PostCSS transforms extended syntaxes and features into modern, browser-friendly CSS. This guide will show you how to get started with Gatsby and PostCSS.

Installing and configuring PostCSS

This guide assumes that you have a Gatsby project set up. If you need to set up a project, head to the quick start guide, then come back.

1. Install the Gatsby plugin gatsby-plugin-postcss.

```
npm install postcss gatsby-plugin-postcss
```

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

```
plugins: [`gatsby-plugin-postcss`],
```

> ***Note**: If you need to pass options to PostCSS use the plugins options; see postcss-loader for all available options.*

3. Write your stylesheets using PostCSS (`.css` files) and require or import them as normal.

```
@custom-media --med (width <= 50rem);

@media (--med) {
  a {
    &:hover {
      color: color-mod(black alpha(54%));
    }
  }
}
```

```
import "./styles.css"
```

With CSS modules

To use CSS modules, prepend `.module` to the extension. For example: `App.css -> App.module.css` . Any file with the module extension will use CSS modules.

PostCSS plugins

If you would prefer to add additional postprocessing to your PostCSS output you can specify plugins in the plugin options:

```
plugins: [
  {
    resolve: `gatsby-plugin-postcss`,
    options: {
      postCssPlugins: [require(`postcss-preset-env`)({ stage: 0 })],
    },
```

```
  },
],
```

Alternatively, you can use `postcss.config.js` to specify your particular PostCSS configuration:

```
const postcssPresetEnv = require(`postcss-preset-env`)

module.exports = () => ({
  plugins: [
    postcssPresetEnv({
      stage: 0,
    }),
  ],
})
```

Other resources

- Introduction to PostCSS