Creating a Generic Plugin

This section aims to explain the structure of a Gatsby plugin and the files you need to create one.

The idea of a generic plugin is to lay more emphasis on the makeup of a plugin rather than the specific labels (source, transformer, local) that are selected based on functionality. As seen in the what is a plugin doc, a plugin is a piece of software that acts as an add-on and gives a Gatsby site additional functionality.

Plugins contain a file, usually in the project root, called `package.json` - this file holds various metadata relevant to the project. The `package.json` file is also used to provide information to npm that identifies the project and allows npm to handle the project's dependencies.

Initializing your plugin project

To initialize a `package.json` for your project, run the following command:

```
npm init
```

Once you've run the command you'll see a series of options listed in the command-line interface (CLI). Those you select are stored in your `package.json` which contains some of the files Gatsby looks for in a Plugin

What happens in a generic plugin?

In a generic plugin the `gatsby-node.js` file enables the use of gatsby node APIs. These APIs, such as `createPage`, `createResolvers`, and `sourceNodes`, manipulate the Node(s) in a Gatsby site. A Node is the smallest unit of data in Gatsby. You can create a Node using the createNode action.

In `gatsby-node.js` you can carry out functions with these APIs, such as:

- Loading API keys
- Sending calls to APIs
- Creating Gatsby-nodes using the API response
- Creating individual pages from nodes

  A good use case of the above would be a plugin that gets data from an API.

An example of a generic plugin

sourceNodes is a lifecycle API that a plugin can use to create Nodes. An example of how to implement a function using `sourceNodes` is shown below:

```
exports.sourceNodes = ({ actions, createNodeId, createContentDigest }) => {
  const nodeData = {
    title: "Test Node",
    description: "Testing the node ",
  }
  const newNode = {
    ...nodeData,
    id: createNodeId("TestNode-testid"),
    internal: {
      type: "TestNode",
      contentDigest: createContentDigest(nodeData),
    },
  }
  actions.createNode(newNode)
}
```

The above code block creates a node called `"Test Node"` as seen from the `title` parameter. If this process is successful restarting the server will make the `allTestNode` query available at `http://localhost:8000/___graphql`.

> Libraries like Axios can be used to handle calls in the `gatsby-node.js` file

Though all plugins have the same structure, their name signals what functionality they provide. See the naming a plugin section for more information.