Transforming data in Gatsby is plugin-driven. Transformer plugins take data fetched using source plugins, and process it into something more usable (e.g. JSON into JavaScript objects, and more).

Transforming Markdown into HTML

The `gatsby-transformer-remark` plugin can transform Markdown files to HTML.

Prerequisites

- A Gatsby site with `gatsby-config.js` and an `index.js` page
- A Markdown file saved in your Gatsby site `src` directory
- A source plugin installed, such as `gatsby-source-filesystem`
- The `gatsby-transformer-remark` plugin installed

Directions

1. Add the transformer plugin in your `gatsby-config.js`:

```
plugins: [
  // not shown: gatsby-source-filesystem for creating nodes to transform
  `gatsby-transformer-remark`
],
```

2. Add a GraphQL query to the `index.js` file of your Gatsby site to fetch `MarkdownRemark` nodes:

```
export const query = graphql`
  query {
    allMarkdownRemark {
      totalCount
      edges {
        node {
          id
          frontmatter {
            title
            date(formatString: "DD MMMM, YYYY")
          }
          excerpt
        }
      }
    }
  }
`
```

3. Restart the development server and open GraphiQL at `http://localhost:8000/___graphql`. Explore the fields available on the `MarkdownRemark` node.

Additional resources

- Tutorial on transforming Markdown to HTML using `gatsby-transformer-remark`
- Browse available transformer plugins in the Gatsby plugin library

Transforming images into grayscale using GraphQL

Prerequisites

- A Gatsby site with a `gatsby-config.js` file and an `index.js` page
- The `gatsby-image`, `gatsby-transformer-sharp`, and `gatsby-plugin-sharp` packages installed
- A source plugin installed, such as `gatsby-source-filesystem`
- An image (`.jpg`, `.png`, `.gif`, `.svg`, etc.) in the `src/images` folder

Directions

1. Edit your `gatsby-config.js` file to source images and configure plugins for Gatsby's GraphQL data layer. A common approach is to source them from an images directory using the `gatsby-source-filesystem` plugin:

```
plugins: [
  {
    resolve: `gatsby-source-filesystem`,
    options: {
      name: `images`,
      path: `${__dirname}/src/images`,
    },
  },
  `gatsby-transformer-sharp`,
  `gatsby-plugin-sharp`,
],
```

2. Query your image using GraphQL and apply a grayscale transformation to the image inline. The `relativePath` should be relative to the path you configured in `gatsby-source-filesystem`.

```
query {
   file(relativePath: { eq: "corgi.jpg" }) {
     childImageSharp {
       // highlight-next-line
       fluid(grayscale: true) {
         ...GatsbyImageSharpFluid
       }
     }
   }
 }
```

Note: You can find these and other parameters in your GraphQL playground located at `http://localhost:8000/__graphql`

3. Next import the `Img` component from "gatsby-image". You'll use this inside your JSX to display the image.

```
import React from "react"
import { useStaticQuery, graphql } from "gatsby"
```

```
import Layout from "../components/layout"
// highlight-next-line
import Img from "gatsby-image"

export default function Home() {
  const data = useStaticQuery(graphql`
    query {
     file(relativePath: { eq: "corgi.jpg" }) {
       childImageSharp {
         // highlight-next-line
         fluid(grayscale: true) {
           ...GatsbyImageSharpFluid
         }
       }
     }
   }
  `)
  return (
    <Layout>
      <h1>I love my corgi!</h1>
      // highlight-start
      <Img
        fluid={data.file.childImageSharp.fluid}
        alt="A corgi smiling happily"
      />
      // highlight-end
    </Layout>
  )
}
```

4. Run `gatsby develop` to start the development server.
5. View your image in the browser: `http://localhost:8000/`

Additional resources

- API docs, including grayscale and duotone query tips
- Gatsby Image docs
- Image processing examples