EDIT THIS PAGE

@material-ui/styles

You can use Material-UI's styling solution in your app, whether or not you are using Material-UI components.

Material-UI aims to provide a strong foundation for building dynamic UIs. For the sake of simplicity, **we expose the styling solution used in Material-UI components** as the `@material-ui/styles` package. You can use it, but you don't have to, since Material-UI is also interoperable with all the other major styling solutions.

Why use Material-UI's styling solution?

In previous versions, Material-UI has used LESS, then a custom inline-style solution to write the component styles, but these approaches have proven to be limited. A *CSS-in-JS* solution overcomes many of those limitations, and **unlocks many great features** (theme nesting, dynamic styles, self-support, etc.).

Material-UI's styling solution is inspired by many other styling libraries such as styled-components and emotion.

- 💅 You can expect the same advantages as styled-components.
- 🚀 It's blazing fast.
- 🔌 It's extensible via a plugin API.
- ⚡ It uses JSS at its core – a high performance JavaScript to CSS compiler which works at runtime and server-side.
- 🛍 Less than 15 KB gzipped; and no bundle size increase if used alongside Material-UI.

Installation

> `@material-ui/styles` is re-exported as `@material-ui/core/styles` - you only need to install it if you wish to use it independently from Material-UI.

To install and save in your `package.json` dependencies, run:

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

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

Getting started

There are 3 possible APIs you can use to generate and apply styles, however they all share the same underlying logic.

Hook API

```jsx
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Button from '@material-ui/core/Button';

const useStyles = makeStyles({
  root: {
    background: 'linear-gradient(45deg, #FE6B8B 30%, #FF8E53 90%)',
    border: 0,
    borderRadius: 3,
    boxShadow: '0 3px 5px 2px rgba(255, 105, 135, .3)',
    color: 'white',
    height: 48,
    padding: '0 30px',
  },
});

export default function Hook() {
  const classes = useStyles();
  return <Button className={classes.root}>Hook</Button>;
}
```

STYLED WITH HOOK API

Styled components API

Note: this only applies to the calling syntax – style definitions still use a JSS object. You can also change this behavior, with some limitations.

```jsx
import React from 'react';
import { styled } from '@material-ui/core/styles';
import Button from '@material-ui/core/Button';

const MyButton = styled(Button)({
  background: 'linear-gradient(45deg, #FE6B8B 30%, #FF8E53 90%)',
  border: 0,
  borderRadius: 3,
  boxShadow: '0 3px 5px 2px rgba(255, 105, 135, .3)',
  color: 'white',
  height: 48,
  padding: '0 30px',
});

export default function StyledComponents() {
  return <MyButton>Styled Components</MyButton>;
}
```

STYLED WITH STYLED-COMPONENTS API

Higher-order component API

```jsx
import React from 'react';
import PropTypes from 'prop-types';
import { withStyles } from '@material-ui/core/styles';
import Button from '@material-ui/core/Button';

const styles = {
  root: {
    background: 'linear-gradient(45deg, #FE6B8B 30%, #FF8E53 90%)',
    border: 0,
    borderRadius: 3,
    boxShadow: '0 3px 5px 2px rgba(255, 105, 135, .3)',
    color: 'white',
    height: 48,
    padding: '0 30px',
  },
};

function HigherOrderComponent(props) {
  const { classes } = props;
  return <Button className={classes.root}>Higher-order component</Button>;
}

HigherOrderComponent.propTypes = {
  classes: PropTypes.object.isRequired,
};

export default withStyles(styles)(HigherOrderComponent);
```

STYLED WITH HOC API

Nesting selectors

You can nest selectors to target elements inside the current class or component. The following example uses the Hook API, but it works the same way with the other APIs.

```
const useStyles = makeStyles({
  root: {
    color: 'red',
    '& p': {
      color: 'green',
      '& span': {
        color: 'blue'
      }
    }
  },
});
```

This is red since it is inside the root.
This is green since it is inside the paragraph and this is blue since it is inside the span

---

Adapting based on props

You can pass a function to `makeStyles` ("interpolation") in order to adapt the generated value based on the component's props. The function can be provided at the style rule level, or at the CSS property level:

```
const useStyles = makeStyles({
  // style rule
  foo: props => ({
    backgroundColor: props.backgroundColor,
  }),
  bar: {
    // CSS property
    color: props => props.color,
  },
});

function MyComponent() {
  // Simulated props for the purpose of the example
  const props = { backgroundColor: 'black', color: 'white' };
  // Pass the props as the first argument of useStyles()
  const classes = useStyles(props);

  return <div className={`${classes.foo} ${classes.bar}`} />
}
```

This button component has a color property that changes its color:

Adapting the hook API

Adapting the styled components API

Adapting the higher-order component API

Stress test

In the following stress test, you can update the *theme color* and the *background-color property* live:

```
const useStyles = makeStyles(theme => ({
  root: props => ({
    backgroundColor: props.backgroundColor,
    color: theme.color,
  }),
}));
```

rendered 1 times
color: #ffffff
backgroundColor: #2196f3

@material-ui/core/styles vs @material-ui/styles

Material-UI's styles are powered by the @material-ui/styles package, (built with JSS). This solution is isolated. It doesn't have a default theme, and can be used to style React applications that are not using Material-UI components.

To reduce the number of packages to install when using Material-UI, and to simplify the imports, `@material-ui/styles` modules are re-exported from `@material-ui/core/styles`.

To remove the need to systematically supply a theme, the default Material-UI theme is applied to the re-exported `makeStyles`, `styled`, `withTheme`, `useTheme`, and `withStyles` modules.

For instance:

```
// Re-export with a default theme
import { makeStyles } from '@material-ui/core/styles';

// Original module with no default theme
import { makeStyles } from '@material-ui/styles';
```

‹ Zoom

Advanced ›