gatsby-transformer-json

Parses raw JSON strings into JavaScript objects e.g. from JSON files. Supports arrays of objects and single objects.

Install

`npm install gatsby-transformer-json`

If you want to transform JSON files, you also need to have `gatsby-source-filesystem` installed and configured so it points to your files.

How to use

In your `gatsby-config.js` :

```
module.exports = {
  plugins: [
    `gatsby-transformer-json`,
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        path: `./src/data/`,
      },
    },
  ],
}
```

Parsing algorithm

You can choose to structure your data as arrays of objects in individual files or as single objects spread across multiple files.

Array of Objects

The algorithm for arrays is to convert each item in the array into a node.

So if your project has a `letters.json` with

```
[{ "value": "a" }, { "value": "b" }, { "value": "c" }]
```

Then the following three nodes would be created:

```
[{ "value": "a" }, { "value": "b" }, { "value": "c" }]
```

Single Object

The algorithm for single JSON objects is to convert the object defined at the root of the file into a node. The type of the node is based on the name of the parent directory.

For example, let's say your project has a data layout like:

```
data/
    letters/
        a.json
        b.json
        c.json
```

Where each of `a.json`, `b.json` and `c.json` look like:

```
{ "value": "a" }
```

```
{ "value": "b" }
```

```
{ "value": "c" }
```

Then the following three nodes would be created:

```
[
  {
    "value": "a"
  },
  {
    "value": "b"
  },
  {
    "value": "c"
  }
]
```

How to query

Regardless of whether you choose to structure your data in arrays of objects or single objects, you'd be able to query your letters like:

```
{
  allLettersJson {
    edges {
      node {
        value
      }
    }
  }
}
```

Which would return:

```
{
  allLettersJson: {
    edges: [
      {
        node: {
          value: "a",
        },
      },
      {
        node: {
          value: "b",
        },
      },
      {
        node: {
          value: "c",
        },
      },
    ]
  }
}
```

Configuration options

`typeName` [string|function][optional]

The default naming convention documented above can be changed with either a static string value (e.g. to be able to query all json with a simple query):

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-transformer-json`,
      options: {
        typeName: `Json`, // a fixed string
      },
    },
  ],
}
```

```
{
  allJson {
    edges {
      node {
        value
      }
    }
  }
}
```

or a function that receives the following arguments:

- `node` : the graphql node that is being processed, e.g. a File node with json content
- `object` : a single object (either an item from an array or the whole json content)
- `isArray` : boolean, true if `object` is part of an array

```
[
  {
    "level": "info",
    "message": "hurray"
  },
  {
    "level": "info",
    "message": "it works"
  },
  {
    "level": "warning",
    "message": "look out"
  }
]
```

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-transformer-json`,
      options: {
        typeName: ({ node, object, isArray }) => object.level,
      },
    },
  ],
}
```

```
{
  allInfo {
    edges {
      node {
        message
      }
    }
  }
}
```

Examples

The gatsbygram example site uses this plugin.

Troubleshooting

If some fields are missing or you see the error on build:

> *There are conflicting field types in your data. GraphQL schema will omit those fields.*

It's probably because you have arrays of mixed values somewhere. For instance:

```
{
  "stuff": [25, "bob"],
  "orEven": [
    [25, "bob"],
    [23, "joe"]
  ]
}
```

If you can rewrite your data with objects, you should be good to go:

```
{
  "stuff": [{ "count": 25, "name": "bob" }],
  "orEven": [
    { "count": 25, "name": "bob" },
    { "count": 23, "name": "joe" }
  ]
}
```

Else, if your data doesn't have a consistent schema, like TopoJSON files, or you can't rewrite it, consider placing the JSON file inside the `static` folder and use the dynamic import syntax (`import('/static/myjson.json')`) within the `componentDidMount` lifecycle or the `useEffect` hook.

`id` and `jsonId` key

If your data contains an `id` key the transformer will automatically convert this key to `jsonId` as `id` is a reserved internal keyword for Gatsby.