EDIT THIS PAGE

Style Library Interoperability

While you can use the JSS based styling solution provided by Material-UI to style your application, you can also use the one you already know and love (from plain CSS to styled-components).

This guide aims to document the most popular alternatives, but you should find that the principles applied here can be adapted to other libraries. There are examples for the following styling solutions:

- Plain CSS
- Global CSS
- Styled Components
- CSS Modules
- Emotion
- React JSS

Plain CSS

Nothing fancy, just plain CSS.

DEFAULT CUSTOMIZED

Edit on **CodeSandbox**

PlainCssButton.css

```
.button {
  background-color: #6772e5;
  color: #fff;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
}
.button:hover {
  background-color: #5469d4;
}
```

PlainCssButton.js

```
import React from 'react';
import Button from '@material-ui/core/Button';
import './PlainCssButton.css';

export default function PlainCssButton() {
  return (
    <div>
      <Button>Default</Button>
      <Button className="button">Customized</Button>
    </div>
  );
}
```

Controlling priority ⚠

Note: JSS injects its styles at the bottom of the `<head>`. If you don't want to mark style attributes with **!important**, you need to change the CSS injection order, as in the demo:

```
import { StylesProvider } from '@material-ui/core/styles';

<StylesProvider injectFirst>
  {/* Your component tree.
      Now, you can override Material-UI's styles. */}
</StylesProvider>
```

Deeper elements

If you attempt to style a Drawer with variant permanent, you will likely need to affect the Drawer's child paper element. However, the paper is not the root element of Drawer and therefore styled-components customization as above will not work. You need to use the `classes` API of Material-UI.

The following example overrides the `label` style of `Button` in addition to the custom styles on the button itself.

PlainCssButtonDeep.css

```
.button {
  background-color: #6772e5;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
}
.button:hover {
  background-color: #5469d4;
}
.button-label {
  color: #fff;
}
```

PlainCssButtonDeep.js

```
import React from 'react';
import Button from '@material-ui/core/Button';
import './PlainCssButtonDeep.css';

export default function PlainCssButtonDeep() {
  return (
    <div>
      <Button>Default</Button>
      <Button classes={{ root: 'button', label: 'button-label' }}>
        Customized
      </Button>
    </div>
  );
}
```

Global CSS

Explicitly providing the class names to the component is too much effort? You can target the class names generated by Material-UI.

Edit on **CodeSandbox**

GlobalCssButton.css

```
.MuiButton-root {
  background-color: #6772e5;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
}
.MuiButton-root:hover {
  background-color: #5469d4;
}
.MuiButton-label {
  color: #fff;
}
```

GlobalCssButton.js

```
import React from 'react';
import Button from '@material-ui/core/Button';
import './GlobalCssButton.css';

export default function GlobalCssButton() {
  return <Button>Customized</Button>;
}
```

Controlling priority ⚠

Note: JSS injects its styles at the bottom of the `<head>`. If you don't want to mark style attributes with **!important**, you need to change the CSS injection order, as in the demo:

```
import { StylesProvider } from '@material-ui/core/styles';

<StylesProvider injectFirst>
  {/* Your component tree.
      Now, you can override Material-UI's styles. */}
</StylesProvider>
```

Styled Components

Star 32k downloads 8.5M/month

The `styled()` method works perfectly on all of the components.

Edit on **CodeSandbox**

```
import React from 'react';
import styled from 'styled-components';
import Button from '@material-ui/core/Button';

const StyledButton = styled(Button)`
  background-color: #6772e5;
  color: #fff;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
  &:hover {
    background-color: #5469d4;
  }
`;

export default function StyledComponents() {
  return (
    <div>
      <Button>Default</Button>
      <StyledButton>Customized</StyledButton>
    </div>
  );
}
```

Controlling priority ⚠

Note: Both styled-components and JSS inject their styles at the bottom of the `<head>`. The best approach to ensuring styled-components styles are loaded last is to change the CSS injection order, as in the demo:

```
import { StylesProvider } from '@material-ui/core/styles';

<StylesProvider injectFirst>
  {/* Your component tree.
      Now, you can override Material-UI's styles. */}
</StylesProvider>
```

Another approach is to use the `&&` characters in styled-components to bump up specificity by repeating the class name. Avoid the usage of `!important`.

Deeper elements

If you attempt to style a Drawer with variant permanent, you will likely need to affect the Drawer's child paper element. However, the paper is not the root element of Drawer and therefore styled-components customization as above will not work. You need to use the `classes` API of Material-UI.

The following example overrides the `label` style of `Button` in addition to the custom styles on the button itself. It also works around this styled-components issue by "consuming" properties that should not be passed on to the underlying component.

```
import React from 'react';
import styled from 'styled-components';
import Button from '@material-ui/core/Button';

const StyledButton = styled(Button)`
  background-color: #6772e5;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
  &:hover {
    background-color: #5469d4;
  }
  & .MuiButton-label {
    color: #fff;
  }
`;

export default function StyledComponentsDeep() {
  return (
    <div>
      <Button>Default</Button>
      <StyledButton>Customized</StyledButton>
    </div>
  );
}
```

The above demo relies on the default `classes` values but you can provide your own class name: `.label`.

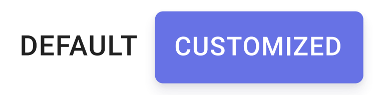
```
import React from 'react';
import styled from 'styled-components';
import Button from '@material-ui/core/Button';

const StyledButton = styled(({ color, ...other }) => (
  <Button classes={{ label: 'label' }} {...other} />
))`
  background-color: #6772e5;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
  &:hover {
    background-color: #5469d4;
  }
  & .label {
    color: #fff;
  }
`;

export default function StyledComponentsDeep() {
  return (
    <div>
      <Button>Default</Button>
      <StyledButton>Customized</StyledButton>
    </div>
  );
}
```

Theme

Material-UI has a rich theme structure that you can leverage for the color manipulations, the transitions, the media queries, and more.

We encourage to share the same theme object between Material-UI and your styles.

```
const StyledButton = styled(Button)`
  ${({ theme }) => `
  background-color: ${theme.palette.primary.main};
  color: #fff;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 4px 10px;
  font-size: 13px;
  &:hover {
    background-color: ${darken(theme.palette.primary.main, 0.2)};
  }
  ${theme.breakpoints.up('sm')} {
    font-size: 14px;
    padding: 7px 14px;
  }
  `}
`;
```



Portals

The Portal provides a first-class way to render children into a DOM node that exists outside the DOM hierarchy of the parent component. Because of the way styled-components scopes its CSS, you may run into issues where styling is not applied.

For example, if you attempt to style the Menu of a Select component using the property `MenuProps` , you will need to pass along the `className` property to the element being rendered outside of it's DOM hierarchy. The following example shows a workaround:

```
import React from 'react';
import styled from 'styled-components';
import Menu from '@material-ui/core/Menu';
import MenuItem from '@material-ui/core/MenuItem';

const StyledMenu = styled(({ className, ...props }) => (
  <Menu {...props} classes={{ paper: className }} />
))`
  box-shadow: none;
  border: 1px solid #d3d4d5;

  li {
    padding-top: 8px;
    padding-bottom: 8px;
  }
`;
```

OPEN MENU

CSS Modules

Star 14k

It's hard to know the market share of this styling solution as it's dependent on the bundling solution people are using.

DEFAULT CUSTOMIZED

Edit on CodeSandbox

CssModulesButton.css

```css
.button {
  background-color: #6772e5;
  color: #fff;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
}
.button:hover {
  background-color: #5469d4;
}
```

CssModulesButton.js

```jsx
import React from 'react';
// webpack, parcel or else will inject the CSS into the page
import styles from './CssModulesButton.css';
import Button from '@material-ui/core/Button';

export default function CssModulesButton() {
  return (
    <div>
      <Button>Default</Button>
      <Button className={styles.button}>Customized</Button>
    </div>
  );
}
```

Controlling priority ⚠

Note: JSS injects its styles at the bottom of the `<head>`. If you don't want to mark style attributes with **!important**, you need to change the CSS injection order, as in the demo:

```jsx
import { StylesProvider } from '@material-ui/core/styles';

<StylesProvider injectFirst>
  {/* Your component tree.
      Now, you can override Material-UI's styles. */}
</StylesProvider>
```

Deeper elements

If you attempt to style a Drawer with variant permanent, you will likely need to affect the Drawer's child paper element. However, the paper is not the root element of Drawer and therefore styled-components customization as above will not work. You need to use the `classes` API of Material-UI.

The following example overrides the `label` style of `Button` in addition to the custom styles on the button itself.

CssModulesButtonDeep.css

```css
.root {
  background-color: #6772e5;
  box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
  padding: 7px 14px;
}
.root:hover {
  background-color: #5469d4;
}
.label {
  color: #fff;
}
```

CssModulesButtonDeep.js

```jsx
import React from 'react';
// webpack, parcel or else will inject the CSS into the page
import styles from './CssModulesButtonDeep.css';
import Button from '@material-ui/core/Button';

export default function CssModulesButtonDeep() {
  return (
    <div>
      <Button>Default</Button>
      <Button classes={styles}>Customized</Button>
    </div>
  );
}
```

Emotion

Star 12k downloads 2.8M/month

The `css` prop

Emotion's **css()** method works seamlessly with Material-UI.

Edit on **CodeSandbox**

```
/** @jsx jsx */
import { jsx, css } from '@emotion/core';
import Button from '@material-ui/core/Button';

export default function EmotionCSS() {
  return (
    <div>
      <Button>Default</Button>
      <Button
        css={css`
          background-color: #6772e5;
          color: #fff;
          box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08)
          padding: 7px 14px;
          &:hover {
            background-color: #5469d4;
          }
        `}
      >
        Customized
      </Button>
    </div>
  );
}
```

Controlling priority ⚠

Note: JSS injects its styles at the bottom of the `<head>`. If you don't want to mark style attributes with **!important**, you need to change the CSS injection order, as in the demo:

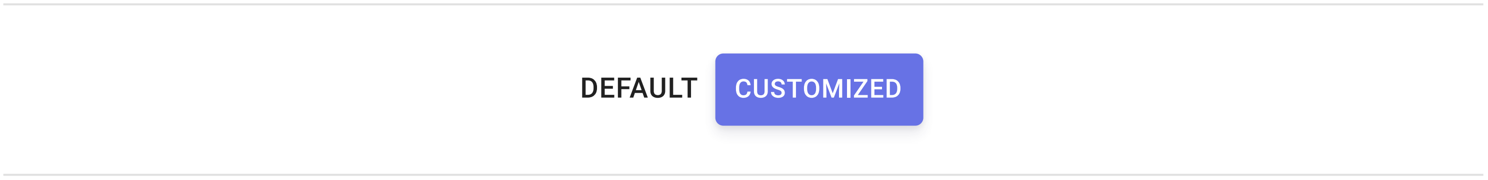
```
import { StylesProvider } from '@material-ui/core/styles';

<StylesProvider injectFirst>
  {/* Your component tree.
      Now, you can override Material-UI's styles. */}
</StylesProvider>
```

Theme

Material-UI has a rich theme structure that you can leverage for the color manipulations, the transitions, the media queries, and more.

We encourage to share the same theme object between Material-UI and your styles.

```
<Button
  css={theme => css`
    background-color: ${theme.palette.primary.main};
    color: #fff;
    box-shadow: 0 4px 6px rgba(50, 50, 93, 0.11), 0 1px 3px rgba(0, 0, 0, 0.08);
    padding: 4px 10px;
    font-size: 13px;
    &:hover {
      background-color: ${darken(theme.palette.primary.main, 0.2)};
    }
    ${theme.breakpoints.up('sm')} {
      font-size: 14px;
      padding: 7px 14px;
    }
  `}
>
  Customized
</Button>
```



The `styled()` API

It works exactly like styled components. You can use the same guide.

TypeScript

Minimizing Bundle Size