



Gaurav Singhal

Require Material-UI by Webpack

Gaurav Singhal

Nov 5, 2020 • 5 Min read • 417 Views

Nov 5, 2020 • 5 Min read • 417 Views

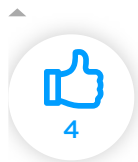
Web Development

Front End Web Dev...

Client-side Framew...

React

Introduction



- [Introduction](#)
- [Install and Get Started with Material-UI](#)
- [Import Material-UI Components Using Webpack](#)
- [Conclusion](#)
- [Top ^](#)

Introduction

React does not have its own UI elements/components, but you can use any third-party UI framework. These frameworks provides a bunch of components for the UI requirements.

The Material-UI is one of the popular UI frameworks designed for React, and contains various ready-to-integrate components. Still, if you use a webpack, then it should be configured while using webpack configuration. This guide will demonstrate how to use Material-UI and its various components configured with webpack.

Install and Get Started with Material-UI

The initial step to getting started with Material-UI is to install the package using the below command.

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

shell

After completing the installation, the next step is to import the useful component from Material-UI and use it in any React component. This example uses the `Breadcrumbs` functionality.

```
1      import Typography from '@material-ui/core/Typography';
2      import Breadcrumbs from '@material-ui/core/Breadcrumbs';
3      import Link from '@material-ui/core/Link';
4
5      function App() {
6        return (
7          <div className="App">
8            <Breadcrumbs aria-label="breadcrumb">
9              <Link color="inherit" href="/">
10                Parent Page
11              </Link>
12              <Link color="inherit" href="/">
13                Child Page
14              </Link>
```

jsx

```
15         <Typography color="textPrimary">Current Page</Typography>
16     </Breadcrumbs>
17 </div>
18 );
19 }
20
21 export default App;
```

The above file imports the `BreadCrumbs` component from `material-ui` and uses it in the render function, and requires child components and props. The above approach is the simplest way to get started with Material-UI in no time. However, if you are using a custom webpack configuration, you need to follow some additional configuration.

Import Material-UI Components Using Webpack

The webpack configuration depends on resolving modules based on the path from which it's referenced. There are multiple ways to require Material-UI or other such libraries, and one of them is the *module resolving* approach along with `eslint`.

Using the `eslint`, specify the module path or require it into the webpack config.

```
1  {  
2    "rules": {  
3      "no-restricted-imports": [  
4        "error",  
5        {  
6          "patterns": ["@material-ui/*//*/"]  
7        }  
8      ]  
9    }  
10 }
```

After configuring the above settings, any false import statement gets highlighted as a false import statement.

If you want to try different plugins, other plugins are available to include the Material-UI and its respective packages:

- [babel-plugin-import](#)
- [babel-plugin-transform-imports](#)

Before using any of the above libraries, you need to create the file `.babelrc.js` in your React app's root directory.

.babelrc.js

```
1  const plugin = [  
2    [  
3      'babel-plugin-transform-imports',
```

```

4      {
5        '@material-ui/core': {
6          'transform': '@material-ui/core/${member}',
7          'preventFullImport': true
8        }
9      }
10    ]
11  ];
12
13  module.exports = {plugin};

```

The above example imports the `@material-ui/core` library and provides transformation to import the package members such as `Button`, `Breadcrumbs`, etc.

```

1      import Breadcrumbs from '@material-ui/core/Breadcrumbs';

```

jsx

The webpack transformed import helps you require the only valuable part of the module rather than importing the whole package and resolving the specific module.

```

1      import { Breadcrumbs } from '@material-ui/core';

```

jsx

The above statement will import the complete library packages and will resolve `Breadcrumbs` at the time of compilation, so the module

resolver comes in handy when you want to decrease the overall bundling efforts.

You can also use `webpack.resolve` to resolve the given path's exact match.

```
1      module.exports = {
2        //...
3        resolve: {
4          alias: {
5            abcd$: path.resolve(__dirname, 'path/to/file_name.js')
6          }
7        }
8      };
```

js

The above config will resolve the path to the normal path from where it gets imported.

```
1      import file1 from 'folder1'; // Path resolved
2      import file2 from 'folder2/file.js'; // Not matched and resolved
```

jsx

The `file1` is an exported member of the `folder1` location. Hence, it will get resolved based on the specified path resolver, and `file2` will not get resolved because it's trying to resolve against the specific path.

Conclusion

Material-UI is a great UI framework for React apps, and using it with the webpack allows you to minimize bundling size by implementing a custom import statement resolver.

You can use module resolver libraries such as [babel-plugin-transform-imports](#) to define the concrete standard to import any external libraries and resolve them efficiently. If you have any queries, feel free to ask at [Code Alphabet](#).

