EDIT THIS PAGE

Icons

Guidance and suggestions for using icons with Material-UI.

Material-UI provides icons support in three ways:

1. Standardized Material Design icons exported as React components (SVG icons).
2. With the SvgIcon component, a React wrapper for custom SVG icons.
3. With the Icon component, a React wrapper for custom font icons.

Material Icons

Material Design has standardized over 1,100 official icons, each in five different "themes" (see below). For each SVG icon, we export the respective React component from the @material-ui/icons package. You can search the full list of these icons.

Installation

Install the package in your project directory with:

```
// with npm
npm install @material-ui/icons

// with yarn
yarn add @material-ui/icons
```

These components use the Material-UI SvgIcon component to render the SVG path for each icon, and so they have a peer-dependency on the next release of Material-UI.

If you are not already using Material-UI in your project, you can add it with:

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

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

Usage

Import icons using one of these two options:

- Option 1:

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

- Option 2:

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

The safest is Option 1 but Option 2 can yield the best developer experience. Make sure you follow the minimizing bundle size guide before using the second approach. The configuration of a Babel plugin is encouraged.

Each icon also has a "theme": Filled (default), Outlined, Rounded, Two tone and Sharp. If you want to import the icon component with a theme other than default, append the theme name to the icon name. For example `@material-ui/icons/Delete` icon with:

- Filled theme (default) is exported as `@material-ui/icons/Delete`,
- Outlined theme is exported as `@material-ui/icons/DeleteOutlined`,
- Rounded theme is exported as `@material-ui/icons/DeleteRounded`,
- Twotone theme is exported as `@material-ui/icons/DeleteTwoTone`,
- Sharp theme is exported as `@material-ui/icons/DeleteSharp`.

> Note: The Material Design specification names 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`). There are three exceptions to this naming rule: `3d_rotation` exported as `ThreeDRotation`, `4k` exported as `FourK`, and `360` exported as `ThreeSixty`.

Filled
Outlined
Rounded
Two Tone
Sharp
Edge-cases

SvgIcon

If you need a custom SVG icon (not available in the Material Icons default set) you can use the `SvgIcon` wrapper. This component extends the native `<svg>` element:

- It comes with built-in accessibility.
- SVG elements should be scaled for a 24x24px viewport, so the resulting icon can be used as is, or included as a child for other Material-UI components that use icons. (This can be customized with the `viewBox` attribute).
- By default, the component inherits the current color. Optionally, you can apply one of the theme colors using the `color` prop.

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

Color

Size

Component prop

You can use the `SvgIcon` wrapper even if your icons are saved in the `.svg` format. svgr has loaders to import SVG files and use them as React components. 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" />
```

It's also possible to use it with "url-loader" or "file-loader". It's the approach used by Create React App.

```
// 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" />
```

Libraries

Material Design (recommended)

Material Design has standardized over 1,100 official icons.

MDI

materialdesignicons.com provides over 2,000 icons. For the wanted icon, copy the SVG `path` they provide, and use it as the child of the `SvgIcon` component.

Note: mdi-material-ui has already wrapped each of these SVG icons with the `SvgIcon` component, so you don't have to do it yourself.

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, for instance, via Google Web Fonts:

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

`Icon` will set the correct class name for the Material icon font. For other fonts, you must supply the class name using the Icon component's `className` property.

To use an icon simply wrap the icon name (font ligature) with the `Icon` component, for example:

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

<Icon>star</Icon>
```

By default, an Icon will inherit the current text color. Optionally, you can set the icon color using one of the theme color properties: `primary`, `secondary`, `action`, `error` & `disabled`.

Font Material icons

Font Awesome

Font Awesome can be used with the `Icon` component as follow:

Font vs SVG. Which approach to use?

Both approaches work fine, however, there are some subtle differences, especially in terms of performance and rendering quality. Whenever possible SVG is preferred as it allows code splitting, supports more icons, renders faster and better.

For more details, you can check out why GitHub migrated from font icons to SVG icons.

Accessibility

Icons can convey all sorts of meaningful information, so it's important that they reach the largest amount of people possible. There are two use cases you'll want to consider:

- **Decorative Icons** are only being used for visual or branding reinforcement. If they were removed from the page, users would still understand and be able to use your page.
- **Semantic Icons** are ones that you're using to convey meaning, rather than just pure decoration. This includes icons without text next to them used as interactive controls — buttons, form elements, toggles, etc.

Decorative SVG Icons

If your icons are purely decorative, you're already done! The `aria-hidden=true` attribute is added so that your icons are properly accessible (invisible).

Semantic SVG Icons

If your icon has semantic meaning, all you need to do is throw in a `titleAccess="meaning"` property. The `role="img"` attribute and the `<title>` element are added so that your icons are properly accessible.

In the case of focusable interactive elements, like when used with an icon button, you can use the `aria-label` property:

```
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>
```

Decorative Font Icons

If your icons are purely decorative, you're already done! The `aria-hidden=true` attribute is added so that your icons are properly accessible (invisible).

Semantic Font Icons

If your icons have semantic meaning, you need to provide a text alternative only visible to assistive technologies.

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

// ...

<Icon>add_circle</Icon>
<Typography variant="srOnly">Create a user</Typography>
```

Reference

- https://developer.paciellogroup.com/blog/2013/12/using-aria-enhance-svg-accessibility/

API

- `<Icon />`
- `<SvgIcon />`

‹ Divider

Material Icons ›