About the lab

This package hosts the incubator components that are not yet ready to move to the core.

The main difference between the lab and the core is how the components are versioned. Having a separate lab package allows us to release breaking changes when necessary while the core package follows a slower-moving policy.

As developers use and test the components and report issues, the maintainers learn more about shortcomings of the components: missing features, accessibility issues, bugs, API design, etc. The older and more used a component is, the less likely it is that new issues will be found and subsequently need to introduce breaking changes.

For a component to be ready to move to the core, the following criteria are considered:

- It needs to be **used**. The MUI team uses Google Analytics in the documentation (among other metrics) to evaluate the usage of each component. A lab component with low usage either means that it isn't fully working yet, or that there is low demand for it.
- It needs to match the **code quality** of the core components. It doesn't have to be perfect to be part of the core, but the component should be reliable enough that developers can depend on it.
  - Each component needs **type definitions**. It is not currently required that a lab component is typed, but it would need to be typed to move to the core.
  - Requires good **test coverage**. Some of the lab components don't currently have comprehensive tests.
- Can it be used as **leverage** to incentivize users to upgrade to the latest major release? The less fragmented the community is, the better.
- It needs to have a low probability of a **breaking change** in the short/medium future. For instance, if it needs a new feature that will likely require a breaking change, it may be preferable to delay its promotion to the core.

Installation

Install the package in your project directory with:

```
// with npm
npm install @mui/lab

// with yarn
yarn add @mui/lab
```

The lab has a peer dependency on the core components. If you are not already using MUI in your project, you can install it with:

```
// with npm
npm install @mui/material

// with yarn
yarn add @mui/material
```

TypeScript

In order to benefit from the CSS overrides and default prop customization with the theme, TypeScript users need to import the following types. Internally, it uses module augmentation to extend the default theme structure with the extension components available in the lab.

```
// When using TypeScript 4.x and above
import type {} from '@mui/lab/themeAugmentation';
// When using TypeScript 3.x and below
import '@mui/lab/themeAugmentation';

const theme = createTheme({
  components: {
    MuiTimeline: {
      styleOverrides: {
        root: {
          backgroundColor: 'red',
        },
      },
    },
  },
});
```