Temas

Customize Material-UI com seu tema. Você pode mudar as cores, a tipografia e muito mais.

O tema especifica a cor dos componentes, o escurecimento das superfícies, o nível de sombra, a opacidade apropriada dos elementos de tinta, etc.

Temas permitem que você aplique um tom consistente na sua aplicação. Ele permite que você **customize todos os aspectos do design** do seu projeto, para atender as necessidades específicas do seu negócio ou marca.

Para promover uma maior consistência entre os aplicativos, os temas claro e escuro estão disponíveis para escolha. Por padrão, os componentes usam o tema claro.

Provedor de Temas

Se você deseja personalizar o tema, você precisa usar o `ThemeProvider` componente para injetar um tema em sua aplicação. No entanto, isso é opcional; Material-UI componentes vêm com um tema padrão.

O `ThemeProvider` depende do recurso de contexto do React afim de passar o tema para baixo na árvore de componentes, então você precisa ter certeza de que o `ThemeProvider` é um pai dos componentes que você está tentando customizar. Você pode aprender mais sobre isso lendo a sessão da API.

Variáveis de configuração do tema

Alterar as variáveis de configuração do tema é a maneira mais eficaz de combinar o Material-UI às suas necessidades. As seções a seguir abordam as variáveis mais importantes do tema:

- `.paleta`
- `.typography`
- `.espaçamento`
- `.pontos de quebra`
- `.zIndex`
- `.transições`
- `.componentes`

Você pode conferir a seção de tema padrão para visualizar o tema padrão na íntegra.

Variáveis customizáveis

When using MUI's theme with MUI System or any other styling solution, it can be convenient to add additional variables to the theme so you can use them everywhere. Por exemplo:

```
const theme = createTheme({
  status: {
    danger: orange[500],
  },
});
```

Se você estiver usando TypeScript, você também deverá usar a extensão de módulos para que o tema aceite os valores acima.

```
declare module '@material-ui/core/styles' {
  interface Theme {
    status: {
      danger: string;
    };
  }
  // allow configuration using `createTheme`
  interface ThemeOptions {
    status?: {
      danger?: string;
    };
  }
}
```

{{“demo”: “CustomStyles.js”}}

Acessando o tema em um componente

Você pode acessar as variáveis do tema dentro de seus componentes React.

- mui-theme-creator: A tool to help design and customize themes for the MUI component library. Inclui modelos de site básicos para mostrar vários componentes e como eles são afetados pelo tema
- create-mui-theme: É uma ferramenta online para criar temas de Material-UI por meio da ferramenta de cor do Material Design.

Acessando o tema em um componente

Você pode acessar as variáveis do tema dentro de seus componentes React.

Aninhando o tema

Você pode aninhar vários provedores de tema.

{{“demo”: “ThemeNesting.js”}}

O tema interno **sobrescreverá** o tema externo. Você pode estender o tema externo fornecendo uma função:

{{“demo”: “ThemeNestingExtend.js”}}

API

`createTheme(options, ...args) => theme`

Gere uma base de temas sobre as opções recebidas. Then, pass it as a prop to `ThemeProvider`.

Argumentos

1. `options` (*object*): Takes an incomplete theme object and adds the missing parts.
2. `...args` (*object[]*): Deep merge the arguments with the about to be returned theme.

   Note: Only the first argument (`options`) is being processed by the `createTheme` function. If you want to actually merge two themes' options and create a new one based on them, you may want to deep merge the two options and provide them as a first argument to the `createTheme` function.

```
import { createTheme } from '@material-ui/core/styles';
import purple from '@material-ui/core/colors/purple';
import green from '@material-ui/core/colors/green';

const theme = createTheme({
  palette: {
    primary: {
      main: purple[500],
    },
    secondary: {
      main: green[500],
    },
  },
});
```

Retornos `theme` (*object*): A complete, ready-to-use theme object.

Examples

```
import { createTheme, responsiveFontSizes } from '@material-ui/core/styles';

let theme = createTheme();
theme = responsiveFontSizes(theme);
```

Argumentos Gere uma base de temas sobre as opções recebidas.

```
-function TabPanel(props) {
+const TabPanel = React.forwardRef(function TabPanel(props, ref) {
```

```
  return <div role="tabpanel" {...props} ref={ref} />;
-}
+});

function Tabs() {
  return <Fade><TabPanel>...</TabPanel></Fade>;
}
```

Think of creating a theme as a two-step composition process: first, you define the basic design options; then, you'll use these design options to compose other options (example above) or to override the design of specific components (example below).

`responsiveFontSizes(theme, options) => theme`

Gera configurações de tipografia responsivas com base nas opções recebidas.

Retornos

1. `theme` (*object*): The theme object to enhance.
2. `options` (*object* [optional]):

- `breakpoints` (*array<string>* [optional]): Default to `['sm', 'md', 'lg']`. Array de pontos de quebra (identificadores).
- `disableAlign` (*bool* [optional]): Default to `false`. Se os tamanhos de fonte mudam pouco, as alturas da linha são preservadas e alinhadas à altura da linha da grade em 4px do Material Design. Isso requer uma altura de linha sem unidade nos estilos do tema.
- `factor` (*number* [optional]): Default to `2`. Este valor determina o fator de redimensionamento do tamanho da fonte. Quanto maior o valor, menor a diferença entre tamanhos de fonte em telas pequenas. Quanto menor o valor, maiores os tamanhos de fonte para telas pequenas. O valor deve ser maior que 1.
- `variants` (*array<string>* [optional]): Default to all. As variantes de tipografia para manipular.

Exemplos `theme` (*object*): The new theme with a responsive typography.

Examples

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

let theme = createTheme();
theme = responsiveFontSizes(theme);
```

`unstable_createMuiStrictModeTheme(options, ...args) => theme`

Usando `unstable_createMuiStrictModeTheme` restringe o uso de alguns de nossos componentes.

Gera um tema que reduz a quantidade de avisos dentro de `React. StrictMode` como por exemplo, `Warning: findDOMNode is deprecated in StrictMode`.

Argumentos Atualmente `unstable_createMuiStrictModeTheme` não adiciona requisitos adicionais.

Retornos

1. `options` (*object*): Takes an incomplete theme object and adds the missing parts.
2. `...args` (*object[]*): Deep merge the arguments with the about to be returned theme.

Exemplos `theme` (*object*): A complete, ready-to-use theme object.

Examples

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

let theme = createTheme({
  shape: {
    borderRadius: 4,
  },
});

theme = createTheme(theme, {
  components: {
    MuiChip: {
      styleOverrides: {
        root: {
          // apply theme's border-radius instead of component's default
          borderRadius: theme.shape.borderRadius,
        },
      },
    },
  },
});
```

`ThemeProvider`

This component takes a `theme` prop and applies it to the entire React tree that it is wrapping around. Deve preferencialmente ser usado na **raiz da sua árvore**

de componentes.

Props

Name	Type	Description
children	node	Your component tree.
theme *	union: object | func	A theme object, usually the result of `createTheme()`. The provided theme will be merged with the default theme. You can provide a function to extend the outer theme.

Examples

```
import * as React from 'react';
import ReactDOM from 'react-dom';
import { red } from '@mui/material/colors';
import { ThemeProvider, createTheme } from '@mui/material/styles';

const theme = createTheme({
  palette: {
    primary: {
      main: red[500],
    },
  },
});

function App() {
  return <ThemeProvider theme={theme}>...</ThemeProvider>;
} ReactDOM.render(<App />, document.querySelector('#app'));
```