Icons 图标

我们提供了一些建议和指导，能够帮助您在 Material-UI 中使用图标。

Material-UI 通过以下三种方式来支持图标的使用：

1. 您可以将标准的 Material Design 图标 导出为 React 组件 (SVG icons)。
2. 或者可以将自定义的 SVG 图标通过 SvgIcon 组件来包装成一个 React 组件。
3. 或者可以将自定义的 font 图标通过 Icon 组件来包装成一个 React 组件。

Material Icons

Google has created over 2,000 official Material icons, each in five different "themes" (see below). For each SVG icon, we export the respective React component from the `@mui/icons-material` package. You can search the full list of these icons.

安装

请在您的项目目录中用以下方式安装依赖包：

```
// 使用 npm 安装
npm install @mui/icons-material

// 使用 yarn 安装
yarn add @mui/icons-material
```

这些组件使用 Material-UI 的 `SvgIcon` 组件来渲染每个图标的 SVG 路径，因此对 `@mui/material` 具有对等依赖性。

如果你的项目中还没有使用 MUI，那么你可以用以下方法添加它：

```
// 使用 npm 安装
npm install @mui/material

// 使用 yarn 安装
yarn add @mui/material
```

使用

有两种导入图标的方法：

- 方法 1:

  ```
  import AccessAlarmIcon from '@mui/icons-material/AccessAlarm';
  import ThreeDRotation from '@mui/icons-material/ThreeDRotation';
  ```

- 方法 2:

  ```
  import { AccessAlarm, ThreeDRotation } from '@mui/icons-material';
  ```

对于捆绑包的大小来说，最安全的是方案 1，但有些开发者更喜欢方案 2。 在使用第二个方法之前，请确保您遵循 最小化捆绑包大小指南。

每个 Material icon 都有一个“主题”：Filled（默认），Outlined, Rounded, Two-tone, 和 Sharp。 要导入非默认主题的图标组件，请在图标名称后附加主题名称。 例如， `@material-ui/icons/Delete` 图标可以：

- 导出为 Filled 主题（默认值）： `@material-ui/icons/Delete` ，
- 导出为 Outlined 主题： `@material-ui/icons/DeleteOutlined` ，
- 导出为 Rounded 主题： `@material-ui/icons/DeleteRounded` ，
- 导出为 Twotone 主题： `@material-ui/icons/DeleteTwoTone` ，
- 导出为 Sharp 主题： `@material-ui/icons/DeleteSharp` ，

> *Note: The Material Design guidelines name the icons using "snake_case" naming (for example `delete_forever` , `add_a_photo`), while `@material-ui/icons` exports the respective icons using "PascalCase" naming (for example `DeleteForever` , `AddAPhoto`). 并且此命名规则有三个特例：`3d_rotation` 导出为 `ThreeDRotation` ， `4k` 导出为 `FourK` ，以及 `360` 导出为 `ThreeSixty` 。 并且此命名规则有三个特例： `3d_rotation` 导出为 `ThreeDRotation` ， `4k` 导出为 `FourK` ，以及 `360` 导出为 `ThreeSixty` 。*

{{"demo": "SvgMaterialIcons.js"}}

测试

出于测试目的，每个从 `@material-ui/icons` 暴露的图标都有一个 `data-testid` 属性，这其中包含了图标的名称。 就像这样：

```
import DeleteIcon from '@mui/icons-material/Delete';
```

一旦挂载后，它就具有以下属性：

```
<svg data-testid="DeleteIcon"></svg>
```

SvgIcon（Svg 图标）

如果你需要使用自定义的 SVG 图标（而它在 Material Icons 中不存在），那么你可以使用 `SvgIcon` 封装。 此组件是原生 `<svg>` 元素的拓展版：

- 它具备一些内置的无障碍设计。
- SVG 元素应该在 24x24px 的视口中进行缩放，这样所渲染的图标就可以按原样使用，或者作为其他使用图标的 Material-UI 组件的子元素。 This can be customized with the `viewBox` attribute. To inherit the `viewBox` value from the original image, the `inheritViewBox` prop can be used.
- 默认情况下，此组件会继承当前的颜色。 当然，通过 `color` 这个属性，你可以让图标使用主题里的颜色。

```
function HomeIcon(props) {
  return (
    <SvgIcon {...props}>
      <path d="M10 20v-6h4v6h5v-8h3L12 3 2 12h3v8z" />
    </SvgIcon>
```

```
  );
}
```

Color 颜色

{{"demo": "SvgIconsColor.js"}}

Size 大小

{{"demo": "SvgIconsSize.js"}}

组件属性

即使图标以 `.svg` 的格式保存，你依然可以使用 `SvgIcon` 来包装它。 svgr has loaders to import SVG files and use them as React components. 譬如，使用 webpack： For example, with webpack:

```
// webpack.config.js
{
  test: /\.svg$/,
  use: ['@svgr/webpack'],
}

// ---
import StarIcon from './star.svg';

<SvgIcon component={StarIcon} viewBox="0 0 600 476.6" />
```

通过 “url-loader” 或 “file-loader” 加载也是可行的。 这是 Create React App 使用的方法。

```
// webpack.config.js
{
  test: /\.svg$/,
  use: ['@svgr/webpack', 'url-loader'],
}

// ---
import { ReactComponent as StarIcon } from './star.svg';

<SvgIcon component={StarIcon} viewBox="0 0 600 476.6" />
```

createSvgIcon

`createSvgIcon` 工具集是用于创建 Material icons 的。 它可以用来包装一个带有 SvgIcon 组件的 SVG 路径。

```
const HomeIcon = createSvgIcon(
  <path d="M10 20v-6h4v6h5v-8h3L12 3 2 12h3v8z" />,
  'Home',
);
```

{{"demo": "CreateSvgIcon.js"}}

Font Awesome

如果你发现在使用来自 `@fortawesome/react-fontawesome` 的 FontAwesomeIcon 时存在布局问题，你可以尝试将 Font Awesome SVG 数据直接传递给 SvgIcon。

如下是一个同时将 `Font Awesome<` 和 `Icon` 一起的示例：

{{"demo": "FontAwesomeSvgIconDemo.js"}}

FontAwesomeIcon 的 `fullWidth` 属性也可以用来应用近似正确的尺寸，但这样的效果并不完美。

其他图标库

MDI

materialdesignicons.com 提供了 2000 多个图标。 对于你想要使用的图标，可以复制它们所提供的 SVG `path`，并将其作为 `SvgIcon` 组件的子组件，或者也可以使用 `createSvgIcon()` 来应用它。

友情提示：mdi-material-ui 已经将每个 SVG 图标用 `SvgIcon` 组件包装起来，你可以高枕无忧了。

Icon (Font icons)

The `Icon` component will display an icon from any icon font that supports ligatures. As a prerequisite, you must include one, such as the Material icon font in your project. 若想要使用图标，您只需把图标名（字体连字）和 `Icon` 组件包装到一起，例如：

```
import Icon from '@material-ui/core/Icon';

<Icon>star</Icon>;
```

默认情况下，一个图标会继承使用当前的文本颜色。 您也可以选择使用以下任何一个主题颜色属性来设置图标的颜色： `primary`， `secondary`， `action`， `error` 以及 `disabled`。

Font Material 图标

`Icon` 默认情况下将为 Material Icons 字体（填充变体）设置正确的基类名称。 所以你需要加载字体即可，例如使用 Google Web Fonts：

```
<link
  rel="stylesheet"
  href="https://fonts.googleapis.com/icon?family=Material+Icons"
/>
```

{{"demo": "Icons.js"}}

自定义字体

对于其他字体，你可以使用 `baseClassName` 属性来自定义基线类名。 例如，你可以使用 Material Design 显示 two-tone 图标：

```
import Icon from '@mui/material/Icon';
```

```
<link
  rel="stylesheet"
  href="https://fonts.googleapis.com/css?family=Material+Icons+Two+Tone"
  // Import the two tones MD variant                              ^^^^^^^^
/>;
```

{{"demo": "TwoToneIcons.js"}}

全局类名

为了每个组件的使用都去修改 `baseClassName` 属性是很繁琐的。 你可以在全局范围内使用主题来改变默认属性。

```
const theme = createTheme({
  components: {
    MuiIcon: {
      defaultProps: {
        // Replace the `material-icons` default value.
        baseClassName: 'material-icons-two-tone',
      },
    },
  },
});
        baseClassName: 'material-icons-two-tone',
      },
    },
  },
});
        baseClassName: 'material-icons-two-tone',
      },
    },
  },
});
```

然后，你就可以直接使用 two-tone 图标了：

```
<Icon>add_circle</Icon>
```

Font Awesome

Font Awesome 可以和 `Icon` 组件一起使用，如下所示：

{{"demo": "FontAwesomeIcon.js"}}

需要注意的是，Font Awesome icons 的设计并不像 Material Design icons 那样（你可以对比之前的两个 demo）。 fa icons 经过裁剪，以利用所有可用空间。 你可以通过全局覆盖的方式来适配它：

```
const theme = createTheme({
  components: {
    MuiIcon: {
      styleOverrides: {
        root: {
```

```
            // Match 24px = 3 * 2 + 1.125 * 16
            boxSizing: 'content-box',
            padding: 3,
            fontSize: '1.125rem',
          },
        },
      },
    },
  },
});
```

{{"demo": "FontAwesomeIconSize.js"}}

Font vs SVG。 使用哪个更好呢?

这两种方法都能很好地工作，但是其中有一些微妙的差异，特别是在性能和渲染质量方面。 在条件允许的情况下，首选使用 SVG 的方式，因为它允许代码分割，能支持更多的图标，并且渲染得更快更好。

更多详情，请查阅 为什么 GitHub 将 font icons 迁移到 SVG icons。

无障碍设计

图标可以传达各种有意义的信息，所以确保它们在适当的地方可以被无障碍访问是很重要的。 下面是两个你需要考虑到的用例：

- **装饰性图标**仅用于增强视觉或强调品牌。 即使将它们从页面中移除，用户仍然可以理解并能够使用整个界面。
- **语义图标**指你用来传达意义的图标，而不仅仅是纯粹的装饰。 这包含了没有文字辅助说明的图标，这些图标一般被用作在交互式控件中 — 按钮、表单元素、切换按钮等。

装饰性图标

如果你的图标纯粹是装饰性的，那么已经大功告成了！ 而添加 `aria-hidden=true` 属性可以让你的图标变成正确的且可访问的（隐形的）。

语义图标

语义化的 SVG icons

你应该在 `titleAccess` 属性中增加一个有意义的值。 `role="img"` 属性和 `<title>` 元素将会被添加，以便你的图标可以正确适配无障碍设计。

对于可聚焦的交互式元素，例如当与图标按钮一起使用时，你可以使用 `aria-label` 属性：

```
import IconButton from '@material-ui/core/IconButton';
import SvgIcon from '@material-ui/core/SvgIcon';

// ...

<IconButton aria-label="delete">
  <SvgIcon>
    <path d="M20 12l-1.41-1.41L13 16.17V4h-2v12.17l-5.58-5.59L4 12l8 8 8-8z" />
  </SvgIcon>
</IconButton>;
```

```
<IconButton aria-label="delete">
  <SvgIcon>
    <path d="M20 12l-1.41-1.41L13 16.17V4h-2v12.17l-5.58-5.59L4 12l8 8 8-8z" />
  </SvgIcon>
</IconButton>;

<IconButton aria-label="delete">
  <SvgIcon>
    <path d="M20 12l-1.41-1.41L13 16.17V4h-2v12.17l-5.58-5.59L4 12l8 8 8-8z" />
  </SvgIcon>
</IconButton>;
```

语义化的 font icons

你需要提供一个只有辅助技术才能看到的文本替代方案：

```
import Box from '@mui/material/Box';
import Icon from '@mui/material/Icon';
import { visuallyHidden } from '@mui/utils';

// ...

import Box from '@material-ui/core/Box';
import Icon from '@material-ui/core/Icon';
import { visuallyHidden } from '@material-ui/utils';

// ...

<Icon>add_circle</Icon>
<Box component="span" sx={visuallyHidden}>Create a user</Box>

<Icon>add_circle</Icon>
<Box component="span" sx={visuallyHidden}>Create a user</Box>
```

参考

- https://www.tpgi.com/using-aria-enhance-svg-accessibility/