# gatsby-plugin-sass

Provides drop-in support for Sass/SCSS stylesheets

## Install

`npm install sass gatsby-plugin-sass`

## How to use

1. Include the plugin in your `gatsby-config.js` file.

```
plugins: [`gatsby-plugin-sass`]
```

2. Write your stylesheets in Sass/SCSS and require or import them as normal.

```
html {
  background-color: rebeccapurple;
  p {
    color: white;
  }
}
```

```
import "./src/index.scss"
```

## Other options

If you need to pass options to Sass use the plugins options, see `node-sass` / `dart-sass` docs for all available options.

```
plugins: [
  {
    resolve: `gatsby-plugin-sass`,
    options: {
      sassOptions: {
        includePaths: ["absolute/path/a", "absolute/path/b"],
        ...
      }
    },
  },
]
```

If you need to override the default options passed into `css-loader`. **Note:** Gatsby is using `css-loader@^5.0.0`.

```
plugins: [
  {
    resolve: `gatsby-plugin-sass`,
    options: {
      cssLoaderOptions: {
        camelCase: false,
      },
    },
  },
]
```

### Alternative Sass Implementations

By default, the Dart implementation of Sass ( `sass` ) is used. To use the implementation written in Node ( `node-sass` ), you can install `node-sass` instead of `sass` and pass it into the options as the implementation:

```
npm install node-sass
```

```
plugins: [
  {
    resolve: `gatsby-plugin-sass`,
    options: {
      implementation: require("node-sass"),
    },
  },
]
```

### Sass Precision

`sass` intentionally doesn't have support for setting a custom `precision` . `node-sass` defaults to 5 digits of precision. If you want some other level of precision (e.g. if you use Bootstrap), you may configure it as follows:

#### Bootstrap 4

See Bootstrap's documentation on theming for reference.

```
plugins: [
  {
    resolve: `gatsby-plugin-sass`,
    options: {
      implementation: require("node-sass"),
      postCssPlugins: [somePostCssPlugin()],
      sassOptions: {
        precision: 6,
      },
    },
  },
]
```

#### Bootstrap 3 (with `bootstrap-sass` )

See `bootstrap-sass` for reference.

```
plugins: [
  {
    resolve: `gatsby-plugin-sass`,
    options: {
      implementation: require("node-sass"),
      postCssPlugins: [somePostCssPlugin()],
      sassOptions: {
        precision: 8,
      },
    },
  },
]
```

### With CSS Modules

Using CSS Modules requires no additional configuration. Simply prepend `.module` to the extension. For example: `app.scss` -> `app.module.scss`. Any file with the `module` extension will use CSS Modules. CSS modules are imported as ES Modules to support treeshaking. You'll need to import styles as: `import { yourClassName, anotherClassName } from './app.module.scss'`

## Sass & CSS Modules file Regexes

To override the file regex for Sass or CSS modules,

```
plugins: [
  {
    resolve: `gatsby-plugin-sass`,
    options: {
      // Override the file regex for Sass
      sassRuleTest: /\.global\.s(a|c)ss$/,
      // Override the file regex for CSS modules
      sassRuleModulesTest: /\.mod\.s(a|c)ss$/,
    },
  },
]
```

### PostCSS plugins

PostCSS is also included to handle some default optimizations like autoprefixing and common cross-browser flexbox bugs. Normally you don't need to think about it, but if you'd prefer to add additional postprocessing to your Sass output you can specify plugins in the plugin options.

## Relative paths & `url()`

This plugin resolves `url()` paths relative to the entry SCSS/Sass file not – as might be expected – the location relative to the declaration. Under the hood, it makes use of `sass-loader` and this is documented in the readme.

Using `resolve-url-loader` provides a workaround, if you want to use relative url just install the plugin and then add it to your Sass plugin options configuration.

First:

```
npm install resolve-url-loader --save-dev
```

And then:

```
plugins: [
  {
    resolve: "gatsby-plugin-sass",
    options: {
      useResolveUrlLoader: true,
    },
  },
]
```

You can also configure `resolve-url-plugin` providing some options (see plugin documentation for all options):

```
plugins: [
  {
    resolve: "gatsby-plugin-sass",
    options: {
      useResolveUrlLoader: {
        options: {
          debug: true,
        },
      },
    },
  },
]
```

**Please note:** Adding `resolve-url-loader` will use `sourceMap: true` on `sass-loader` (as it is required for the plugin to work), you can then activate/deactivate source-map for Sass files in the plugin:

```
plugins: [
  {
    resolve: "gatsby-plugin-sass",
    options: {
      useResolveUrlLoader: {
        options: {
          sourceMap: true, //default is false
        },
      },
    },
  },
]
```

## Breaking changes history

### v3.0.0

- `sass-loader` is updated to v10 which adds support for `node-sass@^5.0.0` but also switches the default `implementation` to `sass`. webpack also recommends using `sass` so this is reflected in the documentation here, too. In the deprecation notice of node-sass it is noted that switching from `node-sass` to `sass` is straightforward as both packages use the same JavaScript API.
- All options for both `node-sass` & `sass` are moved into the `sassOptions` object
- You're now able to override the `importLoaders` option. If you have this in your options but don't intend to override it, you'll need to remove it

### v2.0.0

- `node-sass` is moved to a peer dependency. Installing the package alongside `gatsby-plugin-sass` is now required. Use `npm install node-sass`

- support Gatsby v2 only