This guide will walk you through sourcing data from the filesystem.

Setup

This guide assumes that you have a Gatsby project set up. If you need to set up a project, please reference the Quick Start Guide.

It will also be useful if you are familiar with GraphiQL, a tool that helps you structure your queries correctly.

Using `gatsby-source-filesystem`

`gatsby-source-filesystem` is the Gatsby plugin for creating File nodes from the file system.

Install the plugin at the root of your Gatsby project:

```
npm install gatsby-source-filesystem
```

Then add it to your project's `gatsby-config.js` file:

```
module.exports = {
  siteMetadata: {
    title: `Your Site Name`,
  },
  plugins: [
    // highlight-start
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        name: `src`,
        path: `${__dirname}/src/`,
      },
    },
    // highlight-end
  ],
}
```

Save the `gatsby-config.js` file, and restart the Gatsby development server.

Open up GraphiQL.

If you bring up the autocomplete window, you'll see:

 The GraphiQL IDE showing the new autocomplete options provided by the gatsby-source-filesystem plugin

Hit `Enter` on `allFile` then type `Ctrl + Enter` to run a query.

 The GraphiQL IDE showing the results of a filesystem query

Delete the `id` from the query and bring up the autocomplete again (`Ctrl + Space`).

 The GraphiQL IDE showing autocomplete options

Try adding a number of fields to your query, pressing `Ctrl + Enter` each time to re-run the query. You'll see something like this:

 The GraphiQL IDE showing the results of the query

The result is an array of File "nodes" (node is a fancy name for an object in a "graph"). Each File object has the fields you queried for.

If you have multiple sets of data, you can query specific ones by specifying the `name` property from the config object in the `gatsby-config.js` file. In this case, `name` is set to `src`.

```
{
  resolve: `gatsby-source-filesystem`,
  options: {
    path: `${__dirname}/src`,
    name: `src`,
  },
},
```

You can then update your query using `sourceInstanceName` and the value of the `name` property in a filter like so.

```
{
  allFile(filter: { sourceInstanceName: { eq: "src" } }) {
    edges {
      node {
        relativePath
        prettySize
        extension
        birthTime
      }
    }
  }
}
```

Conditionally sourcing files using environment variables

You can conditionally set the `path` option using environment variables. For context, you might decide to do this if you're sourcing a lot of files and you're interested in only sourcing a smaller batch of files during `gatsby develop`. This is also helpful when you e.g. have a staging and production environment (signaled through environment variables).

The example below shows how to use `NODE_ENV` (which is automatically set to `development` during `gatsby develop`) to only source a smaller portion of the content during development. For `gatsby build` the full dataset will be used.

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        name: `markdown-pages`,
        path: process.env.NODE_ENV === `development` ?
`${__dirname}/src/content/2022` : `${__dirname}/src/content`,
      },
    },
    `gatsby-transformer-remark`,
  ],
}
```

Transforming File nodes

Once files have been sourced, various "transformer" plugins in the Gatsby ecosystem can then be used to transform File nodes into various other types of data. For example, a JSON file can be sourced using `gatsby-source-filesystem`, and then the resulting File nodes can be transformed into JSON nodes using `gatsby-transformer-json`.

Further reference and examples

For further reference, you may be interested in checking out the `gatsby-source-filesystem` package README, and various official and community starters that use the plugin.