your theme. If any of the:

- `.palette.primary`
- `.palette.secondary`
- `.palette.error`
- `.palette.warning`
- `.palette.info`
- `.palette.success`

palette color objects are provided, they will replace the default ones.

The palette color value can either be a color object, or an object with one or more of the keys specified by the following TypeScript interface:

```
interface PaletteColor {
  light?: string;
  main: string;
  dark?: string;
  contrastText?: string;
}
```

Using a color object

The simplest way to customize a palette color is to import one or more of the provided colors and apply them:

```
import { createTheme } from '@mui/material/styles';
import blue from '@mui/material/colors/blue';

const theme = createTheme({
  palette: {
    primary: blue,
  },
});
```

Providing the colors directly

If you wish to provide more customized colors, you can either create your own palette color, or directly supply colors to some or all of the `theme.palette` keys:

```
import { createTheme } from '@mui/material/styles';

const theme = createTheme({
  palette: {
    primary: {
      // light: will be calculated from palette.primary.main,
      main: '#ff4400',
      // dark: will be calculated from palette.primary.main,
      // contrastText: will be calculated to contrast with palette.primary.main
    },
    secondary: {
      light: '#0066ff',
      main: '#0044ff',
      // dark: will be calculated from palette.secondary.main,
      contrastText: '#ffcc00',
    },
    // Used by `getContrastText()` to maximize the contrast between
    // the background and the text.
    contrastThreshold: 3,
    // Used by the functions below to shift a color's luminance by approximately
    // two indexes within its tonal palette.
    // E.g., shift from Red 500 to Red 300 or Red 700.
    tonalOffset: 0.2,
  },
});
```

As in the example above, if the palette color contains custom colors using any of the "main", "light", "dark" or "contrastText" keys, these map as follows:

- If the "dark" and / or "light" keys are omitted, their value(s) will be calculated from "main", according to the "tonalOffset" value.
- If "contrastText" is omitted, its value will be calculated to contrast with "main", according to the "contrastThreshold" value.

Both the "tonalOffset" and "contrastThreshold" values may be customized as needed. The "tonalOffset" value can either be a number between 0 and 1, which will apply to both light and dark variants, or an object with light and dark variants specified by the following TypeScript type:

```
type PaletteTonalOffset =
  | number
  | {
      light: number;
      dark: number;
    };
```

A higher value for "tonalOffset" will make calculated values for "light" lighter, and "dark" darker. A higher value for "contrastThreshold" increases the point at which a background color is considered light, and given a dark "contrastText".

Note that "contrastThreshold" follows a non-linear curve.

Example

{{"demo": "Palette.js", "defaultCodeOpen": true}}

Adding new colors

You can add new colors inside and outside the palette of the theme as follows:

```
import { createTheme } from '@mui/material/styles';

const theme = createTheme({
  status: {
    danger: '#e53e3e',
  },
  palette: {
    primary: {
      main: '#0971f1',
      darker: '#053e85',
    },
    neutral: {
      main: '#64748B',
      contrastText: '#fff',
    },
  },
});
```

If you are using TypeScript, you would also need to use module augmentation for the theme to accept the above values.

```
declare module '@mui/material/styles' {
  interface Theme {
    status: {
      danger: React.CSSProperties['color'];
    };
```

```
  }

  interface Palette {
    neutral: Palette['primary'];
  }
  interface PaletteOptions {
    neutral: PaletteOptions['primary'];
  }

  interface PaletteColor {
    darker?: string;
  }
  interface SimplePaletteColorOptions {
    darker?: string;
  }
  interface ThemeOptions {
    status: {
      danger: React.CSSProperties['color'];
    };
  }
}
```

{{"demo": "CustomColor.js"}}

Picking colors

Need inspiration? The Material Design team has built an palette configuration tool to help you.

Dark mode

For details of how you can set up a dark mode for your theme, head to the dark mode guide.