TypeScript

借助 TypeScript，你可以为 JavaScript 添加静态类型，从而提高代码质量及开发者的工作效率。

`withStyles` 的使用

MUI requires a minimum version of TypeScript 3.5. Material-UI requires a minimum version of TypeScript 3.5. 请查看一下 Create React App with TypeScript 的例子。

请查看 Create React App with TypeScript 的例子。

```
{
  "compilerOptions": {
    "lib": ["es6", "dom"],
    "noImplicitAny": true,
    "noImplicitThis": true,
    "strictNullChecks": true
  }
}
```

对每个发布在 `@types/` 命名空间下的类型声明包，同样需要启用严格模式（strict mode）。 使用不太严格的 `tsconfig.json` 或省略某些库可能会带来一些错误。 若您想获得最佳类型（type）的体验，我们建议设置 `"strict": true` 。

处理 `值` 和事件处理器

很多与用户输入有关的组件会提供一个 `value` 属性或者包含当前 `值` 的事件处理器。 大多数情况下 `值` 只在 React 内被处理，这样的话它能够是任何类型，譬如 objects 或者 arrays。

然而，如果是它依赖于组件子项的情况，此类型无法在编译时被验证，例如对于 `Select` 或者 `RadioGroup` 来说。 这意味着留给我们的最合适的选项是将其输入为 `unknown` 并让开发者自行决定如何来缩小该类型。 与 `event.target` 在 React 中并不通用的原因相同，我们并不推荐您在这些案例中尝试使用一个通用的类型。

这些演示包括使用类型转换的类型变量。 鉴于所有的类型都位于一个文件中，并且都是非常基本的，这样的折衷可以接受。 您必须自行决定是否能够接受同样的折衷。 The library types are strict by default and loose via opt-in.

`component` 属性的用法

Moved to /customization/theming/#custom-variables.

`component` 属性的用法

然而，如果您尝试根据主题来构建样式，那么类型扩展会再次显示其不怎么雅观的部分：