Theming 主题

用你的主题定制MUI。 你可以改变颜色、文字铸排等等。

主题可以指定组件的配色、平面的明暗、阴影的深浅、墨水元素的合适的不透明度等等。

样式可让您为应用程序应用一致的音调。 它可以让你 **自定义所有的设计方面** 项目，以满足您的企业或品牌的特定需求。

为了提高应用程序之间的一致性，你可以在明暗主题类型中选择。 默认情况下，组件会使用浅色的主题样式。

Theme provider

如果你想要使用自定义的主题，那么需要使用 `MuiThemeProvider` 组件将样式注入到你的应用中。 但是，这是可选的；因为 Material-UI 组件带有默认主题。

`ThemeProvider` 依赖于 React 的上下文（context）功能来将主题传递给下级组件，所以你需要确保 `ThemeProvider` 是你试图自定义组件的父级组件。 您可以在 API 章节中了解有关此内容的更多信息 。

主题配置变量

修改主题配置变量是使 MUI 符合你的需求的最有效方法。 以下各节涵盖了一些最重要的主题变量：

- `.palette` 调色板
- `.typography` 文字铸排
- `.spacing` 间距
- `.breakpoints` 断点
- `.zIndex` 层级
- `.transitions` 过渡动画
- `.components` 组件

您可以在默认主题部分查看完整的默认样式。

自定义变量

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. 就像这样： For instance:

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

如果您使用的是 TypeScript，您还需要使用 module augmentation 来让主题接受上述值。

```
declare module '@mui/material/styles' {
  interface Theme {
    status: {
      danger: string;
    };
  }
```

```
  // 允许配置文件使用 `createTheme`
  interface ThemeOptions {
    status?: {
      danger?: string;
    };
  }
}
```

{{"demo": "CustomStyles.js"}}

访问一个组件中的主题

社区中有一些强大的工具来帮您构建主题：

- mui-theme-creator：一个帮助设计和定制 MUI 组件库主题的工具。 这其中包括基本的网站模板，并且展示各种组件以及它们如何受到主题的影响
- Material 调色板生成器：您可以在 Material 调色板生成器中输入的任何颜色，它将帮您生成一系列的颜色组合。

访问一个组件中的主题

你可以访问 React 组件内部的主题变量。

嵌套主题

您可以嵌套多个主题提供者。

{{"demo": "ThemeNesting.js"}}

内部主题将 **覆盖** 外部主题。 你可以提供一个函数来扩展外层主题：

{{"demo": "ThemeNestingExtend.js"}}

API

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

根据接收的选项生成样式。 Then, pass it as a prop to `ThemeProvider`.

参数

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. 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],
    },
  },
});
```

返回结果

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

Examples

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

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

参数

通过接收的选项生成一个主题基础。

```
import { unstable_createMuiStrictModeTheme } from '@material-ui/core/styles';

const theme = unstable_createMuiStrictModeTheme();

function App() {
  return (
    <React.StrictMode>
      <ThemeProvider theme={theme}>
        <LandingPage />
      </ThemeProvider>
    </React.StrictMode>
  );
}
```

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`

根据接收到的选项生成响应式的文字铸排设置。

返回结果

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

- `breakpoints` (*array<string>* [optional]): Default to `['sm', 'md', 'lg']`. 一个 breakpoints 的数组（identifiers）。 一个 breakpoints 的数组（identifiers）。
- `disableAlign` (*bool* [optional]): Default to `false`. 字体大小是否略有变化，这样能够保持行高并与 Material Design 的 4px 行高网格相对齐。 字体大小是否略有变化，这样能够保持行高并与 Material Design 的 4px 行高网格相对齐。 这需要主题样式中的无单位行高度。
- `factor` (*number* [optional]): Default to `2`. 此值决定了字体大小调整的强度。 此值决定了字体大小调整的强度。 值越高的话，在较小的屏幕上字体大小之间的差异就越小。 值越低的话，在较小屏幕上的字体就越大。 该值必须大于 1。
- `variants` (*array<string>* [optional]): Default to all. 需要处理的文字变体。 需要处理的文字变体。

例子

`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`

WARNING: Do not use this method in production.

生成一个减少 `React.StrictMode` 内的警告数量的主题，类似于 `Warning: findDOMNode is deprecated in StrictMode`。

参数

目前 `unstable_createMuiStrictModeTheme` 没有添加额外的要求。

返回结果

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.

例子

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

Examples

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

const theme = unstable_createMuiStrictModeTheme();

function App() {
  return (
    <React.StrictMode>
      <ThemeProvider theme={theme}>
        <LandingPage />
      </ThemeProvider>
    </React.StrictMode>
  );
}
```

`ThemeProvider`

This component takes a `theme` prop and applies it to the entire React tree that it is wrapping around. 最好在**您的组件树的根目录**中使用它。 It should preferably be used at **the root of your component tree**.

属性

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'));
```