从 v4 迁移到 v5 版本

是的，我们已经发布了 v5 版本！

如果你在寻找 v4 版本的文档，可以在这里 查看最近版本。

简介

这是一个将您的网站从 MUI core v4 版本升级到 v5 版本的参考。 您可能不需要将本篇文章涵盖的所有内容运用到你的站点上。 我们将尽最大努力使文档易于理解，并尽可能有序地向您介绍，以便您可以快速上手 v5！

为什么您需要迁移呢

能够获得对之前版本 bug 的修复，并增加了很多改进：如使用了新的样式引擎。 这个文档包含 **如何**将 v4 版本迁移到 v5 版。 关于迁移的**原因**，我们 发布了一篇博客来详细解说。

迁移步骤

- 更新 React & TypeScript
- 安装 ThemeProvider
- 更新 MUI
- 运行代码模块（codemods）
  - preset-safe
  - variant-prop (可选)
  - link-underline-hover (可选)
- 处理重大变更
- Migrate theme's `styleOverrides` to emotion
- Migrate from JSS
- CSS 特性
- Troubleshooting

> 💡 *目标是创建最小的更改，使迁移更顺利。 如果您遇到任何问题，请查看 疑难解答 章节。 对于其它没有在此文档描述的错误，请以此格式* `[Migration] Summary of your issue` *创建问题。*

更新 React & TypeScript 版本

- 支持**React**的最低版本从 v16.8.0 提高至 v17.0.0。
- 支持**TypeScript** 的最低版本从 v3.2 提高至 v3.5.

  > *我们尝试尽可能的与发布在 DefinitelyTyped 中的类型保持一致（如：发布于 npm 中* `@types` *命名空间内的包）。 我们不会在主要版本的 MUI 中更改支持的最低版本。 然而，我们通常建议不要使用低于 DefinitelyTyped 支持的最低版本的 TypeScript 版本。*

注意： 如果您的项目包含以下包，请将它们升级到 `最新` 版本。

- `react-scripts`
- `@types/react`
- `@types/react-dom`

> 📝 *请确保在继续下一步之前您的应用能够* ***正常运行****没有报错并且* ***应用了*** *更改。*

安装 `ThemeProvider`

在升级到 v5 前，请确保 `ThemeProvider` 应用在您程序的根节点（即使您正在使用**default theme**）并且在 `<ThemeProvider>` 之前**没有**调用 `useStyles` 。这是因为我们将要使用 `@mui/styles` **临时的** (JSS style-engine), 他需要使用 `ThemeProvider` 。

```
import { ThemeProvider, createMuiTheme, makeStyles } from '@material-
ui/core/styles';

const theme = createMuiTheme();

const useStyles = makeStyles((theme) => {
  root: {
    // 一些能够访问到theme的CSS代码
  }
});

function App() {
  const classes = useStyles(); // ❌ 如果您用到这段代码，请考虑将它移动到包裹在
<ThemeProvider>的组件内，
  return <ThemeProvider theme={theme}>{children}</ThemeProvider>;
}
```

> 📝 *Please make sure that your application is still **running** without errors and **commit** the change before continuing the next step.*

升级 MUI 版本

为了使用 `v5` 版本的 MUI Core，您首选需要升级下面的软件包：

```
npm install @mui/material @mui/styles

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

可选：如果您使用了下面的软件包，单独安装新的软件包：

- `@material-ui/lab` ，对应 `@mui/lab`
- `@material-ui/icons` ，对应 `@mui/icons-material`

▶ 查看所有软件包更改

然后，您需要添加新的对等依赖 - emotion 软件包：

```
npm install @emotion/react @emotion/styled

// or with `yarn`
yarn add @emotion/react @emotion/styled
```

> 💡 *如果您想要使用 MUI Core v5 的同时使用 **styled-components** 而不是 emotion，请查看安装指引。*

If you are using SSR (or a framework that depends on it), there is currently a known bug with the babel plugin for `styled-components`, which prevents `@mui/styled-engine-sc` (the adapter for `styled-components`) from being used. We strongly recommend using the default setup with emotion instead.

如果您使用 `@material-ui/pickers`，必须将其迁移到 `@mui/lab`。 您可以参考 这些步骤。

至此，您应该已经安装了 `@mui/styles`。 It includes JSS, which duplicate with emotion. It's meant to allow a gradual migration to v5. 您可以依照这些步骤移除依赖。

> 📝 *Please make sure that your application is still **running** without errors and **commit** the change before continuing the next step.*

我们准备了一些 codemods，提高您的迁移体验。

运行 codemods

We have prepared these codemods to ease your migration experience.

preset-safe

这个 codemods 包含了大部分的有助于迁移的转换器。 （**这个 codemod 在每个目录下仅应当应用一次**）

```
npx @mui/codemod v5.0.0/preset-safe <path>
```

> *如果您想逐一运行此转换器，请查看这个文档 preset-safe codemod 。*

variant-prop

但是，如果您想要在组件中保留 `variant="standard"`，请执行此 codemod 或在主题中配置默认属性。

> ❗ *如果您已经在主题中定义了默认值 `variant: "outlined"`，那么您**不应该**应用此 codemod。*

```
// if you have theme setup like this, ❌ don't run this codemod.
// 这些默认属性可以在之后移除，因为`always`在v5里面是默认值。
createMuiTheme({
  components: {
    MuiLink: {
      defaultProps: {
        underline: 'always',
      },
    },
  },
});
```

更多技术细节请参考此链接 variant-prop codemod。

```
npx @mui/codemod v5.0.0/variant-prop <path>
```

`@mui/material/colors/red` is considered private since v1.0.0. You should replace the import, more details about this error.

link-underline-hover

但是，如果您想要在组件中保留 `variant="hover"` ，请执行此 codemod 或在主题中配置默认属性。

> ❗*如果您已经在主题中定义了 `underline: "always"` ，那么您**不应该**使用此 codemod。*

```
// 如果您的主题像这样设置，❌请不要运行此codemod。
// 这些默认属性可以在之后移除，因为`outlined`在v5里面是默认值。 createMuiTheme({
  components: {
    MuiTextField: {
      defaultProps: {
        variant: 'outlined',
      },
    },
  },
});
```

If, however, you want to keep `underline="hover"`, run this codemod or configure theme default props.

```
npx @mui/codemod v5.0.0/link-underline-hover <path>
```

`span` element that wraps children has been removed. `wrapper` classKey is also removed. More details about this change.

一旦您完成了 codemod 步骤，请尝试再次运行您的应用程序。 此刻，您的程序应该可以运行并没有报错。 否则查看故障排除章节。 下一步，处理各组件中不兼容的改动。

变更

支持的浏览器和 node 版本

默认捆绑包的目标已更改。 实际支持的版本将在发布时从浏览器列表中查询 `"> 0.5%, last 2 versions, Firefox ESR, not dead, not IE 11, maintained node versions"` 。

默认捆绑包支持以下最小版本：

- Node 12（最低兼容到 8）
- Chrome 84（最低兼容到 49）
- Edge 91（最低兼容到 14）
- Firefox 78（最低兼容到 52）
- Safari 14 (macOS) 和 12.5 (iOS)（最低兼容到 10）
- 更多内容请（参阅 .browserslistrc (`stable` entry))

不再支持 IE 11。 如果你需要对 IE 11 进行兼容性支持，请查看我们的 旧版本包。

非转发类（non-ref-forwarding class）组件

对 `component` 属性中的非转发（non-ref-forwarding）类组件或作为直接 子类（`children`） 的支持已被放弃。 如果你使用了 `unstable_createStrictModeTheme` 或者在 `React.StrictMode` 中没有看到任何与 `findDOMNode` 相关的任何警告，那么你不需要做任何事情。 否则请查看我们指南中的 "注意事项与参考文献"部分 来了解如何迁移。 这个变化几乎影响了所有使用 `component` 属性的组件或者将 `children` 传递给要求 `children` 作为元素的组件（例如 `<MenuList><CustomMenuItem /></MenuList>` ）

Ref type specificity

For some components, you may get a type error when passing `ref`. To avoid the error, you should use a specific element type. For example, `Card` expects the type of `ref` to be `HTMLDivElement`, and `ListItem` expects its `ref` type to be `HTMLLIElement`. To avoid the error, you should use a specific element type. For example, `Card` expects the type of `ref` to be `HTMLDivElement`, and `ListItem` expects its `ref` type to be `HTMLLIElement`.

Here is an example:

```diff
 import * as React from 'react';
import Card from '@mui/material/Card';
import ListItem from '@mui/material/ListItem';

export default function SpecificRefType() {
- const cardRef = React.useRef<HTMLElement>(null);
+ const cardRef = React.useRef<HTMLDivElement>(null);

- const listItemRef = React.useRef<HTMLElement>(null);
+ const listItemRef = React.useRef<HTMLLIElement>(null);
  return (
    <div>
      <Card ref={cardRef}></Card>
      <ListItem ref={listItemRef}></ListItem>
    </div>
  );
}
```

The list of components that expect a specific element type is as follows:

`@mui/material`

- Accordion - `HTMLDivElement`
- Alert - `HTMLDivElement`
- Avatar - `HTMLDivElement`
- ButtonGroup - `HTMLDivElement`
- Card - `HTMLDivElement`
- Dialog - `HTMLDivElement`
- ImageList - `HTMLUListElement`
- List - `HTMLUListElement`
- Tab - `HTMLDivElement`
- Tabs - `HTMLDivElement`
- ToggleButton - `HTMLButtonElement`

`@mui/lab`

- Timeline - `HTMLUListElement`

Style library

The style library used by default in v5 is `emotion`. While migrating from JSS to emotion, and if you are using JSS style overrides for your components (for example overrides created by `makeStyles`), you will need to take care of the CSS injection order. To do so, you need to have the `StyledEngineProvider` with the `injectFirst` option at the **top of your component tree**. While migrating from JSS to emotion, and if you are using JSS style overrides for your components (for example overrides created by `makeStyles`), you will need to take care of the CSS injection order. To do so, you need to have the `StyledEngineProvider` with the `injectFirst` option at the **top of your component tree**.

> ✅ *This is handled in the preset-safe codemod.*

Here is an example:

```
import * as React from 'react';
import { StyledEngineProvider } from '@mui/material/styles';

export default function GlobalCssPriority() {
  return (
    {/* Inject emotion before JSS */}
    <StyledEngineProvider injectFirst>
      {/* Your component tree. 现在您可以覆盖 Material-UI 的样式。 Now you can override
MUI's styles. */}
    </StyledEngineProvider>
  );
}
```

> ***注意：*** *如果您使用 emotion 样式库渲染您的应用，并且有一个自定义缓存，它会覆盖 MUI 提供的缓存。 为了使注入顺序仍然正确，您需要添加 `prepend` 选项到 `createCache` 中。*
>
> ✅ *This is handled in the preset-safe codemod.*

Here is an example:

```
 import * as React from 'react';
 import { CacheProvider } from '@emotion/react';
 import createCache from '@emotion/cache';

 const cache = createCache({
   key: 'css',
+  prepend: true,
 });

 export default function PlainCssPriority() {
   return (
     <CacheProvider value={cache}>
       {/* Your component tree. Now you can override MUI's styles. */}
     </CacheProvider>
   );
 }
```

> ***注意：*** *如果您正在使用 styled-components 并且有带有自定义 `target` 属性的 `StyleSheetManager`，请确保目标是 HTML `<head>` 中的第一个元素。 如果想了解更多细节，请查阅 `@mui/styled-engine-sc` 程序包中的 `StyledEngineProvider` implementation。*

Theme structure

The structure of the theme has changed in v5. You need to update its shape. The structure of the theme has changed in v5. You need to update its shape. For a smoother transition, the `adaptV4Theme` helper allows you to iteratively upgrade some of the theme changes to the new theme structure.

> ✅ *This is handled in the [preset-safe codemod](#).*

```diff
-import { createMuiTheme } from '@mui/material/styles';
+import { createTheme, adaptV4Theme } from '@mui/material/styles';

-const theme = createMuiTheme({
+const theme = createTheme(adaptV4Theme({
   // v4 theme
-});
+}));
```

> ⚠️ *此适配器只处理 `createTheme` 的输入参数， 在创建主题后修改的样式需要手动迁移。*

The following changes are supported by the adapter:

- The "gutters" abstraction hasn't proven to be used frequently enough to be valuable.

  ```diff
  -theme.mixins.gutters(),
  +paddingLeft: theme.spacing(2),
  +paddingRight: theme.spacing(2),
  +[theme.breakpoints.up('sm')]: {
  +  paddingLeft: theme.spacing(3),
  +  paddingRight: theme.spacing(3),
  +},
  ```

- `theme.spacing` now returns single values with px units by default. This change improves the integration with styled-components & emotion. This change improves the integration with styled-components & emotion.

  > ✅ *This is handled in the [preset-safe codemod](#) by removing any 'px' suffix from `theme.spacing` calls in a template string.*

  Before:

  ```
  theme.spacing(2) => 16
  ```

  After:

  ```
  theme.spacing(2) => '16px'
  ```

- The `theme.palette.type` key was renamed to `theme.palette.mode`, to better follow the "dark mode" term that is usually used for describing this feature.

  > ✅ *This is handled in the [preset-safe codemod](#).*

> ```diff
>  import { createTheme } from '@mui/material/styles';
> -const theme = createTheme({palette: { type: 'dark' }}),
> +const theme = createTheme({palette: { mode: 'dark' }}),
> ```

- The default `theme.palette.info` colors were changed to pass AA standard contrast ratio in both light & dark mode.

  ```diff
   info = {
  -  main: cyan[500],
  +  main: lightBlue[700], // lightBlue[400] in "dark" mode

  -  light: cyan[300],
  +  light: lightBlue[500], // lightBlue[300] in "dark" mode

  -  dark: cyan[700],
  +  dark: lightBlue[900], // lightBlue[700] in "dark" mode
   }
  ```

- The default `theme.palette.success` colors were changed to pass AA standard contrast ratio in both light & dark mode.

  ```diff
   success = {
  -  main: green[500],
  +  main: green[800], // green[400] in "dark" mode

  -  light: green[300],
  +  light: green[500], // green[300] in "dark" mode

  -  dark: green[700],
  +  dark: green[900], // green[700] in "dark" mode
   }
  ```

- The default `theme.palette.warning` colors were changed to pass AA standard contrast ratio in both light & dark mode.

  ```diff
   warning = {
  -  main: orange[500],
  +  main: "#ED6C02", // orange[400] in "dark" mode

  -  light: orange[300],
  +  light: orange[500], // orange[300] in "dark" mode

  -  dark: orange[700],
  +  dark: orange[900], // orange[700] in "dark" mode
   }
  ```

- The `theme.palette.text.hint` key was unused in MUI components, and has been removed. If you depend on it, you can add it back: If you depend on it, you can add it back:

```diff
 import { createTheme } from '@mui/material/styles';

-const theme = createTheme(),
+const theme = createTheme({
+  palette: { text: { hint: 'rgba(0, 0, 0, 0.38)' } },
+});
```

- The components' definitions in the theme were restructured under the `components` key, to allow for easier discoverability of the definitions related to any one component.

  1. `props`

```diff
 import { createTheme } from '@mui/material/styles';

 const theme = createTheme({
-  props: {
-    MuiButton: {
-      disableRipple: true,
-    },
-  },
+  components: {
+    MuiButton: {
+      defaultProps: {
+        disableRipple: true,
+      },
+    },
+  },
 });
```

  2. `overrides`

```diff
 import { createTheme } from '@mui/material/styles';

 const theme = createTheme({
-  overrides: {
-    MuiButton: {
-      root: { padding: 0 },
-    },
-  },
+  components: {
+    MuiButton: {
+      styleOverrides: {
+        root: { padding: 0 },
+      },
+    },
+  },
 });
```

Styles

- Renamed `fade` to `alpha` to better describe its functionality. The previous name was leading to confusion when the input color already had an alpha value. Renamed `fade` to `alpha` to better describe its functionality. The previous name was leading to confusion when the input color already had an alpha value. The helper **overrides** the alpha value of the color.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > - import { fade } from '@mui/material/styles';
  > + import { alpha } from '@mui/material/styles';
  >
  > const classes = makeStyles(theme => ({
  > -  backgroundColor: fade(theme.palette.primary.main,
  > theme.palette.action.selectedOpacity),
  > +  backgroundColor: alpha(theme.palette.primary.main,
  > theme.palette.action.selectedOpacity),
  > }));
  > ```

- The `createStyles` function from `@mui/material/styles` was moved to the one exported from `@mui/styles`. It is necessary for removing the dependency to `@mui/styles` in the core package.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import { createStyles } from '@mui/material/styles';
  > +import { createStyles } from '@mui/styles';
  > ```

@mui/styles

ThemeProvider

If you are using the utilities from `@mui/styles` together with the `@mui/material`, you should replace the use of `ThemeProvider` from `@mui/styles` with the one exported from `@mui/material/styles`. This way, the `theme` provided in the context will be available in both the styling utilities exported from `@mui/styles`, like `makeStyles`, `withStyles` etc. and the MUI components. This way, the `theme` provided in the context will be available in both the styling utilities exported from `@mui/styles`, like `makeStyles`, `withStyles` etc. and the MUI components.

```diff
-import { ThemeProvider } from '@mui/styles';
+import { ThemeProvider } from '@mui/material/styles';
```

The `@mui/styles` package is no longer part of `@mui/material/styles`. If you are using `@mui/styles` together with `@mui/material` you need to add a module augmentation for the `DefaultTheme`.

Default theme (TypeScript)

The `@mui/styles` package is no longer part of `@mui/material/styles`. If you are using `@mui/styles` together with `@mui/material` you need to add a module augmentation for the `DefaultTheme`.

> ✅ *This is handled in the preset-safe codemod.*

```
// in the file where you are creating the theme (invoking the function
`createTheme()`)
import { Theme } from '@mui/material/styles';

declare module '@mui/styles' {
  interface DefaultTheme extends Theme {}
}
```

@mui/material/colors

- Nested imports of more than 1 level are private. Nested imports of more than 1 level are private. You can't import color from `@mui/material/colors/red`.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import red from '@mui/material/colors/red';
  > +import { red } from '@mui/material/colors';
  > ```

@mui/material/styles

createGenerateClassName

- The `createGenerateClassName` function is no longer exported from `@mui/material/styles`. You should import it directly from `@mui/styles`. You should import it directly from `@mui/styles`.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import { createGenerateClassName } from '@mui/material/styles';
  > +import { createGenerateClassName } from '@mui/styles';
  > ```

  To generate custom class names **without** using `@mui/styles`, check out ClassNameGenerator for more details.

createMuiTheme

- The function `createMuiTheme` was renamed to `createTheme` to make it more intuitive to use with `ThemeProvider`.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import { createMuiTheme } from '@mui/material/styles';
  > +import { createTheme } from '@mui/material/styles';
  >
  > -const theme = createMuiTheme({
  > +const theme = createTheme({
  > ```

jssPreset

- The `jssPreset` object is no longer exported from `@mui/material/styles`. You should import it directly from `@mui/styles`. You should import it directly from `@mui/styles`.

> ✅ *This is handled in the [preset-safe codemod](#).*
>
> ```
> -import { jssPreset } from '@mui/material/styles';
> +import { jssPreset } from '@mui/styles';
> ```

makeStyles

- The `styled` JSS utility is no longer exported from `@mui/material/styles`. You can use the one exported from `@mui/styles` instead. Make sure to add a `ThemeProvider` at the root of your application, as the `defaultTheme` is no longer available. If you are using this utility together with `@mui/material`, it's recommended you use the `ThemeProvider` component from `@mui/material/styles` instead. You can use `@mui/styles/makeStyles` instead. Make sure to add a `ThemeProvider` at the root of your application, as the `defaultTheme` is no longer available. If you are using this utility together with `@mui/material`, it's recommended that you use the `ThemeProvider` component from `@mui/material/styles` instead.

> ✅ *This is handled in the [preset-safe codemod](#).*
>
> ```
> -import { makeStyles } from '@mui/material/styles';
> +import { makeStyles } from '@mui/styles';
> +import { createTheme, ThemeProvider } from '@mui/material/styles';
>
> +const theme = createTheme();
>  const useStyles = makeStyles((theme) => ({
>    background: theme.palette.primary.main,
>  }));
>  function Component() {
>    const classes = useStyles();
>    return <div className={classes.root} />
>  }
>
>  // In the root of your app
>  function App(props) {
> -  return <Component />;
> +  return <ThemeProvider theme={theme}><Component {...props} />
> </ThemeProvider>;
>  }
> ```

MuiThemeProvider

- The `MuiThemeProvider` component is no longer exported from `@mui/material/styles`. Use `ThemeProvider` instead. Use `ThemeProvider` instead.

> ✅ *This is handled in the [preset-safe codemod](#).*
>
> ```
> -import { MuiThemeProvider } from '@mui/material/styles';
> +import { ThemeProvider } from '@mui/material/styles';
> ```

ServerStyleSheets

- The `ServerStyleSheets` component is no longer exported from `@mui/material/styles`. You should import it directly from `@mui/styles`. You should import it directly from `@mui/styles`.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import { ServerStyleSheets } from '@mui/material/styles';
  > +import { ServerStyleSheets } from '@mui/styles';
  > ```

styled

- The `styled` JSS utility is no longer exported from `@mui/material/styles`. You can use the one exported from `@mui/styles` instead. Make sure to add a `ThemeProvider` at the root of your application, as the `defaultTheme` is no longer available. If you are using this utility together with `@mui/material`, it's recommended you use the `ThemeProvider` component from `@mui/material/styles` instead.

```
-import { styled } from '@mui/material/styles';
+import { styled } from '@mui/styles';
+import { createTheme, ThemeProvider } from '@mui/material/styles';

+const theme = createTheme();
 const MyComponent = styled('div')(({ theme }) => ({ background:
theme.palette.primary.main }));

 function App(props) {
-  return <MyComponent />;
+  return <ThemeProvider theme={theme}><MyComponent {...props} />
</ThemeProvider>;
 }
```

StylesProvider

- The `StylesProvider` component is no longer exported from `@mui/material/styles`. You should import it directly from `@mui/styles`. You should import it directly from `@mui/styles`.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import { StylesProvider } from '@mui/material/styles';
  > +import { StylesProvider } from '@mui/styles';
  > ```

useThemeVariants

- The `useThemeVariants` hook is no longer exported from `@mui/material/styles`. You should import it directly from `@mui/styles`. You should import it directly from `@mui/styles`.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import { useThemeVariants } from '@mui/material/styles';
  > +import { useThemeVariants } from '@mui/styles';
  > ```

withStyles

- The `withStyles` JSS utility is no longer exported from `@mui/material/styles`. You can use `@mui/styles/withStyles` instead. Make sure to add a `ThemeProvider` at the root of your application, as the `defaultTheme` is no longer available. If you are using this utility together with `@mui/material`, you should use the `ThemeProvider` component from `@mui/material/styles` instead.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```
  > -import { withStyles } from '@mui/material/styles';
  > +import { withStyles } from '@mui/styles';
  > +import { createTheme, ThemeProvider } from '@mui/material/styles';
  >
  > +const defaultTheme = createTheme();
  >  const MyComponent = withStyles((props) => {
  >    const { classes, className, ...other } = props;
  >    return <div className={clsx(className, classes.root)} {...other} />
  >  })(({ theme }) => ({ root: { background: theme.palette.primary.main
  > }}));
  >
  >  function App() {
  > -  return <MyComponent />;
  > +  return <ThemeProvider theme={defaultTheme}><MyComponent />
  > </ThemeProvider>;
  >  }
  > ```

- Replace the `innerRef` prop with the `ref` prop. Refs are now automatically forwarded to the inner component. Refs are now automatically forwarded to the inner component.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```
  >  import * as React from 'react';
  >  import { withStyles } from '@mui/styles';
  >
  >  const MyComponent = withStyles({
  >    root: {
  >      backgroundColor: 'red',
  >    },
  >  })(({ classes }) => <div className={classes.root} />);
  >
  >  function MyOtherComponent(props) {
  >    const ref = React.useRef();
  > -  return <MyComponent innerRef={ref} />;
  > +  return <MyComponent ref={ref} />;
  >  }
  > ```

withTheme

- The `withTheme` HOC utility has been removed from the `@mui/material/styles` package. You can use `@mui/styles/withTheme` instead. Make sure to add a `ThemeProvider` at the root of your application, as the `defaultTheme` is no longer available. If you are using this utility together with `@mui/material`, it's recommended you use the `ThemeProvider` component from `@mui/material/styles` instead. You can use `@mui/styles/withTheme` instead. Make sure to add a `ThemeProvider` at the root of your application, as the `defaultTheme` is no longer available. If you are using this utility together with `@mui/material`, it's recommended you use the `ThemeProvider` component from `@mui/material/styles` instead.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```
  >  -import { withTheme } from '@mui/material/styles';
  >  +import { withTheme } from '@mui/styles';
  >  +import { createTheme, ThemeProvider } from '@mui/material/styles';
  >
  >  +const theme = createTheme();
  >   const MyComponent = withTheme(({ theme }) => <div>
  > {props.theme.direction}</div>);
  >
  >   function App(props) {
  >  -  return <MyComponent />;
  >  +  return <ThemeProvider theme={theme}><MyComponent {...props} />
  > </ThemeProvider>;
  >   }
  > ```

- Replace the `innerRef` prop with the `ref` prop. Refs are now automatically forwarded to the inner component. Refs are now automatically forwarded to the inner component.

  ```
  import * as React from 'react';
  import { withTheme } from '@mui/styles';

  const MyComponent = withTheme(({ theme }) => <div>{props.theme.direction}
  </div>);

  function MyOtherComponent(props) {
    const ref = React.useRef();
  - return <MyComponent innerRef={ref} />;
  + return <MyComponent ref={ref} />;
  }
  ```

withWidth

- This HOC was removed. This HOC was removed. There's an alternative using the [`useMediaQuery` hook](#).

  > ✅ *This is handled in the [preset-safe codemod](#) by applying hard-coded function to prevent the application from crashing.*

@mui/icons-material

GitHub

The `GitHub` icon was reduced in size from 24px to 22px wide to match the other icons size.

@material-ui/pickers

We have a dedicated page for migrating `@material-ui/pickers` to v5

System

- The following system functions (and properties) were renamed because they are considered deprecated CSS:
  - `gridGap` to `gap`
  - `gridRowGap` to `rowGap`
  - `gridColumnGap` to `columnGap`

  > ✅ *This is handled in the preset-safe codemod.*

- Use spacing unit in `gap`, `rowGap`, and `columnGap`. Use spacing unit in `gap`, `rowGap`, and `columnGap`. If you were using a number previously, you need to mention the px to bypass the new transformation with `theme.spacing`.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  >  <Box
  > -  gap={2}
  > +  gap="2px"
  >  >
  > ```

- Replace `css` prop with `sx` to avoid collision with styled-components & emotion `css` prop.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -<Box css={{ color: 'primary.main' }} />
  > +<Box sx={{ color: 'primary.main' }} />
  > ```

  > *Note that the system grid function wasn't documented in v4.*

Core components

As the core components use emotion as their style engine, the props used by emotion are not intercepted. The prop `as` in the following code snippet will not be propagated to `SomeOtherComponent`. The prop `as` in the following code snippet will not be propagated to `SomeOtherComponent`.

```
<MuiComponent component={SomeOtherComponent} as="button" />
```

AppBar

- Remove z-index when position static and relative. This avoids the creation of a stacking context and rendering issues. This avoids the creation of a stacking context and rendering issues.

- The `color` prop has no longer any effect in dark mode. The `color` prop has no longer any effect in dark mode. The app bar uses the background color required by the elevation to follow the [Material Design guidelines](). Use `enableColorOnDark` to restore the behavior of v4. Use `enableColorOnDark` to restore the behavior of v4.

  ```
  <AppBar enableColorOnDark />
  ```

Alert

- Move the component from the lab to the core. The component is now stable. The component is now stable.

  > ✅ *This is handled in the [preset-safe codemod]().*
  >
  > ```
  > -import Alert from '@mui/lab/Alert';
  > -import AlertTitle from '@mui/lab/AlertTitle';
  > +import Alert from '@mui/material/Alert';
  > +import AlertTitle from '@mui/material/AlertTitle';
  > ```

Autocomplete

- Move the component from the lab to the core. The component is now stable. The component is now stable.

  > ✅ *This is handled in the [preset-safe codemod]().*
  >
  > ```
  > -import Autocomplete from '@mui/lab/Autocomplete';
  > -import useAutocomplete  from '@mui/lab/useAutocomplete';
  > +import Autocomplete from '@mui/material/Autocomplete';
  > +import useAutoComplete from '@mui/material/useAutocomplete';
  > ```

- Remove `debug` prop. Remove `debug` prop. There are a couple of simpler alternatives: `open={true}`, Chrome devtools ["Emulate focused"](), or React devtools prop setter.

- `renderOption` should now return the full DOM structure of the option. It makes customizations easier. You can recover from the change with: It makes customizations easier. You can recover from the change with:

  ```
   <Autocomplete
  -  renderOption={(option, { selected }) => (
  -    <React.Fragment>
  +  renderOption={(props, option, { selected }) => (
  +    <li {...props}>
         <Checkbox
           icon={icon}
           checkedIcon={checkedIcon}
           style={{ marginRight: 8 }}
           checked={selected}
         />
         {option.title}
  -    </React.Fragment>
  +    </li>
  ```

```
  )}
/>
```

- Rename `closeIcon` prop to `clearIcon` to avoid confusion.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```diff
  > -<Autocomplete closeIcon={defaultClearIcon} />
  > +<Autocomplete clearIcon={defaultClearIcon} />
  > ```

- The following values of the reason argument in `onChange` and `onClose` were renamed for consistency:

  1. `create-option` to `createOption`
  2. `select-option` to `selectOption`
  3. `remove-option` to `removeOption`

- Change the CSS rules that use `[data-focus="true"]` to use `.Mui-focused`. The `data-focus` attribute is not set on the focused option anymore, instead, global class names are used. The `data-focus` attribute is not set on the focused option anymore, instead, global class names are used.

  ```diff
  -'.MuiAutocomplete-option[data-focus="true"]': {
  +'.MuiAutocomplete-option.Mui-focused': {
  ```

- Rename `getOptionSelected` to `isOptionEqualToValue` to better describe its purpose.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```diff
  >  <Autocomplete
  > -  getOptionSelected={(option, value) => option.title === value.title}
  > +  isOptionEqualToValue={(option, value) => option.title ===
  > value.title}
  > ```

Avatar

- Rename `circle` to `circular` for consistency:

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```diff
  > -<Avatar variant="circle">
  > -<Avatar classes={{ circle: 'className' }}>
  > +<Avatar variant="circular">
  > +<Avatar classes={{ circular: 'className' }}>
  > ```

  Since `circular` is the default value, the variant prop can be deleted:

  ```diff
  -<Avatar variant="circle">
  +<Avatar>
  ```

- Move the AvatarGroup from the lab to the core.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -import AvatarGroup from '@mui/lab/AvatarGroup';
> +import AvatarGroup from '@mui/material/AvatarGroup';
> ```

Badge

- Rename `circle` to `circular` and `rectangle` to `rectangular` for consistency.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -<Badge overlap="circle">
> -<Badge overlap="rectangle">
> +<Badge overlap="circular">
> +<Badge overlap="rectangular">
> ```

```
 <Badge classes={{
-  anchorOriginTopRightRectangle: 'className',
-  anchorOriginBottomRightRectangle: 'className',
-  anchorOriginTopLeftRectangle: 'className',
-  anchorOriginBottomLeftRectangle: 'className',
-  anchorOriginTopRightCircle: 'className',
-  anchorOriginBottomRightCircle: 'className',
-  anchorOriginTopLeftCircle: 'className',
+  anchorOriginTopRightRectangular: 'className',
+  anchorOriginBottomRightRectangular: 'className',
+  anchorOriginTopLeftRectangular: 'className',
+  anchorOriginBottomLeftRectangular: 'className',
+  anchorOriginTopRightCircular: 'className',
+  anchorOriginBottomRightCircular: 'className',
+  anchorOriginTopLeftCircular: 'className',
 }}>
```

BottomNavigation

- TypeScript: The `event` in `onChange` is no longer typed as a `React.ChangeEvent` but `React.SyntheticEvent`.

```
-<BottomNavigation onChange={(event: React.ChangeEvent<{}>) => {}} />
+<BottomNavigation onChange={(event: React.SyntheticEvent) => {}} />
```

BottomNavigationAction

- Remove the `span` element that wraps the children. Remove the `wrapper` classKey too. More details about this change. Remove the `wrapper` classKey too. More details about this change.

```
 <button class="MuiBottomNavigationAction-root">
-  <span class="MuiBottomNavigationAction-wrapper">
```

```diff
       {icon}
       <span class="MuiBottomNavigationAction-label">
         {label}
       </span>
-   </span>
  </button>
```

Box

- The `borderRadius` system prop value transformation has been changed. The `borderRadius` system prop value transformation has been changed. If it receives a number, it multiplies this value with the `theme.shape.borderRadius` value. Use a string to provide an explicit `px` value. Use a string to provide an explicit `px` value.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```diff
  > -<Box borderRadius="borderRadius">
  > +<Box borderRadius={1}>
  > ```

  ```diff
  -<Box borderRadius={16}>
  +<Box borderRadius="16px">
  ```

- The Box system props have an optional alternative API in v5, using the `sx` prop. You can [read this section](#) for the "why" behind this new API. You can [read this section](#) for the "why" behind this new API.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```
  > <Box border="1px dashed grey" p={[2, 3, 4]} m={2}>
  > <Box sx={{ border: "1px dashed grey", p: [2, 3, 4], m: 2 }}>
  > ```

- The following properties have been renamed because they are considered deprecated CSS properties by the CSS specification:

  > ✅ *This is handled in the [preset-safe codemod](#).*

  1. `gridGap` to `gap`
  2. `gridColumnGap` to `columnGap`
  3. `gridRowGap` to `rowGap`

  ```diff
  -<Box gridGap={1}>
  -<Box gridColumnGap={2}>
  -<Box gridRowGap={3}>
  +<Box gap={1}>
  +<Box columnGap={2}>
  +<Box rowGap={3}>
  ```

  (Note that the system grid function wasn't documented in v4.)

- The `clone` prop was removed because its behavior can be obtained by applying the `sx` prop directly to the child if it is a MUI component.

  ```diff
  -<Box sx={{ border: '1px dashed grey' }} clone>
  -  <Button>Save</Button>
  -</Box>
  +<Button sx={{ border: '1px dashed grey' }}>Save</Button>
  ```

- The ability to pass a render prop was removed because its behavior can be obtained by applying the `sx` prop directly to the child if it is a MUI component.

  ```diff
  -<Box sx={{ border: '1px dashed grey' }}>
  -  {(props) => <Button {...props}>Save</Button>}
  -</Box>
  +<Button sx={{ border: '1px dashed grey' }}>Save</Button>
  ```

  For non-MUI components, use the `component` prop.

  ```diff
  -<Box sx={{ border: '1px dashed grey' }}>
  -  {(props) => <button {...props}>Save</button>}
  -</Box>
  +<Box component="button" sx={{ border: '1px dashed grey' }}>Save</Box>
  ```

Button

- The button `color` prop is now "primary" by default, and "default" has been removed. This makes the button closer to the Material Design guidelines and simplifies the API. This makes the button closer to the Material Design guidelines and simplifies the API.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```diff
  >  <Button color="default">
  > +<Button>
  > ```

  If you prefer to use the `default` color in v4, take a look at this CodeSandbox

- `span` element that wraps children has been removed. `label` classKey is also removed. More details about this change. The `label` classKey is also removed. More details about this change.

  ```diff
   <button class="MuiButton-root">
  -  <span class="MuiButton-label">
       children
  -  </span>
   </button>
  ```

Chip

- Rename `default` variant to `filled` for consistency.

> ✅ *This is handled in the [preset-safe codemod](#).*

Since `filled` is the default value, the variant prop can be deleted:

```diff
-<Chip variant="default">
+<Chip>
```

Checkbox

- The checkbox color prop is now "primary" by default. To continue using the "secondary" color, you must explicitly indicate `secondary`. This brings the checkbox closer to the Material Design guidelines.

```diff
- <span class="MuiIconButton-root MuiButtonBase-root MuiCheckbox-root PrivateSwitchBase-root">
-   <span class="MuiIconButton-label">
-     <input class="PrivateSwitchBase-input">
+ <span class="MuiButtonBase-root MuiCheckbox-root PrivateSwitchBase-root">
+   <span class="PrivateSwitchBase-input">
```

- The component doesn't have `.MuiIconButton-root` and `.MuiIconButton-label` class names anymore, target `.MuiButtonBase-root` instead.

```diff
-<span class="MuiIconButton-root MuiButtonBase-root MuiCheckbox-root PrivateSwitchBase-root">
-  <span class="MuiIconButton-label">
-    <input class="PrivateSwitchBase-input">
+<span class="MuiButtonBase-root MuiCheckbox-root PrivateSwitchBase-root">
+  <span class="PrivateSwitchBase-input">
```

CircularProgress

- The `static` variant has been renamed to `determinate`, and the previous appearance of `determinate` has been replaced by that of `static`. It was an exception to Material Design, and was removed from the specification. It was an exception to Material Design, and was removed from the specification.

> ✅ *This is handled in the [preset-safe codemod](#).*
>
> ```diff
> -<CircularProgress variant="static" classes={{ static: 'className' }} />
> +<CircularProgress variant="determinate" classes={{ determinate: 'className' }} />
> ```

> *注意：如果你之前已经定制了 determinate，那么你的定制可能不再有效。 所以请删除它们。*

Collapse

- The `collapsedHeight` prop was renamed `collapsedSize` to support the horizontal direction.

> ✅ *This is handled in the [preset-safe codemod](#).*

> ```diff
> -<Collapse collapsedHeight={40}>
> +<Collapse collapsedSize={40}>
> ```

- The `classes.container` key was changed to match the convention of the other components.

  ```diff
  -<Collapse classes={{ container: 'collapse' }}>
  +<Collapse classes={{ root: 'collapse' }}>
  ```

CssBaseline

- The component was migrated to use the `@mui/styled-engine` (`emotion` or `styled-components`) instead of `jss` . You should remove the `@global` key when defining the style overrides for it. You could also start using the CSS template syntax over the JavaScript object syntax. You should remove the `@global` key when defining the style overrides for it. You could also start using the CSS template syntax over the JavaScript object syntax.

  ```diff
  const theme = createTheme({
    components: {
      MuiCssBaseline: {
  -     styleOverrides: {
  -       '@global': {
  -         html: {
  -           WebkitFontSmoothing: 'auto',
  -         },
  -       },
  -     },
  +     styleOverrides: `
  +       html {
  +         -webkit-font-smoothing: auto;
  +       }
  +     `
      },
    },
  });
  ```

- The `body` font size has changed from `theme.typography.body2` (`0.875rem`) to `theme.typography.body1` (`1rem`). To return to the previous size, you can override it in the theme: To return to the previous size, you can override it in the theme:

  ```js
  const theme = createMuiTheme({
    components: {
      MuiCssBaseline: {
        styleOverrides: {
          body: {
            fontSize: '0.875rem',
            lineHeight: 1.43,
            letterSpacing: '0.01071em',
          },
  ```

```
        },
      },
    },
  });
```

Dialog

- The onE* transition props were removed. Use TransitionProps instead. Use TransitionProps instead.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```diff
  >   <Dialog
  > -   onEnter={onEnter}
  > -   onEntered={onEntered}
  > -   onEntering={onEntering}
  > -   onExit={onExit}
  > -   onExited={onExited}
  > -   onExiting={onExiting}
  > +   TransitionProps={{
  > +     onEnter,
  > +     onEntered,
  > +     onEntering,
  > +     onExit,
  > +     onExited,
  > +     onExiting,
  > +   }}
  >   >
  > ```

- Remove the `disableBackdropClick` prop because it is redundant. Use `onClose` with `reason === 'backdropClick'` instead. Remove the `disableBackdropClick` prop because it is redundant. Ignore close events from `onClose` when `reason === 'backdropClick'` instead.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```diff
  >   <Dialog
  > -   disableBackdropClick
  > -   onClose={handleClose}
  > +   onClose={(event, reason) => {
  > +     if (reason !== 'backdropClick') {
  > +       handleClose(event, reason);
  > +     }
  > +   }}
  >   />
  > ```

- Remove the `withMobileDialog` higher-order component. The hook API allows a simpler and more flexible solution: The hook API allows a simpler and more flexible solution:

  > ✅ *This is handled in the preset-safe codemod by applying hard-coded function to prevent application crash, further fixes are required.*

> ```
> -import withMobileDialog from '@mui/material/withMobileDialog';
> +import { useTheme, useMediaQuery } from '@mui/material';
>
> function ResponsiveDialog(props) {
> - const { fullScreen } = props;
> + const theme = useTheme();
> + const fullScreen = useMediaQuery(theme.breakpoints.down('sm'));
>   const [open, setOpen] = React.useState(false);
>
> // ...
>
> -export default withMobileDialog()(ResponsiveDialog);
> +export default ResponsiveDialog;
>
> -export default withMobileDialog()(ResponsiveDialog);
> +export default ResponsiveDialog;
> ```

- Flatten DialogTitle DOM structure, remove `disableTypography` prop

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -<DialogTitle disableTypography>
> -  <Typography variant="h4" component="h2">
> +<DialogTitle>
> +  <Typography variant="h4" component="span">
> >        My header
> >      </Typography>
> ```

Divider

- Use border instead of background color. It prevents inconsistent height on scaled screens. Use border instead of background color. It prevents inconsistent height on scaled screens. If you have customized the color of the border, you will need to update the CSS property override:

```diff
.MuiDivider-root {
- background-color: #f00;
+ border-color: #f00;
}
```

ExpansionPanel

- Rename the `ExpansionPanel` components to `Accordion` to use a more common naming convention:

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -import ExpansionPanel from '@mui/material/ExpansionPanel';
> -import ExpansionPanelSummary from
> '@mui/material/ExpansionPanelSummary';
> -import ExpansionPanelDetails from
> ```

```
'@mui/material/ExpansionPanelDetails';
  -import ExpansionPanelActions from
'@mui/material/ExpansionPanelActions';
  +import Accordion from '@mui/material/Accordion';
  +import AccordionSummary from '@mui/material/AccordionSummary';
  +import AccordionDetails from '@mui/material/AccordionDetails';
  +import AccordionActions from '@mui/material/AccordionActions';

  -<ExpansionPanel>
  >   +<Accordion>
  >   -  <ExpansionPanelSummary>
  >   +  <AccordionSummary>
  >        <Typography>Location</Typography>
  >        <Typography>Select trip destination</Typography>
  >   -  </ExpansionPanelSummary>
  >   +  </AccordionSummary>
  >   -  <ExpansionPanelDetails>
  >   +  <AccordionDetails>
  >        <Chip label="Barbados" onDelete={() => {}} />
  >        <Typography variant="caption">Select your destination of
choice</Typography>
  >   -  </ExpansionPanelDetails>
  >   +  </AccordionDetails>
  >      <Divider />
  >   -  <ExpansionPanelActions>
  >   +  <AccordionActions>
  >        <Button size="small">Cancel</Button>
  >        <Button size="small">Save</Button>
  >   -  </ExpansionPanelActions>
  >   +  </AccordionActions>
  >   -</ExpansionPanel>
  +</Accordion>
```

- TypeScript: The `event` in `onChange` is no longer typed as a `React.ChangeEvent` but `React.SyntheticEvent`.

```
-<Accordion onChange={(event: React.ChangeEvent<{}>, expanded: boolean) =>
{}} />
+<Accordion onChange={(event: React.SyntheticEvent, expanded: boolean) => {}}
/>
```

ExpansionPanelDetails

- Remove `display: flex` from `AccordionDetails` (formerly `ExpansionPanelDetails`) as its too opinionated. Most developers expect a display block. Most developers expect a display block.

ExpansionPanelSummary

- Rename `focused` to `focusVisible` for consistency:

```diff
 <AccordionSummary
   classes={{
-    focused: 'custom-focus-visible-classname',
+    focusVisible: 'custom-focus-visible-classname',
   }}
  />
```

- Remove `IconButtonProps` prop from `AccordionSummary` (formerly `ExpansionPanelSummary`). The component renders a `<div>` element instead of an `IconButton`. The prop is no longer necessary. The component renders a `<div>` element instead of an `IconButton`. The prop is no longer necessary.

Fab

- Rename `round` to `circular` for consistency:

> ✅ *This is handled in the [preset-safe codemod](#).*
>
> ```diff
> -<Fab variant="round">
> +<Fab variant="circular">
> ```

- `span` element that wraps children has been removed. `label` classKey is also removed. More details about [this change](#). The `label` classKey is also removed. More details about [this change](#).

```diff
 <button class="MuiFab-root">
-  <span class="MuiFab-label">
     {children}
-  </span>
 </button>
```

FormControl

- Change the default variant from `standard` to `outlined`. Standard has been removed from the Material Design guidelines. Standard has been removed from the Material Design guidelines.

> ✅ *This is handled in [variant-prop codemod](#), read the details before running this codemod.*
>
> ```diff
> -<FormControl value="Standard" />
> -<FormControl value="Outlined" variant="outlined" />
> +<FormControl value="Standard" variant="standard" />
> +<FormControl value="Outlined" />
> ```

FormControlLabel

- The `label` prop is now required. The `label` prop is now required. If you were using a `FormControlLabel` without a `label`, you can replace it with just the value of the `control` prop.

```diff
-<FormControlLabel control={<Checkbox />} />
+<Checkbox />
```

Grid

- Rename `justify` prop to `justifyContent` to align with the CSS property name.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```diff
  > -<Grid justify="center">
  > +<Grid justifyContent="center">
  > ```

- The props: `alignItems` `alignContent` and `justifyContent` and their `classes` and style overrides keys were removed: "align-items-xs-center", "align-items-xs-flex-start", "align-items-xs-flex-end", "align-items-xs-baseline", "align-content-xs-center", "align-content-xs-flex-start", "align-content-xs-flex-end", "align-content-xs-space-between", "align-content-xs-space-around", "justify-content-xs-center", "justify-content-xs-flex-end", "justify-content-xs-space-between", "justify-content-xs-space-around" and "justify-content-xs-space-evenly". These props are now considered part of the system, not on the `Grid` component itself. If you still wish to add overrides for them, you can use the `theme.components.MuiGrid.variants` options. These props are now considered part of the system, not on the `Grid` component itself. If you still wish to add overrides for them, you can use the [callback as a value in `styleOverrides`](#).

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```diff
  > const theme = createTheme({
  >   components: {
  >     MuiGrid: {
  > -     styleOverrides: {
  > -       "align-items-xs-flex-end": {
  > -         marginTop: '20px',
  > -       },
  > -     },
  > +     variants: {
  > +       props: { alignItems: "flex-end" },
  > +       style: {
  > +         marginTop: '20px',
  > +       },
  > +     }],
  >     },
  >   },
  > });
  > ```

GridList

- Rename the `GridList` components to `ImageList` to align with the current Material Design naming.

  > ✅ *This is handled in the [preset-safe codemod](#).*

- Rename the GridList `spacing` prop to `gap` to align with the CSS attribute.

- Rename the GridList `cellHeight` prop to `rowHeight`.

- Add the `variant` prop to GridList.
- Rename the GridListItemBar `actionPosition` prop to `position`. (Note also the related classname changes.) (Note also the related classname changes.)
- Use CSS object-fit. Use CSS object-fit. For IE11 support either use a polyfill such as https://www.npmjs.com/package/object-fit-images, or continue to use the v4 component.

```diff
-import GridList from '@mui/material/GridList';
-import GridListTile from '@mui/material/GridListTile';
-import GridListTileBar from '@mui/material/GridListTileBar';
+import ImageList from '@mui/material/ImageList';
+import ImageListItem from '@mui/material/ImageListItem';
+import ImageListItemBar from '@mui/material/ImageListItemBar';

-<GridList spacing={8} cellHeight={200}>
-  <GridListTile>
+<ImageList gap={8} rowHeight={200}>
+  <ImageListItem>
    <img src="file.jpg" alt="Image title" />
-    <GridListTileBar
+    <ImageListItemBar
      title="Title"
      subtitle="Subtitle"
    />
-  </GridListTile>
-</GridList>
+  </ImageListItem>
+</ImageList>
```

Hidden

- This component is deprecated because its functionality can be created with the `sx` prop or the `useMediaQuery` hook.

  > ✅ *This is handled in the preset-safe codemod by applying fake `Hidden` component to prevent application crash, further fixes are required.*

  Use the `sx` prop to replace `implementation="css"`:

```diff
-<Hidden implementation="css" xlUp><Paper /></Hidden>
-<Hidden implementation="css" xlUp><button /></Hidden>
+<Paper sx={{ display: { xl: 'none', xs: 'block' } }} />
+<Box component="button" sx={{ display: { xl: 'none', xs: 'block' } }} />
```

```diff
-<Hidden implementation="css" mdDown><Paper /></Hidden>
-<Hidden implementation="css" mdDown><button /></Hidden>
+<Paper sx={{ display: { xs: 'none', md: 'block' } }} />
+<Box component="button" sx={{ display: { xs: 'none', md: 'block' } }} />
```

Use the `useMediaQuery` hook to replace `implementation="js"` :

```diff
-<Hidden implementation="js" xlUp><Paper /></Hidden>
+const hidden = useMediaQuery(theme => theme.breakpoints.up('xl'));
+return hidden ? null : <Paper />; null : <Paper />;
```

Icon

- The default value of `fontSize` was changed from `default` to `medium` for consistency. The default value of `fontSize` was changed from `default` to `medium` for consistency. In the unlikely event that you were using the value `default` , the prop can be removed:

```diff
-<Icon fontSize="default">icon-name</Icon>
+<Icon>icon-name</Icon>
```

IconButton

- The default size's padding is reduced to `8px` which makes the default IconButton size of `40px` . To get the old default size (`48px`), use `size="large"` . The change was done to better match Google's products when Material Design stopped documenting the icon button pattern. To get the old default size (`48px`), use `size="large"` . The change was done to better match Google's products when Material Design stopped documenting the icon button pattern.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```diff
  >  - <IconButton>
  >  + <IconButton size="large">
  > ```

- The `span` element that wraps children has been removed. The `label` classKey is also removed. More details about this change.

```diff
 <button class="MuiIconButton-root">
-  <span class="MuiIconButton-label">
     <svg />
-  </span>
 </button>
```

Link

- The default `underline` prop is changed from `"hover"` to `"always"` . To get the same behavior as in v4, apply `defaultProps` in theme To get the same behavior as in v4, apply `defaultProps` in theme

  > ✅ *This is handled in link-underline-hover codemod, read the details before running this codemod.*
  >
  > ```js
  >  createTheme({
  >  components: {
  >    MuiLink: {
  >      defaultProps: {
  > ```

>   underline: 'hover',
>   },
>   },
>   },
>   });

Menu

- The onE* transition props were removed. Use TransitionProps instead. Use TransitionProps instead.

> ✅ *This is handled in the [preset-safe codemod](#).*
>
> ```diff
>  <Menu
> -  onEnter={onEnter}
> -  onEntered={onEntered}
> -  onEntering={onEntering}
> -  onExit={onExit}
> -  onExited={onExited}
> -  onExiting={onExiting}
> +  TransitionProps={{
> +    onEnter,
> +    onEntered,
> +    onEntering,
> +    onExit,
> +    onExited,
> +    onExiting,
> +  }}
>  >
> ```

> *Note: The `selectedMenu` variant will no longer vertically align the selected item with the anchor.*

- Change the default value of `anchorOrigin.vertical` to follow the Material Design guidelines. The menu now displays below the anchor instead of on top of it. You can restore the previous behavior with:

```diff
 <Menu
+  anchorOrigin={{
+    vertical: 'top',
+    horizontal: 'left',
+  }}
```

MenuItem

- The `MenuItem` component inherits the `ButtonBase` component instead of `ListItem`. The class names related to "MuiListItem-*" are removed and theming `ListItem` is no longer affecting `MenuItem`.

```diff
-<li className="MuiButtonBase-root MuiMenuItem-root MuiListItem-root">
+<li className="MuiButtonBase-root MuiMenuItem-root">
```

- prop `listItemClasses` is removed, use `classes` instead.

  ```
  -<MenuItem listItemClasses={{...}}>
  +<MenuItem classes={{...}}>
  ```

  Read more about [MenuItem CSS API]()

Modal

- Remove the `disableBackdropClick` prop because it is redundant. Use `onClose` with `reason === 'backdropClick'` instead.

  > ✅ *This is handled in the [preset-safe codemod]().*
  >
  > ```
  >  <Modal
  > -  disableBackdropClick
  > -  onClose={handleClose}
  > +  onClose={(event, reason) => {
  > +    if (reason !== 'backdropClick') {
  > +      handleClose(event, reason);
  > +    }
  > +  }}
  >  />
  > ```

- Remove the `onEscapeKeyDown` prop because it is redundant. Remove the `onEscapeKeyDown` prop because it is redundant. Use `onClose` with `reason === "escapeKeyDown"` instead.

  > ✅ *This is handled in the [preset-safe codemod]().*
  >
  > ```
  >  <Modal
  > -  onEscapeKeyDown={handleEscapeKeyDown}
  > +  onClose={(event, reason) => {
  > +    if (reason === 'escapeKeyDown') {
  > +      handleEscapeKeyDown(event);
  > +    }
  > +  }}
  >  />
  > ```

- Remove `onRendered` prop. Remove `onRendered` prop. Depending on your use case either use a [callback ref]() on the child element or an effect hook in the child component.

NativeSelect

- Merge the `selectMenu` slot into `select`. Slot `selectMenu` was redundant. Merge the `selectMenu` slot into `select`. Slot `selectMenu` was redundant. The `root` slot is no longer applied to the select, but to the root.

  ```
  -<NativeSelect classes={{ root: 'class1', select: 'class2', selectMenu:
  'class3' }} />
  ```

```diff
+<NativeSelect classes={{ select: 'class1 class2 class3' }} />
```

OutlinedInput

- Remove the `labelWidth` prop. Remove the `labelWidth` prop. The `label` prop now fulfills the same purpose, using CSS layout instead of JavaScript measurement to render the gap in the outlined.

```diff
-<OutlinedInput labelWidth={20} />
+<OutlinedInput label="First Name" />
```

Paper

- Change the background opacity based on the elevation in dark mode. This change was done to follow the Material Design guidelines. You can revert it in the theme: This change was done to follow the Material Design guidelines. You can revert it in the theme:

```diff
const theme = createTheme({
  components: {
    MuiPaper: {
+     styleOverrides: { root: { backgroundImage: 'unset' } },
    },
  },
});
```

Pagination

- Move the component from the lab to the core. The component is now stable. The component is now stable.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```diff
> -import Pagination from '@mui/lab/Pagination';
> -import PaginationItem from '@mui/lab/PaginationItem';
> -import { usePagination } from '@mui/lab/Pagination';
> +import Pagination from '@mui/material/Pagination';
> +import PaginationItem from '@mui/material/PaginationItem';
> +import usePagination from '@mui/material/usePagination';
> ```

- Rename `round` to `circular` for consistency:

> ✅ *This is handled in the preset-safe codemod.*
>
> ```diff
> -<Pagination shape="round">
> -<PaginationItem shape="round">
> +<Pagination shape="circular">
> +<PaginationItem shape="circular">
> ```

Popover

- The onE* transition props were removed. Use TransitionProps instead. Use TransitionProps instead.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```diff
>  <Popover
> -  onEnter={onEnter}
> -  onEntered={onEntered}
> -  onEntering={onEntering}
> -  onExit={onExit}
> -  onExited={onExited}
> -  onExiting={onExiting}
> +  TransitionProps={{
> +    onEnter,
> +    onEntered,
> +    onEntering,
> +    onExit,
> +    onExited,
> +    onExiting,
> +  }}
>  >
> ```

- The `getContentAnchorEl` prop was removed to simplify the positioning logic.

Popper

- Upgrade Popper.js from v1 to v2. Upgrade Popper.js from v1 to v2. This third-party library has introduced a lot of changes.
  You can read their migration guide or the following summary:

  - The CSS prefixes have changed:

    ```diff
     popper: {
      zIndex: 1,
    - '&[x-placement*="bottom"] .arrow': {
    + '&[data-popper-placement*="bottom"] .arrow': {
    ```

  - Method names have changed:

    ```diff
    -popperRef.current.scheduleUpdate()
    +popperRef.current.update()
    ```

    ```diff
    -popperRef.current.update()
    +popperRef.current.forceUpdate()
    ```

  - Modifiers' API has changed a lot. There are too many changes to be covered here. There are too many changes to be covered here.

Portal

- Remove `onRendered` prop. Remove `onRendered` prop. Depending on your use case either use a callback ref on the child element or an effect hook in the child component.

Radio

- The radio color prop is now "primary" by default. The radio color prop is now "primary" by default. To continue using the "secondary" color, you must explicitly indicate `secondary`. This brings the radio closer to the Material Design guidelines. This brings the radio closer to the Material Design guidelines.

```diff
-<Radio />
+<Radio color="secondary />
```

- The component doesn't have `.MuiIconButton-root` and `.MuiIconButton-label` class names anymore, target `.MuiButtonBase-root` instead.

```diff
- <span class="MuiIconButton-root MuiButtonBase-root MuiRadio-root PrivateSwitchBase-root">
-   <span class="MuiIconButton-label">
-     <input class="PrivateSwitchBase-input">
+ <span class="MuiButtonBase-root MuiRadio-root PrivateSwitchBase-root">
+   <span class="PrivateSwitchBase-input">
```

Rating

- Move the component from the lab to the core. The component is now stable. The component is now stable.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```diff
> -import Rating from '@mui/lab/Rating';
> +import Rating from '@mui/material/Rating';
> ```

- Change the default empty icon to improve accessibility. Change the default empty icon to improve accessibility. If you have a custom `icon` prop but no `emptyIcon` prop, you can restore the previous behavior with:

```diff
 <Rating
   icon={customIcon}
+  emptyIcon={null}
 />
```

- Rename `visuallyhidden` to `visuallyHidden` for consistency:

```diff
 <Rating
   classes={{
-    visuallyhidden: 'custom-visually-hidden-classname',
+    visuallyHidden: 'custom-visually-hidden-classname',
   }}
 />
```

RootRef

- This component was removed. You can get a reference to the underlying DOM node of our components via `ref` prop. The component relied on [`ReactDOM.findDOMNode`]() which is [deprecated in `React.StrictMode`]().

  > ✅ *This is handled in the [preset-safe codemod]() by applying fake `RootRef` component to prevent application crash, further fixes are required.*
  >
  > ```
  > <RootRef rootRef={ref}>
  > >   -  <Button />
  > >   -</RootRef>
  > +<Button re
  > ```

Select

- Change the default variant from `standard` to `outlined`. Standard has been removed from the Material Design guidelines. Change the default variant from `standard` to `outlined`. Standard has been removed from the Material Design guidelines. If you are composing the Select with a form control component, you only need to update `FormControl`, the select inherits the variant from its context.

  > ✅ *This is handled in [variant-prop codemod](), read the details before running this codemod.*
  >
  > ```
  > -<Select value="Standard" />
  > -<Select value="Outlined" variant="outlined" />
  > +<Select value="Standard" variant="standard" />
  > +<Select value="Outlined" />
  > ```

- Remove the `labelWidth` prop. The `label` prop now fulfills the same purpose, using CSS layout instead of JavaScript measurement to render the gap in the outlined. The TextField already handles it by default.

  ```diff
  -<Select variant="outlined" labelWidth={20} />
  +<Select variant="outlined" label="Gender" />
  ```

- Merge the `selectMenu` slot into `select`. Slot `selectMenu` was redundant. Merge the `selectMenu` slot into `select`. Slot `selectMenu` was redundant. The `root` slot is no longer applied to the select, but to the root.

  ```diff
  -<Select classes={{ root: 'class1', select: 'class2', selectMenu: 'class3' }}
  />
  +<Select classes={{ select: 'class1 class2 class3' }} />
  ```

- The `event` in `onChange` is now a synthetic, native `Event` not a React event.

  ```diff
  -<Select onChange={(event: React.SyntheticEvent, value: unknown) => {}} />
  +<Select onChange={(event: Event, value: unknown) => {}} />
  ```

  This was necessary to prevent overriding of `event.target` of the events that caused the change.

Skeleton

- Move the component from the lab to the core. The component is now stable. The component is now stable.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -import Skeleton from '@mui/lab/Skeleton';
  > +import Skeleton from '@mui/material/Skeleton';
  > ```

- Rename `circle` to `circular` and `rect` to `rectangular` for consistency:

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```
  > -<Skeleton variant="circle" />
  > -<Skeleton variant="rect" />
  > -<Skeleton classes={{ circle: 'custom-circle-classname', rect: 'custom-rect-classname',  }} />
  > +<Skeleton variant="circular" />
  > +<Skeleton variant="rectangular" />
  > +<Skeleton classes={{ circular: 'custom-circle-classname', rectangular: 'custom-rect-classname',  }} />
  > ```

Slider

- The `event` in `onChange` is now a synthetic, native `Event`, not a React event.

  ```
  -<Slider onChange={(event: React.SyntheticEvent, value: unknown) => {}} />
  +<Slider onChange={(event: Event, value: unknown) => {}} />
  ```

  This was necessary to prevent overriding of `event.target` of the events that caused the change.

- The `ValueLabelComponent` and `ThumbComponent` prop is now part of the `components` prop.

  ```
   <Slider
  -  ValueLabelComponent={CustomValueLabel}
  -  ThumbComponent={CustomThumb}
  +  components={{
  +    ValueLabel: CustomValueLabel,
  +    Thumb: CustomThumb,
  +  }}
   />
  ```

- Rework the CSS to match the latest Material Design guidelines and make custom styles more intuitive. See

documentation. See documentation.

  You can reduce the density of the slider, closer to v4 with the `size="small"` prop.

Snackbar

- The notification now displays at the bottom left on large screens. This better matches the behavior of Gmail, Google Keep, material.io, etc. You can restore the previous behavior with:

  ```diff
  -<Snackbar />
  +<Snackbar anchorOrigin={{ vertical: 'bottom', horizontal: 'center' }} />
  ```

- The onE* transition props were removed. Use TransitionProps instead. Use TransitionProps instead.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```diff
  >  <Snackbar
  > -  onEnter={onEnter}
  > -  onEntered={onEntered}
  > -  onEntering={onEntering}
  > -  onExit={onExit}
  > -  onExited={onExited}
  > -  onExiting={onExiting}
  > +  TransitionProps={{
  > +    onEnter,
  > +    onEntered,
  > +    onEntering,
  > +    onExit,
  > +    onExited,
  > +    onExiting,
  > +  }}
  >  >
  > ```

SpeedDial

- Move the component from the lab to the core. The component is now stable. The component is now stable.

  > ✅ *This is handled in the preset-safe codemod.*
  >
  > ```diff
  > -import SpeedDial from '@mui/lab/SpeedDial';
  > -import SpeedDialAction from '@mui/lab/SpeedDialAction';
  > -import SpeedDialIcon from '@mui/lab/SpeedDialIcon';
  > +import SpeedDial from '@mui/material/SpeedDial';
  > +import SpeedDialAction from '@mui/material/SpeedDialAction';
  > +import SpeedDialIcon from '@mui/material/SpeedDialIcon';
  > ```

Stepper

- The root component (Paper) was replaced with a div. Stepper no longer has elevation, nor inherits Paper's props. This change is meant to encourage composition. Stepper no longer has elevation, nor inherits Paper's props. This change is meant to encourage composition.

  ```diff
  +<Paper square elevation={2}>
  -  <Stepper elevation={2}>
  ```

```diff
+  <Stepper>
     <Step>
       <StepLabel>Hello world</StepLabel>
     </Step>
   </Stepper>
+<Paper>
```

- Remove the built-in 24px padding.

```diff
-<Stepper>
+<Stepper style={{ padding: 24 }}>
   <Step>
     <StepLabel>Hello world</StepLabel>
   </Step>
 </Stepper>
```

SvgIcon

- The default value of `fontSize` was changed from `default` to `medium` for consistency. The default value of `fontSize` was changed from `default` to `medium` for consistency. In the unlikey event that you were using the value `default`, the prop can be removed:

```diff
-<SvgIcon fontSize="default">
+<SvgIcon>
   <path d="M10 20v-6h4v6h5v-8h3L12 3 2 12h3v8z" />
 </SvgIcon>
```

Switch

- Deprecate the second argument from `onChange`. Deprecate the second argument from `onChange`. You can pull out the checked state by accessing `event.target.checked`.

```diff
function MySwitch() {
- const handleChange = (event: React.ChangeEvent<HTMLInputElement>, checked: boolean) => {
+ const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
+   const checked = event.target.checked;
  };

  return <Switch onChange={handleChange} />;
}
```

- The switch color prop is now "primary" by default. To continue using the "secondary" color, you must explicitly indicate `secondary`. This brings the switch closer to the Material Design guidelines. To continue using the "secondary" color, you must explicitly indicate `secondary`. This brings the switch closer to the Material Design guidelines.

```diff
-<Switch />
+<Switch color="secondary" />
```

- The component doesn't have `.MuiIconButton-root` and `.MuiIconButton-label` class names anymore, target `.MuiButtonBase-root` instead.

```diff
 <span class="MuiSwitch-root">
-  <span class="MuiIconButton-root MuiButtonBase-root MuiSwitch-switchBase PrivateSwitchBase-root">
-    <span class="MuiIconButton-label">
-      <input class="MuiSwitch-input PrivateSwitchBase-input">
+  <span class="MuiButtonBase-root MuiSwitch-switchBase PrivateSwitchBase-root">
+    <span class="MuiSwitch-input PrivateSwitchBase-input">
```

Table

- Rename the `default` value of the `padding` prop to `normal`.

```diff
-<Table padding="default" />
-<TableCell padding="default" />
+<Table padding="normal" />
+<TableCell padding="normal" />
```

TablePagination

- The customization of the table pagination's actions labels must be done with the `getItemAriaLabel` prop. This increases consistency with the `Pagination` component. This increases consistency with the `Pagination` component.

```diff
 <TablePagination
-  backIconButtonText="Avant"
-  nextIconButtonText="Après"
+  getItemAriaLabel={…}
```

- Rename `onChangeRowsPerPage` to `onRowsPerPageChange` and `onChangePage` to `onPageChange` due to API consistency.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```diff
>  <TablePagination
> -  onChangeRowsPerPage={()=>{}}
> -  onChangePage={()=>{}}
> +  onRowsPerPageChange={()=>{}}
> +  onPageChange={()=>{}}
> ```

- Separate classes for different table pagination labels. This allows simpler customizations. This allows simpler customizations.

```diff
 <TablePagination
-  classes={{ caption: 'foo' }}
+  classes={{ selectLabel: 'foo', displayedRows: 'foo' }}
 />
```

- Move the custom class on `input` to `select`. The `input` key is being applied on another element. The `input` key is being applied on another element.

```diff
 <TablePagination
-  classes={{ input: 'foo' }}
+  classes={{ select: 'foo' }}
 />
```

Tabs

- Change the default `indicatorColor` and `textColor` prop values to "primary". This is done to match the most common use cases with Material Design. This is done to match the most common use cases with Material Design.

```diff
-<Tabs />
+<Tabs indicatorColor="primary" textColor="inherit" />
```

- TypeScript: The `event` in `onChange` is no longer typed as a `React.ChangeEvent` but `React.SyntheticEvent`.

```diff
-<Tabs onChange={(event: React.ChangeEvent<{}>, value: unknown) => {}} />
+<Tabs onChange={(event: React.SyntheticEvent, value: unknown) => {}} />
```

- The API that controls the scroll buttons has been split it in two props.
  - The `scrollButtons` prop controls when the scroll buttons are displayed depending on the space available.
  - The `allowScrollButtonsMobile` prop removes the CSS media query that systematically hide the scroll buttons on mobile.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```diff
> -<Tabs scrollButtons="on" />
> -<Tabs scrollButtons="desktop" />
> -<Tabs scrollButtons="off" />
> +<Tabs scrollButtons allowScrollButtonsMobile />
> +<Tabs scrollButtons />
> +<Tabs scrollButtons={false} />
> ```

Tab

- Tab `minWidth` changed from `72px` => `90px` (without media-query) according to material-design spec

- Tab `maxWidth` changed from `264px` => `360px` according to material-design spec
- `span` element that wraps children has been removed. `wrapper` classKey is also removed. More details about this change.

```
 <button class="MuiTab-root">
-  <span class="MuiTab-wrapper">
     {icon}
     {label}
-  </span>
 </button>
```

TextField

- Change the default variant from `standard` to `outlined`. Standard has been removed from the Material Design guidelines. Standard has been removed from the Material Design guidelines.

> ✅ *This is handled in variant-prop codemod, read the details before running this codemod.*
>
> ```
> -<TextField value="Standard" />
> -<TextField value="Outlined" variant="outlined" />
> +<TextField value="Standard" variant="standard" />
> +<TextField value="Outlined" />
> ```

- Rename `rowsMax` prop with `maxRows` for consistency with HTML attributes.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -<TextField rowsMax={6}>
> +<TextField maxRows={6}>
> ```

- Better isolate the fixed textarea height behavior to the dynamic one. You need to use the `minRows` prop in the following case: You need to use the `minRows` prop in the following case:

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -<TextField rows={2} maxRows={5} />
> +<TextField minRows={2} maxRows={5} />
> ```

- Change ref forwarding expectations on custom `inputComponent`. The component should forward the `ref` prop instead of the `inputRef` prop. The component should forward the `ref` prop instead of the `inputRef` prop.

```
-function NumberFormatCustom(props) {
-  const { inputRef, onChange, ...other } = props;
+const NumberFormatCustom = React.forwardRef(function NumberFormatCustom(
+  props,
+  ref,
+) {
```

```
  const { onChange, ...other } = props;

  return (
    <NumberFormat
      {...other}
-     getInputRef={inputRef}
+     getInputRef={ref}
```

- Rename `marginDense` and `inputMarginDense` classes to `sizeSmall` and `inputSizeSmall` to match the prop.

```
-<Input margin="dense" />
+<Input size="small" />
```

- Set the InputAdornment `position` prop to `start` or `end`. Use `start` if used as the value of the `startAdornment` prop. Use `end` if used as the value of the `endAdornment` prop. Use `start` if used as the value of the `startAdornment` prop. Use `end` if used as the value of the `endAdornment` prop.

```
-<TextField startAdornment={<InputAdornment>kg</InputAdornment>} />
-<TextField endAdornment={<InputAdornment>kg</InputAdornment>} />
+<TextField startAdornment={<InputAdornment
position="start">kg</InputAdornment>} />
+<TextField endAdornment={<InputAdornment position="end">kg</InputAdornment>}
/>
```

TextareaAutosize

- Remove the `rows` prop, use the `minRows` prop instead. This change aims to clarify the behavior of the prop. This change aims to clarify the behavior of the prop.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -<TextareaAutosize rows={2} />
> +<TextareaAutosize minRows={2} />
> ```

- Rename `rowsMax` prop with `maxRows` for consistency with HTML attributes.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -<TextareAutosize rowsMax={6}>
> +<TextareAutosize maxRows={6}>
> ```

- Rename `rowsMin` prop with `minRows` for consistency with HTML attributes.

> ✅ *This is handled in the preset-safe codemod.*
>
> ```
> -<TextareAutosize rowsMin={1}>
> +<TextareAutosize minRows={1}>
> ```

ToggleButton

- Move the component from the lab to the core. The component is now stable. The component is now stable.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```
  > -import ToggleButton from '@mui/lab/ToggleButton';
  > -import ToggleButtonGroup from '@mui/lab/ToggleButtonGroup';
  > +import ToggleButton from '@mui/material/ToggleButton';
  > +import ToggleButtonGroup from '@mui/material/ToggleButtonGroup';
  > ```

- The `span` element that wraps children has been removed. The `label` classKey is also removed. `span` element that wraps children has been removed. `label` classKey is also removed. More details about [this change](#).

  ```
   <button class="MuiToggleButton-root">
  -  <span class="MuiToggleButton-label">
       {children}
  -  </span>
   </button>
  ```

Tooltip

- Tooltips are now interactive by default.

  The previous default behavior failed [success criterion 1.4.3 ("hoverable") in WCAG 2.1](#). To reflect the new default value, the prop was renamed to `disableInteractive`. If you want to restore the old behavior (thus not reaching level AA), you can apply the following diff: To reflect the new default value, the prop was renamed to `disableInteractive`. If you want to restore the old behavior (thus not reaching level AA), you can apply the following diff:

  ```
  -<Tooltip>
  +<Tooltip disableInteractive>

  # Interactive tooltips no longer need the `interactive` prop.
  -<Tooltip interactive>
  +<Tooltip>
  -<Tooltip interactive>
  +<Tooltip>
  ```

Typography

- Remove the `srOnly` variant. Remove the `srOnly` variant. You can use the `visuallyHidden` utility in conjunction with the `sx` prop instead.

  ```
  +import { visuallyHidden } from '@mui/utils';
  ```

```diff
-<Typography variant="srOnly">Create a user</Typography>
+<span style={visuallyHidden}>Create a user</span>
```

- The following `classes` and style overrides keys were removed: "colorInherit", "colorPrimary", "colorSecondary", "colorTextPrimary", "colorTextSecondary", "colorError", "displayInline" and "displayBlock". These props are now considered part of the system, not on the `Typography` component itself. If you still wish to add overrides for them, you can use the `theme.components.MuiTypography.variants` options. For example These props are now considered part of the system, not on the `Typography` component itself. If you still wish to add overrides for them, you can use the callback as a value in `styleOverrides`. For example:

```diff
const theme = createTheme({
  components: {
    MuiTypography: {
-     styleOverrides: {
-       colorSecondary: {
-         marginTop: '20px',
-       },
-     },
+     variants: {
+       props: { color: "secondary" },
+       style: {
+         marginTop: '20px',
+       },
+     }],
    },
  },
});
```

Theme

- The default background color is now `#fff` in light mode and `#121212` in dark mode. This matches the Material Design guidelines. This matches the Material Design guidelines.

- Breakpoints are now treated as values instead of ranges. The behavior of `down(key)` was changed to define a media query below the value defined by the corresponding breakpoint (exclusive), rather than the breakpoint above. `between(start, end)` was also updated to define a media query for the values between the actual values of start (inclusive) and end (exclusive). When using the `down()` breakpoints utility you need to update the breakpoint key by one step up. When using the `between(start, end)` the end breakpoint should also be updated by one step up. The behavior of `down(key)` was changed to define a media query below the value defined by the corresponding breakpoint (exclusive), rather than the breakpoint above. `between(start, end)` was also updated to define a media query for the values between the actual values of start (inclusive) and end (exclusive). When using the `down()` breakpoints utility you need to update the breakpoint key by one step up. When using the `between(start, end)` the end breakpoint should also be updated by one step up.

  > ✅ *This is handled in the preset-safe codemod.*

  Here are some examples of the changes required:

```diff
-theme.breakpoints.down('sm') // '@media (max-width:959.95px)' - [0, sm + 1)
=> [0, md)
+theme.breakpoints.down('md') // '@media (max-width:959.95px)' - [0, md)
```

```diff
-theme.breakpoints.between('sm', 'md') // '@media (min-width:600px) and (max-
width:1279.95px)' - [sm, md + 1) => [0, lg)
+theme.breakpoints.between('sm', 'lg') // '@media (min-width:600px) and (max-
width:1279.95px)' - [0, lg)
```

```diff
-theme.breakpoints.between('sm', 'xl') // '@media (min-width:600px)'
+theme.breakpoints.up('sm') // '@media (min-width:600px)'
```

The same should be done when using the `Hidden` component:

```diff
-<Hidden smDown>{...}</Hidden> // '@media (min-width:600px)'
+<Hidden mdDown>{...}</Hidden> // '@media (min-width:600px)'
```

- The default breakpoints were changed to better match the common use cases. They also better match the Material Design guidelines. Read more about the change They also better match the Material Design guidelines. Read more about the change

```diff
{
  xs: 0,
  sm: 600,
- md: 960,
+ md: 900,
- lg: 1280,
+ lg: 1200,
- xl: 1920,
+ xl: 1536,
}
```

If you prefer the old breakpoint values, use the snippet below.

```js
import { createTheme } from '@mui/material/styles';

const theme = createTheme({
  breakpoints: {
    values: {
      xs: 0,
      sm: 600,
      md: 960,
      lg: 1280,
      xl: 1920,
    },
  },
});
```

- The `theme.breakpoints.width` utility was removed because it's redundant. Use `theme.breakpoints.values` to get the same values. Use `theme.breakpoints.values` to get the same values.

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```
  > -theme.breakpoints.width('md')
  > +theme.breakpoints.values.md
  > ```

- The signature of `theme.palette.augmentColor` helper has changed:

  ```diff
  -theme.palette.augmentColor(red);
  +theme.palette.augmentColor({ color: red, name: 'brand' });
  ```

- The `theme.typography.round` helper was removed because it was no longer used. If you need it, use the function below: If you need it, use the function below:

  > ✅ *This is handled in the [preset-safe codemod](#).*
  >
  > ```
  >   function round(value) {
  >   return Math.round(value * 1e5) / 1e5;
  > }
  > ```

`@mui/types`

- Rename the exported `Omit` type in `@mui/types`. The module is now called `DistributiveOmit`. The change removes the confusion with the built-in `Omit` helper introduced in TypeScript v3.5. The built-in `Omit`, while similar, is non-distributive. This leads to differences when applied to union types. [See this StackOverflow answer for further details](#).

  ```diff
  -import { Omit } from '@mui/types';
  +import { DistributiveOmit } from '@mui/types';
  ```

Migrate theme's `styleOverrides` to emotion

Although your style overrides defined in the theme may partially work, there is an important difference on how the nested elements are styled. The `$` syntax used with JSS will not work with Emotion. You need to replace those selectors with a valid class selector.

Replace state class names

```
 +import { outlinedInputClasses } from '@mui/material/OutlinedInput';

const theme = createTheme({
  components: {
    MuiOutlinedInput: {
      styleOverrides: {
```

```diff
        root: {
-          '& $notchedOutline': {
+          [`& .${outlinedInputClasses['notchedOutline']}`]: {
            borderWidth: 1,
          }
        }
      }
    }
  }
});
```

Replace nested classes selectors with global class names

```diff
const theme = createTheme({
  components: {
    MuiOutlinedInput: {
      styleOverrides: {
        root: {
-          '& $notchedOutline': {
+          '& .MuiOutlinedInput-notchedOutline': {
            borderWidth: 1,
          }
        }
      }
    }
  }
});
```

> *注意：对于每个组件我们导出一个包含该组件所有嵌套类的* `[component]类` *常数。 您可以依靠这个而不是硬编码类。*

```diff
import Typography from '@mui/material/Typography';
-import makeStyles from '@mui/styles/makeStyles';
+import { styled } from '@mui/material/styles';

-const useStyles = makeStyles((theme) => ({
-  root: {
-    display: 'flex',
-    alignItems: 'center',
-    backgroundColor: theme.palette.primary.main
-  },
-  cta: {
-    borderRadius: theme.shape.radius
-  },
-  content: {
-    color: theme.palette.common.white,
-    fontSize: 16,
-    lineHeight: 1.7
-  },
-}))
```

```diff
+const PREFIX = 'MyCard';
+const classes = {
+  root: `${PREFIX}-root`,
+  cta: `${PREFIX}-cta`,
+  content: `${PREFIX}-content`,
+}
+const Root = styled('div')(({ theme }) => ({
+  [`&.${classes.root}`]: {
+    display: 'flex',
+    alignItems: 'center',
+    backgroundColor: theme.palette.primary.main
+  },
+  [`& .${classes.cta}`]: {
+    borderRadius: theme.shape.radius
+  },
+  [`& .${classes.content}`]: {
+    color: theme.palette.common.white,
+    fontSize: 16,
+    lineHeight: 1.7
+  },
+}))

 export const MyCard = () => {
-  const classes = useStyles();
   return (
-    <div className={classes.root}>
+    <Root className={classes.root}>
       {/* The benefit of this approach is that the code inside Root stays the same.
*/}
       <Typography className={classes.content}>...</Typography>
       <Button className={classes.cta}>Go</Button>
-    </div>
+    </Root>
   )
 }
```

Take a look at the whole list of global state classnames available.

Migrate from JSS

This is the last step in the migration process to remove `@mui/styles` package from your codebase. We can use one of these two options, by order of preference: You can use one of these two options, by order of preference:

1. 1. Use `styled` or `sx` API

Codemod

We provide a codemod to help migrate JSS styles to `styled` API, but this approach **increases the CSS specificity**.

> *Note: Usually, you wouldn't write the styles like this if you were to write them manually. However, this is the best trasnformation that can be created via codemod we could come up with. So, if you want to refine them later, you can refer to the examples shown in the sections below.*

```
npx @mui/codemod v5.0.0/jss-to-styled <path>
```

Example transformation:

```
 const theme = createTheme({
  components: {
    MuiOutlinedInput: {
      styleOverrides: {
        root: {
-         '&$focused': {
+         '&.Mui-focused': {
            borderWidth: 1,
          }
        }
      }
    }
  }
}); */}
       <Typography className={classes.content}>...</Typography>
       <Button className={classes.cta}>Go</Button>
-     </div>
+     </Root>
   )
 }
```

> 💡 *你应该按照文件的小块运行这个 codemod，然后检查更改，因为在某些情况下，你可能需要在转换后调整代码(这个 codemod 不会涵盖所有案例)。*

手动

We recommend `sx` API over `styled` when you have to create responsive styles or needs minor CSS overrides. Read more about `sx` . Read more about `sx` .

```
 import Chip from '@mui/material/Chip';
-import makeStyles from '@mui/styles/makeStyles';
+import { styled } from '@mui/material/styles';

-const useStyles = makeStyles((theme) => ({
-  wrapper: {
-    display: 'flex',
-  },
-  chip: {
-    padding: theme.spacing(1, 1.5),
-    boxShadow: theme.shadows[1],
-  }
-}))
+const Root = styled('div')({
+  display: 'flex',
+})
```

```diff
 function App() {
-  const classes = useStyles();
   return (
-    <div>
-      <Chip className={classes.chip} label="Chip" />
-    </div>
+    <Root>
+      <Chip label="Chip" sx={{ py: 1, px: 1.5, boxShadow: 1 }} />
+    </Root>
   )
 }
```

In some cases, you might want to create multiple styled components in a file instead of increasing CSS specificity. for example: for example:

```diff
-import makeStyles from '@mui/styles/makeStyles';
+import { styled } from '@mui/material/styles';

-const useStyles = makeStyles((theme) => ({
-  root: {
-    display: 'flex',
-    alignItems: 'center',
-    borderRadius: 20,
-    background: theme.palette.grey[50],
-  },
-  label: {
-    color: theme.palette.primary.main,
-  }
-}))
+const Root = styled('div')(({ theme }) => ({
+  display: 'flex',
+  alignItems: 'center',
+  borderRadius: 20,
+  background: theme.palette.grey[50],
+}))

+const Label = styled('span')(({ theme }) => ({
+  color: theme.palette.primary.main,
+}))

 function Status({ label }) {
-  const classes = useStyles();
   return (
-    <div className={classes.root}>
-      {icon}
-      <span className={classes.label}>{label}</span>
-    </div>
+    <Root>
+      {icon}
+      <Label>{label}</Label>
+    </Root>
```

```
  )
}
```

> ***注意：*** *[https://siriwatk.dev/tool/jssto-styploed](https://siriwatk.dev/tool/jssto-styled) 是一个有助于将 JS 转换成多种样式组件而不增加 CSS 针对性的工具。 （这个工具**不是**由 MUI 维护）*

2. Use tss-react

> *注意：这个设置可能无法在所有情况下工作。*

The API is similar to JSS `makeStyles` but works with emotion. It is also features a much better TypeScript support than v4's `makeStyles` .

If you want to apply styles to components by importing a css file, you need to bump up specificity in order to always select the correct component. Consider the following example.

```
2. Use <a href="https://github.com/garronej/tss-react">tss-react</a>
```

...and to edit your providers:

```
 import * as React from 'react';
 import { CacheProvider } from '@emotion/react';
 import createCache from '@emotion/cache';

 const cache = createCache({
   key: 'css',
+  prepend: true,
 });

 export default function PlainCssPriority() {
   return (
     <CacheProvider value={cache}>
       {/* Your component tree. Now you can override MUI's styles. */}
     </CacheProvider>
   );
 }
```

Then here is one example:

```
 The <code>withStyles</code> JSS utility is no longer exported from
<code>@mui/material/styles</code>. You can use <code>@mui/styles/withStyles</code>
instead. Make sure to add a <code>ThemeProvider</code> at the root of your
application, as the <code>defaultTheme</code> is no longer available. If you are
using this utility together with <code>@mui/material</code>, you should use the
<code>ThemeProvider</code> component from <code>@mui/material/styles</code> instead.
```

JSS utility is no longer exported from @mui/material/styles. You can use @mui/styles/withStyles instead. Make sure to add a ThemeProvider at the root of your application, as the defaultTheme is no longer available. If you are using this utility together with @mui/material, you should use the ThemeProvider component from @mui/material/styles instead.

If you were using the `$` syntax, the transformation would look like this:

```
 The <code>makeStyles</code> JSS utility is no longer exported from
<code>@mui/material/styles</code>. You can use <code>@mui/styles/makeStyles</code>
instead. Make sure to add a <code>ThemeProvider</code> at the root of your
application, as the <code>defaultTheme</code> is no longer available. If you are
using this utility together with <code>@mui/material</code>, it's recommended that
you use the <code>ThemeProvider</code> component from
<code>@mui/material/styles</code> instead.
```

JSS utility is no longer exported from @mui/material/styles. You can use @mui/styles/makeStyles instead. Make sure to add a ThemeProvider at the root of your application, as the defaultTheme is no longer available. If you are using this utility together with @mui/material, it's recommended that you use the ThemeProvider component from @mui/material/styles instead.

> 📝 *在进行下一步前，请确保您的程序仍然可以正确**运行**没有报错并且已经**应用了**更改。*

Now, a comprehensive example using both the `$` syntax, `useStyles()` parameters and an explicit name for the stylesheet.

```
-import clsx from 'clsx';
-import { makeStyles, createStyles } from '@material-ui/core/styles';
+import { makeStyles } from 'tss-react/mui';

-const useStyles = makeStyles((theme) => createStyles<
-  'root' | 'small' | 'child', { color: 'primary' | 'secondary' }
->({
+const useStyles = makeStyles<
+  { color: 'primary' | 'secondary' }, 'child' | 'small'
+>({ name: 'App' })((theme, { color }, classes) => ({
-  root: ({ color })=> ({
+  root: {
     padding: 30,
-    '&:hover .child': {
+    [`&:hover .${classes.child}`]: {
       backgroundColor: theme.palette[color].main,
     }
-  }),
+  },
  small: {},
  child: {
    border: '1px solid black',
    height: 50,
-    '&.small': {
+    [`&.${classes.small}`]: {
        height: 30
    }
  }
-}, { name: 'App' });
+}));
```

```
 function App() {
-  const classes = useStyles({ color: 'primary' });
+  const { classes, cx } = useStyles({ color: 'primary' });

   return (
     <div className={classes.root}>
       <div className={classes.child}>
         The Background take the primary theme color when the mouse hovers the
parent.
       </div>
-      <div className={clsx(classes.child, classes.small)}>
+      <div className={cx(classes.child, classes.small)}>
         The Background take the primary theme color when the mouse hovers the
parent.
         I am smaller than the other child.
       </div>
     </div>
   );
 }

 export default App;
```

> ***WARNING**: You should drop `clsx` in favor of `cx`. The key advantage of `cx` is that it detects emotion generated class names ensuring styles are overwritten in the correct order. **Note**: To ensure that your class names always includes the actual name of your components, you can provide the `name` as an implicitly named key (`name: { App }`). See doc.*

`withStyles()`

This component was removed. You can get a reference to the underlying DOM node of our components via `ref` prop. The component relied on `ReactDOM.findDOMNode` which is deprecated in `React.StrictMode`.

> ***注意：这个库不是由 MUI 维护。** 如果您对此有任何问题，请在 tss-react repository 中创建一个问题。*

```
// Ex. 1 ✅ html tag works since it is a DOM
<Fade in>
  <div>
    <CustomComponent />
  </div>
</Fade>

// Ex. 2 ❌ This will cause error. don't use Fragment as a child
<Fade in>
  <React.Fragment>
    <CustomComponent />
  </React.Fragment>
</Fade>;

// Ex. 3 ❌ This will cause error because `CustomComponent` does not forward ref to
DOM
function CustomComponent() {
  return <div>...</div>;
}

<Fade in>
  <CustomComponent />
</Fade>;
```

将 theme 的 `styleOverrides` 迁移至 emotion

Since `makeStyles` is now exported from `@mui/styles` package which does not know about `Theme` in the core package. To fix this, you need to augment the `DefaultTheme` (empty object) in `@mui/styles` with `Theme` from the core. Read more about module augmentation

```
// ✅ Fixed by using `React.forwardRef` and pass to DOM.
const CustomComponent = React.forwardRef(function CustomComponent(props, ref) {
  return (
    <div ref={ref}>
      ...
    </div>
  )
})

<Fade in>
  <CustomComponent />
</Fade>
```

You may end up with eslint warnings like this one if you deconstruct more that one item.
Don't hesitate to disable `eslint(prefer-const)`, like this in a regular project, or like this in a CRA.

Note: `tss-react` is **not maintained** by MUI. If you have any question about how to setup SSR (Next.js) or if you are wondering how to customize the `theme` object please refer to `tss-react`'s documentation, the Mui integration section in particular. You can also submit an issue for any bug or feature request and start a discussion if you need help.

💡 Once you migrate all of the styling, remove unnecessary `@mui/styles` by

```
npm uninstall @mui/styles

// or with `yarn`
yarn remove @mui/styles
```

> *The `createStyles` function from `@mui/material/styles` was moved to the one exported from `@mui/styles`. It is necessary for removing the dependency to `@mui/styles` in the core package.*

CSS 特性

If you want to apply styles to components by importing a css file, you need to bump up specificity in order to always select the correct component. Consider the following example:

```
import './style.css';
import Chip from '@mui/material/Chip';
```

```
const ChipWithGreenIcon = () => (
  <Chip
    classes={{ deleteIcon: 'green' }}
    label="delete icon is green"
    onDelete={() => {}}
  />
);
```

In this example, in order to correctly apply a particular style to the delete icon of `Chip`, you need to bump the specificity as shown below:

```
.MuiChip-root .green {
  color: green;
}
```

The following will not correctly apply the style to the delete icon:

```
.green {
  color: green;
}
```

Troubleshooting

Storybook emotion with v5

If your project uses Storybook v6.x, you will need to update `.storybook/main.js` webpack config to use the most recent version of emotion.

```
// .storybook/main.js

const path = require('path');
const toPath = (filePath) => path.join(process.cwd(), filePath);

module.exports = {
  webpackFinal: async (config) => {
    return {
      ...config,
      resolve: {
        ...config.resolve,
        alias: {
          ...config.resolve.alias,
          '@emotion/core': toPath('node_modules/@emotion/react'),
          'emotion-theming': toPath('node_modules/@emotion/react'),
        },
      },
    };
  },
};
```

and update `.storybook/preview.js` (otherwise, the "Docs" tab in storybook will display empty page)

```
// .storybook/preview.js

import { ThemeProvider, createTheme } from '@mui/material/styles';
import { ThemeProvider as Emotion10ThemeProvider } from 'emotion-theming';

const defaultTheme = createTheme(); // or your custom theme

const withThemeProvider = (Story, context) => {
  return (
    <Emotion10ThemeProvider theme={defaultTheme}>
      <ThemeProvider theme={defaultTheme}>
        <Story {...context} />
      </ThemeProvider>
    </Emotion10ThemeProvider>
  );
};

export const decorators = [withThemeProvider];

// ...other storybook exports
```

Tested versions

```
{
  "@storybook/react": "6.3.8",
  "@storybook/addon-docs": "6.3.8",
  "@emotion/react": "11.4.1",
  "@emotion/styled": "11.3.0",
  "@mui/material": "5.0.2"
}
```

> *Note: This setup is a workaround and might not work in all cases.*

For more details, checkout these issues on GitHub.

- https://github.com/storybookjs/storybook/issues/16099
- https://github.com/mui/material-ui/issues/24282#issuecomment-796755133

Cannot read property `scrollTop` of null

This error comes from `Fade`, `Grow`, `Slide`, `Zoom` components due to missing DOM Node.

You need to make sure that the children forward ref to DOM for custom component.

```
// Ex. 1 ✅ html tag works since it is a DOM
<Fade in>
  <div>
    <CustomComponent />
  </div>
</Fade>
```

```
// Ex. 2 ❌ This will cause error. don't use Fragment as a child
<Fade in>
  <React.Fragment>
    <CustomComponent />
  </React.Fragment>
</Fade>;

// Ex. 3 ❌ This will cause error because `CustomComponent` does not forward ref to
DOM
function CustomComponent() {
  return <div>...</div>;
}

<Fade in>
  <CustomComponent />
</Fade>;
```

```
// ✅ Fixed by using `React.forwardRef` and pass to DOM.
const CustomComponent = React.forwardRef(function CustomComponent(props, ref) {
  return (
    <div ref={ref}>
      ...
    </div>
  )
})

<Fade in>
  <CustomComponent />
</Fade>
```

For more details, checkout this issue on GitHub.

[Types] Property "palette", "spacing" does not exist on type 'DefaultTheme'

Since `makeStyles` is now exported from `@mui/styles` package which does not know about `Theme` in the core package. To fix this, you need to augment the `DefaultTheme` (empty object) in `@mui/styles` with `Theme` from the core. Read more about module augmentation

TypeScript Project

Put this snippet to your theme file:

```
// it could be your App.tsx file or theme file that is included in your
tsconfig.json
import { Theme } from '@mui/material/styles';

declare module '@mui/styles/defaultTheme' {
  // eslint-disable-next-line @typescript-eslint/no-empty-interface (remove this
line if you don't have the rule enabled)
```

```
  interface DefaultTheme extends Theme {}
}
```

Javascript Project

If your IDE (ex. VSCode) is able to infer types from `d.ts` file, create `index.d.ts` in your `src` folder with this snippet:

```
// index.d.ts
declare module "@mui/private-theming" {
  import type { Theme } from "@mui/material/styles";

  interface DefaultTheme extends Theme {}
}
```

[Jest] SyntaxError: Unexpected token 'export'

`@mui/material/colors/red` is considered private since v1.0.0. You should replace the import, more details about this error.

You can use this codemod (**recommended**) to fix all the import in your project:

```
npx @mui/codemod v5.0.0/optimal-imports <path>
```

or fix it manually like this:

```diff
-import red from '@mui/material/colors/red';
+import { red } from '@mui/material/colors';
```

makeStyles - TypeError：无法读取未定义的属性“drawer”

This error occurs when calling `useStyles` (result of `makeStyles`) or `withStyles` outside of `<ThemeProvider>` scope like this:

```
import * as React from 'react';
import { ThemeProvider, createTheme } from '@mui/material/styles';
import makeStyles from '@mui/styles/makeStyles';
import Card from '@mui/material/Card';
import CssBaseline from '@mui/material/CssBaseline';

const useStyles = makeStyles((theme) => ({
  root: {
    display: 'flex',
    backgroundColor: theme.palette.primary.main,
    color: theme.palette.common.white,
  },
}));

const theme = createTheme();
```

```
function App() {
  const classes = useStyles(); // ❌ called outside of ThemeProvider
  return (
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <Card className={classes.root}>...</Card>
    </ThemeProvider>
  );
}

export default App;
```

You can fix by moving `useStyles` inside another component so that it is called under `<ThemeProvider>`.

```
// ...imports

function AppContent(props) {
  const classes = useStyles(); // ✅ This is safe because it is called inside
ThemeProvider
  return <Card className={classes.root}>...</Card>;
}

function App(props) {
  return (
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <AppContent {...props} />
    </ThemeProvider>
  );
}

export default App;
```

TypeError: Cannot read properties of undefined (reading 'pxToRem')

The root cause of this error comes from accessing empty theme. Make sure that you have follow these checklist: Make sure that you have follow these checklist:

- `styled` should only be imported from `@mui/material/styles` (If you are not using standalone `@mui/system`)

  ```
  import { styled } from '@mui/material/styles';
  ```

- Make sure that no `useStyles` is called outside of `<ThemeProvider>`. If you have, consider fixing it like this suggestion If you have, consider fixing it like this suggestion

For more details, checkout this issue

Styles broken after migrating to v5

There are two reasons why the styles of the components may be broken after you finished with all the steps in the previous sections.

First, check if you have configured the `StyledEngineProvider` correct as shown in the Style library section.

If the `StyledEngineProvider` is already used at the top of your application and the styles are still broken, it may be the case that you still have `@material-ui/core` in your application. It may be coming from some of the dependencies that you have, that still depend on `@material-ui/core` (v4). It may be coming from some of the dependencies that you have, that still depend on `@material-ui/core` (v4).

The easiest way to check this is to run `npm ls @material-ui/core` (or `yarn why @material-ui/core`) which will give you the necessary information.

Here is one example:

```
$ npm ls @material-ui/core
project@0.1.0 /path/to/project
└─┬  @mui/x-data-grid@4.0.0
  └──  @material-ui/core@4.12.3
```

You can notice based on the output above that `@material-ui/core` is a dependency of `@mui/x-data-grid`. You can notice based on the output above that `@material-ui/core` is a dependency of `@mui/x-data-grid`. In this specific example, you need to bump the version of `@mui/x-data-grid` to version 5 so that it depends on `@mui/material` instead.