If a plugin is only relevant to your specific use-case, or if you're developing a plugin and want a simpler workflow, a locally defined plugin is a convenient way to create and manage your plugin code.

Project structure for a local plugin

Place the code in the `plugins` folder in the root of your project like this:

```
/my-gatsby-site
└── gatsby-config.js
└── /src
└── /plugins
    └── /my-own-plugin
        └── package.json
```

The plugin also needs to be added to your `gatsby-config.js`, because there is no auto-detection of plugins. It can be added alongside any other 3rd party Gatsby plugins already included in your config.

For the plugin to be discovered when you run `gatsby develop`, the plugin's root folder name needs to match the name used in the `gatsby-config.js` (*not* the *name* it goes by in your `package.json` file). For example, in the above structure, the correct way to load the plugin is:

```
module.exports = {
  plugins: [
    `gatsby-third-party-plugin`,
    `my-own-plugin`, // highlight-line
  ],
}
```

Then the plugin can begin to hook into Gatsby through Node and SSR APIs.

Developing a local plugin that is outside your project

Your plugin doesn't have to be in your project in order to be tested or worked on. If you'd like to decouple your plugin from your site you can follow one of the methods described below. This is a useful thing to do if you want to publish the plugin as its own package, or test/develop a forked version of a community authored plugin.

To get started developing a plugin outside of your site's root folder, you can quickly generate one using `gatsby new` with the starter for plugins:

```
gatsby new gatsby-plugin-foo https://github.com/gatsbyjs/gatsby-starter-plugin
```

Using `require.resolve` and a filepath

Including a `plugins` folder is not the only way to reference a local plugin. Alternatively, you can include a plugin in your `gatsby-config.js` file by directly referencing its path (relative to the `gatsby-config.js` file) with `require`.

```
module.exports = {
  plugins: [
```

```
    `gatsby-plugin-react-helmet`,
    // highlight-start
    {
      // including a plugin from outside the plugins folder needs the path to it
      resolve: require.resolve(`../path/to/gatsby-local-plugin`),
    },
    // highlight-end
  ],
}
```

Using `npm link` or `yarn link`

You can use `npm link` or `yarn link` to reference a package from another location on your machine.

By running `npm link ../path/to/my-plugin` in the root of your Gatsby site, your computer will create a symlink to your package.

This is a similar process to setting up yarn workspaces for development with Gatsby themes (which is the recommended approach for developing themes). You can read how to set up a site in this manner in the Building a Theme guide.

Note: See an example of using a local plugin from the plugins folder, with `require.resolve`, and `npm link` in this example repository.

Compilation and processing with Babel

Except for the `gatsby-browser.js` file, which is processed as a part of the webpack bundling step, the code from all `gatsby-*` files is not processed by babel. If you want to use JavaScript syntax which isn't supported by your version of Node.js, you can place the files in your `src` subfolder and build them to the plugin folder root.