从 v3 迁移到 v4 版本

是的，我们已经发布了 v4 版本！

您还在找 v3 版本的文档吗？ 您可以在这里找到它们 。

> *此文档尚未完成。 您是否已经升级了站点并且遇到了一些并没有在此涉及的问题？ 请在 GitHub 添加您的更改。*

简介

This is a reference for upgrading your site from MUI v3 to v4. 您可能不会将这里所有涵盖的内容运用到你的站点上。While there's a lot covered here, you probably won't need to do everything for your site. 我们会尽我们最大的努力让文档简单易懂，并尽可能有序地介绍，这样您可以迅速对 v4 版本游刃有余。

为什么您需要迁移呢

此文档介绍了**如何**从 v3 版本迁移到 v4 版本。 关于迁移的**原因**，我们则在 Medium 上发布了一篇博客来详细解说。

更新您的依赖包

您需要做的第一件事，就是更新您的依赖包。

升级 Material-UI 的版本

若想要使用最新版本的 Material-UI，您必须更新 `package.json` 。

```
"dependencies": {
  "@material-ui/core": "^4.0.0"
}
```

或者运行

```
npm install @material-ui/core
```

或者

```
yarn add @material-ui/core
```

更新 React 的版本

对于 React 版本的最低要求是从 `react@^16.3.0` 升级到 `react@^16.8.0` 。 这样一来我们能够依赖 Hooks 的功能（我们已经不再使用 class API）。

更新 Material-UI Styles 的版本

若您以前使用 v3 版本的 `@material-ui/styles` ，您则需要更新 `package.json` ，这样才能使用最新版本的 Material-UI Styles。

```
"dependencies": {
  "@material-ui/styles": "^4.0.0"
}
```

或者运行

```
npm install @material-ui/styles
```

或者

```
yarn add @material-ui/styles
```

处理变化带来的系统崩溃

Core

- 每个组件会提供他们的 ref。 这是通过使用 `React.forwardRef()` 实现的。 这回影响到内部的组件树和显示的名称，进而会使得 shallow 或者 snapshot 测试崩溃。 `innerRef` 不再返回一个实例的 ref（或者当内部组件是一个函数组件时，什么都不返回），而是返回一个它根组件的 ref。 我们已经将相应的 API 文档在根组件中列出。

Styles（样式表单）

- ⚠ MUI depends on JSS v10. JSS v10 版本与 v9 版本不向后兼容。 JSS v10 is not backward compatible with v9. 请保证您的开发环境中未安装 JSS v9 版本。 （在您的 `package.json` 中删除 `react-jss` 会有所帮助）。 StylesProvider 组件替代了 JssProvider 组件。
- 请移除 `withTheme()` 中的第一个可选的参数。 （第一个参数是为从未出现的可能的未来选项的一个占位符。）

  It matches the emotion API and the styled-components API.

  ```
  -const DeepChild = withTheme()(DeepChildRaw);
  +const DeepChild = withTheme(DeepChildRaw);
  ```

- 重命名 `convertHexToRGB` 为 `hexToRgb`。

  ```
  -import { convertHexToRgb } from '@material-ui/core/styles/colorManipulator';
  +import { hexToRgb } from '@material-ui/core/styles';
  ```

- 设置 keyframes API 的范围。 您应该在您的代码中做出以下改变。 这对分离动画的逻辑有所帮助：

  ```
    rippleVisible: {
      opacity: 0.3,
  -   animation: 'mui-ripple-enter 100ms cubic-bezier(0.4, 0, 0.2, 1)',
  +   animation: '$mui-ripple-enter 100ms cubic-bezier(0.4, 0, 0.2, 1)',
    },
    '@keyframes mui-ripple-enter': {
      '0%': {
        opacity: 0.1,
      },
      '100%': {
        opacity: 0.3,
  ```

```
    },
  },
```

主题

- `theme.palette.augmentColor()` 方法不再对输入框的颜色产生副作用。 若想要正确地使用它，您必须使用其返回值。

```
-const background = { main: color };
-theme.palette.augmentColor(background);
+const background = theme.palette.augmentColor({ main: color });

console.log({ background });
```

- —您可以从主题创建中安全地移除下一个变体：

```
typography: {
- useNextVariants: true,
},
```

- 我们已经不再使用 `theme.spacing.unit`，您可以用新的 API 了：

```
label: {
  [theme.breakpoints.up('sm')]: {
-   paddingTop: theme.spacing.unit * 12,
+   paddingTop: theme.spacing(12),
  },
}
```

  提示：您可以提供多个参数：`theme.spacing(1, 2) // = '8px 16px'`。

  您可以在项目中使用 迁移小帮手 来让您的迁移流程更加顺畅。

布局

- [Grid] 本着支持任意间距值并且摈弃心理上一直需要在 8 的基础上计数的目的，我们改变了 spacing 的 API:

```
  /**
   * 在类别为`item` 组件之间定义间距。
   * 它只能用于类型为 `container` 的组件。
   */
-  spacing: PropTypes.oneOf([0, 8, 16, 24, 32, 40]),
+  spacing: PropTypes.oneOf([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]),
```

  从今往后，您可以使用主题来实现 一个自定义的网格间距变换函数。

- [Container] 从 `@material-ui/lab` 迁移到 `@material-ui/core`。

```
-import Container from '@material-ui/lab/Container';
+import Container from '@material-ui/core/Container';
```

TypeScript

`value` 类型

将 input 组件的 `value` 属性的类型正常化，这样可以使用 `unknown` 了。 这会影响 `InputBase`，`NativeSelect`，`OutlinedInput`，`Radio`，`RadioGroup`，`Select`，`SelectInput`，`TextArea` 和 `TextField`。

```
function MySelect({ children }) {
- const handleChange = (event: any, value: string) => {
+ const handleChange = (event: any, value: unknown) => {
    // handle value
  };

  return <Select onChange={handleChange}>{children}</Select>
}
```

我们在 TypeScript 指南中更详细地解释了此变更。

Button

- [Button] 删除不推荐使用的按钮变体（flat，raised 和 fab）：

  ```
  -<Button variant="raised" />
  +<Button variant="contained" />
  ```

  ```
  -<Button variant="flat" />
  +<Button variant="text" />
  ```

  ```
  -import Button from '@material-ui/core/Button';
  -<Button variant="fab" />
  +import Fab from '@material-ui/core/Fab';
  +<Fab />
  ```

  ```
  -import Button from '@material-ui/core/Button';
  -<Button variant="extendedFab" />
  +import Fab from '@material-ui/core/Fab';
  +<Fab variant="extended" />
  ```

- [ButtonBase] 传递给 组件 的属性的组件需要能接受一个 ref。 我们在 组合指南 中解释了迁移的策略。

  当 `button` 属性设置为 true 时，这也适用于 `BottomNavigationAction`，`Button`，`CardActionArea`，`Checkbox`，`ExpansionPanelSummary`，`Fab`，`IconButton`，`MenuItem`，`Radio`，`StepButton`，`Tab`，`TableSortLabel` 以及 `ListItem`。

Card（卡片）

- [CardActions] 将 `disableActionSpacing` 属性重命名为 `disableSpacing`。
- [CardActions] 移除 CSS 类中的 `disableActionSpacing`。
- [CardActions] 将 CSS 类 `action` 重命名为 `spacing`。

ClickAwayListener

- [ClickAwayListener] 隐藏 react-event-listener 的属性。

Dialog

- [DialogActions] 将 `disableActionSpacing` 属性重命名为 `disableSpacing`。
- [DialogActions] 将 CSS 类 `action` 重命名为 `spacing`。
- [DialogContentText] 不使用文字铸排变体 `subtitle1`，而使用 `body1`。
- [Dialog] 子组件能够接受一个 ref。 我们在 组合指南 中解释了迁移的策略。

Divider

- [Divider] 移除了弃用的 `inset` 属性：

```
-<Divider inset />
+<Divider variant="inset" />
```

ExpansionPanel（扩展面板）

- [ExpansionPanelActions] 将 CSS 类 `action` 重命名为 `spacing`。
- [ExpansionPanel] 提高 `disabled` 和 `expanded` 样式规则的 CSS 优先级。
- [ExpansionPanel] 将 `CollapseProps` 属性重命名为 `TransitionProps`。

Lists（列表）

- [List] 为了符合规范，我们重新在列表组件上做了调整：
  - 当使用头像时，您必须要使用 `ListItemAvatar` 组件。
  - 当使用左边的复选框时，您必须使用 `ListItemIcon` 组件。
  - 您必须要在图标按钮上设置 `edge` 属性。
- [List] `dense` 不再减少 `List` 元素的上下边距。
- [ListItem] 加强 `disabled` 和 `focusVisible` 样式规则的 CSS 特性。

Menu

- [MenuItem] 删除 MenuItem 的固定高度。 浏览器将会自行根据间距和行高来计算高度。

Modal

- [Modal] 子组件能够接受一个 ref。 组合指南解释了迁移的策略。

  这也适用于 `Dialog` 和 `Popover`。
- [Modal] 删除 Modal 组件类的自定义 API (独立使用时将减少-74%的打包大小)。
- [Modal] 现在忽略了 event.defaultPrevented。 即使当向下离开事件调用了 `event.preventDefault()`，新的逻辑也会关闭模态框。 `event.preventDefault()` 旨在禁用一些默认的行为，如单击一个复选框来

选中它；点击按钮来提交表单；以及点击左键来移除文本输入框的光标等等。 只有一些特殊的 HTML 元素才具有这些默认的行为。 若您不想触发模态框的 `onClose` 事件，您需要使用 `event.stopPropagation()` 。

Paper

- [Paper] 减小默认的 elevation（阴影高度）。 为了适配卡片组件和扩展面板组件，请更改默认纸张的阴影高度：

```
-<Paper />
+<Paper elevation={2} />
```

这也会影响 `扩展面板` 。

Portal

- [Portal] 当使用 `disablePortal` 属性的时候，子元素需要能够接受一个 ref。 组合指南解释了迁移的策略。

Slide 滑动

- [Slide] 子组件能够接受一个 ref。 组合指南解释了迁移的策略。

Slider

- [Slider] 从 `@material-ui/lab` 迁移到 `@material-ui/core` 。

```
-import Slider from '@material-ui/lab/Slider'
+import Slider from '@material-ui/core/Slider'
```

Switch 开关

- [Switch] 重新编写实施的代码能够更容易覆盖样式表。 请重命名类的名字以匹配规范的用词：

```
-icon
-bar
+thumb
+track
```

Snackbar（消息条）

- [Snackbar] 匹配新的规范。
  - 更改尺寸。
  - 将默认的过渡动画从 `Slide` 改成 `Grow` 。

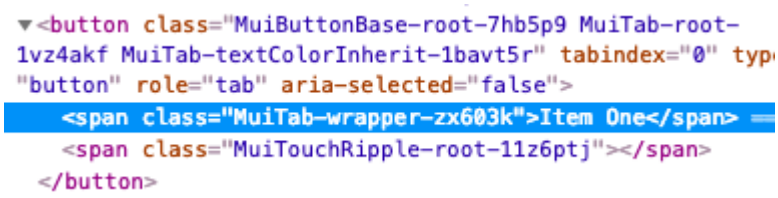
SvgIcon（Svg 图标）

- [SvgIcon] 重命名 nativeColor - > htmlColor。 React 在 `for` 这个 HTML 属性上也遇到了同样的问题，他们选择命名这个属性为 `htmlFor` 。 此变化的原因大同小异。

```
-<AddIcon nativeColor="#fff" />
+<AddIcon htmlColor="#fff" />
```

Tabs 选项卡

- [Tab] 为了简单起见，删除了 `labelContainer`， `label` 和 `labelWrapped` 等类的 key。 这使得我们可以移走两个中间的 DOM 元素。 您应该可以将自定义的样式移到 根元素 的类的键上。



- [Tabs] 移除了弃用的

```
-<Tabs fullWidth scrollable />
+<Tabs variant="scrollable" />
```

Table

- [TableCell] 移除了弃用的 `numeric` 属性：

```
-<TableCell numeric>{row.calories}</TableCell>
+<TableCell align="right">{row.calories}</TableCell>
```

- [TableRow] 删除了 CSS 属性中的固定高度。 浏览器将会自行根据间距和行高来计算单元格的高度。
- [TableCell] 将 `dense` 模式移至一个不同的属性：

```
-<TableCell padding="dense" />
+<TableCell size="small" />
```

- [TablePagination] 此组件不再修复无效的属性（ `page`， `count`， `rowsPerPage` ）组合。 相反的，它会给出一个警告。

TextField

- [InputLabel] 凭借 InputLabel 组件的类 API，您应该可以覆盖 FormLabel 组件所有的样式表。 我们移除了 `FormLabelClasses` 属性。

```
<InputLabel
- FormLabelClasses={{ asterisk: 'bar' }}
+ classes={{ asterisk: 'bar' }}
>
Foo
</InputLabel>
```

- [InputBase] 改变了默认的盒子模型的大小。 现如今它则使用以下的 CSS：

  ```
  box-sizing: border-box;
  ```

  与 `fullWidth` 属性有关的问题迎刃而解。

- [InputBase] 从 `InputBase` 中移走了 `inputType` 类。

Tooltip

- [Tooltip] 子组件能够接受一个 ref。 组合指南解释了迁移的策略。
- [Tooltip] 相比以前任何聚焦都会出现，现在只会在 focus-visible 聚焦的时候出现。

文字铸排

- [Typography] 移除了各种弃用的铸排变体。 您可以通过执行以下的替换来升级：
  - display4 => h1
  - display3 => h2
  - display2 => h3
  - display1 => h4
  - headline => h5
  - title => h6
  - subheading => subtitle1
  - body2 => body1
  - body1 (default) => body2 (default)
- [Typography] 移除了固定的 `display: block` 这个默认的铸排样式。 您现在可以使用新的 `display?: 'initial' | 'inline' | 'block';` 属性。
- [Typography] 为了达到更好的排版效果，请重命名属性 `headlineMapping` 为 `variantMapping`。

  ```
  -<Typography headlineMapping={headlineMapping}>
  +<Typography variantMapping={variantMapping}>
  ```

- [Typography] 将默认的字体从 `body2` 换成 `body1`。 默认为 16px 的字体大小比默认为 14px 好。 Bootstrap，material.io，甚至本文档都使用的是 16px 作为默认字体大小。 像 Ant Design 一样使用 14px 是可以理解的，因为中国的用户使用了不同的字母表。 我们建议将 12px 作为日语的默认字体大小。
- [Typography] 移除了铸排变体的默认颜色。 大多数情况下，字体颜色应该是继承而来的。 这是网站的默认行为。
- [Typography] 按照 该讨论 的逻辑，我们将 `color="default"` 重命名为 `color="initial"`。 你不应该再使用 _default_，因为它缺少明确的语义。

Node

- 是时候放弃对 node 6 的支持，而升级到 node 8 啦。

UMD

- 此更改简化了 Material-UI 与 CDN 的使用：

```
const {
  Button,
  TextField,
-} = window['material-ui'];
+} = MaterialUI;
```

它与其他 React 的项目保持一致：

- material-ui => MaterialUI
- react-dom => ReactDOM
- prop-types => PropTypes