# 从 @material-ui-pickers 进行迁移

@material-ui/pickers 现在已经整合进了 @material-ui/lab。

> ⚠️ ***时间选择器组件已被重写。*** *我们重写了大部分的逻辑，所以不可能维护整个更改列表。 这个部分概述了最重要的改变内容。 如果你要升级，最简单的方法可能是把你代码库中每个选择器的用法都浏览一遍，然后逐一重写。 别忘了每次都要运行你的测试代码！*

本指南概述了自选择器 v3.2.10 版本依赖所改变的核心内容。

## 安装

You need to install the `@material-ui/lab` package if it's not already installed. ⚠️ Make sure you have installed the latest version, `"@material-ui/lab": ^5.0.0-alpha.30` or above. ⚠️ Make sure you have installed the latest version, `"@mui/lab": ^5.0.0-alpha.30` or above. ⚠️ Make sure you have installed the latest version, `"@mui/lab": ^5.0.0-alpha.30` or above.

## 导入

选择器的 `keyboard` 版本不再发布。 不管是移动还是桌面端的选择器都实现了键盘输入以便于无障碍化的推广。

```
-import { KeyboardDatePicker } from '@material-ui/pickers';
+import DatePicker from '@material-ui/lab/DatePicker';

-<KeyboardDatePicker />
+<DatePicker />
```

另外，这些属性没有提供 `variant` 属性，而是被移到了不同的引入方式（import）中，这意味着如果你只使用桌面端的选择器，那么捆绑包将不会包含 `对话框` 。

- `<DesktopDatePicker />` – 仅桌面视图。
- `<MobileDatePicker />` – 仅移动视图。
- `<DatePicker />` – 根据用户**指针**的偏好来提供移动或桌面视图。
- `<StaticDatePicker />` – 选择器本身，不包含任何输入或者其他的包装器。

```
-import { DatePicker } from '@material-ui/pickers';
+import DesktopDatePicker from '@material-ui/lab/DesktopDatePicker';

-<DatePicker variant="inline" />
+<DesktopDatePicker />
```

同样的约定也适用于 `TimePicker` – `<DesktopTimePicker>` 和 `<MobileTimePicker />` 。

## MuiPickersUtilsProvider

`MuiPickersUtilsProvider` 已被移除，取而代之的是 `LocalizationProvider` 。 此外，选择器不再需要你手动安装 date-io 适配器。 所有东西都包含在了 `lab` 中。

❌ 之前：

```
import AdapterDateFns from '@date-io/date-fns';
import { MuiPickersUtilsProvider } from '@material-ui/pickers';
```

✅ 之后:

```
import AdapterDateFns from '@material-ui/lab/AdapterDateFns';
import LocalizationProvider from '@material-ui/lab/LocalizationProvider';


function App() {
  return (
    <LocalizationProvider dateAdapter={AdapterDateFns}>
      ...
    </LocalizationProvider>
  )
);
    </LocalizationProvider>
  )
);
    </LocalizationProvider>
  )
);
```

## 渲染输入

我们在 `renderInput` 中引入了一个新的 **required** 属性。 这简化了在非 Material-UI 输入框组件中的使用成本。

```
<DatePicker renderInput={(props) => <TextField {...props} />} />
<TimePicker renderInput={(props) => <TextField {...props} />} />
```

之前，属性将会在 `<TextField />` 组件上传播。 从现在起，你需要使用新的 `renderInput` 属性来提供这些:

```
 <DatePicker
- label="Date"
- helperText="Something"
+ renderInput={props => <TextField label="Date" helperText="Something" /> }
/>
```

## 状态管理

选择器的状态/值管理逻辑是从头开始重写的。 当日期选择器的每个视图结束时，它现在将会调用 `onChange` 属性。 `onError` 的处理程序现在也完全不同了。 你选择器的三重检查（Triple-check）与表单集成，因为表单集成问题可能很微妙。

## 无必填的掩码（mask）

掩码不再需要。 另外，如果你提供的掩码无效，拾取者将直接忽略掩码，并允许任意输入。

```jsx
<DatePicker
  mask="mm"
  value={new Date()}
  onChange={console.log}
  renderInput={(props) => (
    <TextField {...props} helperText="invalid mask" />
  )}
/>

<DatePicker
  value={new Date()}
  onChange={console.log}
  renderInput={(props) => (
    <TextField {...props} helperText="valid mask" />
  )}
/>
```

## 更多内容

- ````diff <DatePicker
  - format="DD-MMM-YYYY"
  - inputFormat="DD-MMM-YYYY" ``` ````

除此之外还有很多改动，需要当心，确保你的测试和构建通过。 如果你对日期选择器有进阶使用方法，可能会更简单地重写它。

如果你注意到有改进指南的机会，请打开一个拉取请求。