gatsby-transformer-csv

Parses CSV files into JSON arrays.

Install

`npm install gatsby-transformer-csv`

Note: You generally will use this plugin together with the `gatsby-source-filesystem` plugin. `gatsby-source-filesystem` reads in the files then this plugin *transforms* the files into data you can query.

How to use

If you put your `.csv` files in `./src/data`:

```
// In your gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        name: `data`,
        path: `${__dirname}/src/data/`,
      },
    },
    `gatsby-transformer-csv`,
  ],
}
```

Configuration

The example above is the minimal configuration required to begin working. Additional customization of the parsing process is possible using the parameters listed in csvtojson. Any parameter listed on that page can be passed directly to the library using the plugin options.

For example, to pass the `noheader` option, you can configure like so:

```
// In your gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        name: `data`,
        path: `${__dirname}/src/data/`,
      },
```

```
    },
    {
      resolve: `gatsby-transformer-csv`,
      options: {
        noheader: true,
      },
    },
  ];
}
```

By default, files that do not have a `.csv` extension will not be parsed, but this can be configured using the `extensions` option which takes an array of strings.

For example, if you need to parse TSV files, you can configure the plugin like so:

```
// In your gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        name: `data`,
        path: `${__dirname}/src/data/`,
      },
    },
    {
      resolve: `gatsby-transformer-csv`,
      options: {
        extensions: [`tsv`],
        delimiter: '\t'
      },
    },
  ];
}
```

You can see an example project at https://github.com/gatsbyjs/gatsby/tree/master/examples/using-csv.

Parsing algorithm

By default each row is converted into a node with CSV headers as the keys.

If your project has a `letters.csv` with:

```
letter,value
a,65
b,66
c,67
```

The following three nodes would be created:

```
[
  { "letter": "a", "value": 65, "type": "LettersCsv" },
  { "letter": "b", "value": 66, "type": "LettersCsv" },
  { "letter": "c", "value": 67, "type": "LettersCsv" }
]
```

Alternatively the `typeName` plugin option can be used to modify this behaviour.

Its arguments are either a string denoting the type name or a function that accepts an argument object of `{ node, object }` which should return the string type name.

Two predefined functions are provided.

```
const { typeNameFromDir, typeNameFromFile } = require("gatsby-transformer-csv")
```

`typeNameFromFile` will produce a type per CSV file. When the `typeName` plugin option is undefined, this is the default case. A file name of `letters.csv` will produce a type of `LettersCsv`.

`typeNameFromDir` will produce a type per folder of CSVs. A folder called `things` containing CSVs will return a type of `ThingsCsv`.

As an example of a custom function, if the CSVs are in a group of folders, and you wish to create a group per folder with the suffix "Data". In this case a folder called `things` containing CSVs will return a type of `ThingsData`.

```
// In your gatsby-config.js
const _ = require(`lodash`)
const path = require(`path`)

module.exports = {
  plugins: [
    {
      resolve: `gatsby-transformer-csv`,
      options: {
        typeName: ({ node, object }) =>
          _.upperFirst(_.camelCase(`${path.basename(node.dir)} Data`)),
      },
    },
  ],
}
```

The suffix `Csv` is not added when providing your own function.

If you wanted to have a group per folder with the suffix "Csv", the `typeNameFromDir` provided function would be appropriate.

```
// In your gatsby-config.js
const { typeNameFromDir } = require("gatsby-transformer-csv")
```

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-transformer-csv`,
      options: {
        typeName: typeNameFromDir,
      },
    },
  ],
}
```

Alternate content behaviour

The `nodePerFile` plugin option can either be `false`, which creates a node per line like above, `true`, which creates a node per file, with the key `items` containing the content, or a string which is the key containing the content.

For example, if there are a series of csv files called `vegetables.csv`, `grains.csv`, the following config would produce the following result.

The config:

```
// In your gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-transformer-csv`,
      options: {
        typeName: () => `Foodstuffs`,
        nodePerFile: `ingredients`,
      },
    },
  ],
}
```

A query:

```
{
  allFoodstuffs {
    nodes {
      ingredients {
        ingredient
        amount
      }
      parent {
        ... on File {
          name
```

```
        }
      }
    }
  }
}
```

The result:


```
{
  "data": {
    "allFoodstuffs": {
      "nodes": [
        {
          "parent": {
            "name": "vegetables"
          },
          "ingredients": [
            {
              "ingredient": "potato",
              "amount": 32
            },
            {
              "ingredient": "lettuce",
              "amount": 12
            }
          ]
        },
        {
          "parent": {
            "name": "grains"
          },
          "ingredients": [
            {
              "ingredient": "barley",
              "amount": 2
            },
            {
              "ingredient": "wheat",
              "amount": 42
            }
          ]
        }
      ]
    }
  }
}
```

How to query

In the default configuration, items can be queried like this:

```
{
  allLettersCsv {
    edges {
      node {
        letter
        value
      }
    }
  }
}
```

Which would return:

```
{
  "allLettersCsv": {
    "edges": [
      {
        "node": {
          "letter": "a",
          "value": 65
        }
      },
      {
        "node": {
          "letter": "b",
          "value": 66
        }
      },
      {
        "node": {
          "letter": "c",
          "value": 67
        }
      }
    ]
  }
}
```