TypeScript

You can add static typing to JavaScript to improve developer productivity and code quality thanks to TypeScript.

Minimum configuration

MUI requires a minimum version of TypeScript 3.5. Have a look at the Create React App with TypeScript example.

For types to work, it's recommended that you have at least the following options enabled in your `tsconfig.json`:

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

The strict mode options are the same that are required for every types package published in the `@types/` namespace. Using a less strict `tsconfig.json` or omitting some of the libraries might cause errors. To get the best type experience with the types we recommend setting `"strict": true`.

Handling `value` and event handlers

Many components concerned with user input offer a `value` prop or event handlers which include the current `value`. In most situations that `value` is only handled within React which allows it be of any type, such as objects or arrays.

However, that type cannot be verified at compile time in situations where it depends on the component's children e.g. for `Select` or `RadioGroup`. This means that the soundest option is to type it as `unknown` and let the developer decide how they want to narrow that type down. We do not offer the possibility to use a generic type in those cases for the same reasons `event.target` is not generic in React.

The demos include typed variants that use type casting. It is an acceptable tradeoff because the types are all located in a single file and are very basic. You have to decide for yourself if the same tradeoff is acceptable for you. The library types are strict by default and loose via opt-in.

Customization of `Theme`

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

Usage of `component` prop

Moved to /guides/composition/#with-typescript.