gatsby-transformer-remark

Parses Markdown files using remark.

Install

`npm install gatsby-transformer-remark`

How to use

```
// In your gatsby-config.js
plugins: [
  {
    resolve: `gatsby-transformer-remark`,
    options: {
      // Footnotes mode (default: true)
      footnotes: true,
      // GitHub Flavored Markdown mode (default: true)
      gfm: true,
      // Plugins configs
      plugins: [],
    },
  },
],
```

The following parts of `options` enable the `remark-footnotes` and `remark-gfm` plugins:

- `options.footnotes`
- `options.gfm`

A full explanation of how to use markdown in Gatsby can be found here: Adding Markdown Pages

There are many Gatsby Remark plugins which you can install to customize how Markdown is processed. Many of them are demoed at https://using-remark.gatsbyjs.org/. See also the source code for using-remark.

Parsing algorithm

It recognizes files with the following extensions as Markdown:

- `md`
- `markdown`

Each Markdown file is parsed into a node of type `MarkdownRemark`.

All frontmatter fields are converted into GraphQL fields through inference.

This plugin adds additional fields to the `MarkdownRemark` GraphQL type including `html`, `excerpt`, `headings`, etc. Other Gatsby plugins can also add additional fields.

How to query

A sample GraphQL query to get MarkdownRemark nodes:

```
{
  allMarkdownRemark {
    edges {
      node {
        html
        headings {
          depth
          value
        }
        frontmatter {
          # Assumes you're using title in your frontmatter.
          title
        }
      }
    }
  }
}
```

Getting table of contents

Using the following GraphQL query you'll be able to get the table of contents

```
{
  allMarkdownRemark {
    edges {
      node {
        html
        tableOfContents
      }
    }
  }
}
```

Configuring the `tableOfContents`

By default, `absolute` is set to `false`, generating a relative path. If you'd like to generate an absolute path, pass `absolute: true`. In that case, be sure to pass the `pathToSlugField` parameter, often `fields.slug`, to create absolute URLs. **Note** that providing a non-existent field will cause the result to be `null`. To alter the default values for `tableOfContents` generation, include values for `heading` (string) and/or `maxDepth` (number 1 to 6) in GraphQL query. If a value for `heading` is given, the first heading that matches will be omitted and the ToC is generated from the next heading of the same depth onwards. Value for `maxDepth` sets the maximum depth of the toc (i.e. if a maxDepth of 3 is set, only h1 to h3 headings will appear in the toc).

```
{
  allMarkdownRemark {
    edges {
      node {
        html
        tableOfContents(
          absolute: true
          pathToSlugField: "frontmatter.path"
          heading: "only show toc from this heading onwards"
          maxDepth: 2
        )
        frontmatter {
          # Assumes you're using path in your frontmatter.
          path
        }
      }
    }
  }
}
```

To pass default options to the plugin generating the `tableOfContents`, configure it in `gatsby-config.js` as shown below. The options shown below are the defaults used by the plugin.

```
// In your gatsby-config.js
plugins: [
  {
    resolve: `gatsby-transformer-remark`,
    options: {
      tableOfContents: {
        heading: null,
        maxDepth: 6,
      },
    },
  },
]
```

Excerpts

Length

By default, excerpts have a maximum length of 140 characters. You can change the default using the `pruneLength` argument. For example, if you need 500 characters, you can specify:

```
{
  allMarkdownRemark {
    edges {
      node {
        html
        excerpt(pruneLength: 500)
      }
    }
  }
}
```

Format

By default, Gatsby will return excerpts as plain text. This might be useful for populating opengraph HTML tags for SEO reasons. You can also explicitly specify a `PLAIN` format like so:

```
{
  allMarkdownRemark {
    edges {
      node {
        excerpt(format: PLAIN)
      }
    }
  }
}
```

It's also possible to ask Gatsby to return excerpts formatted as HTML. You might use this if you have a blog post whose excerpt contains markdown content -- e.g. header, link, etc. -- and you want these links to render as HTML.

```
{
  allMarkdownRemark {
    edges {
      node {
        excerpt(format: HTML)
      }
    }
  }
}
```

You can also get excerpts in Markdown format.

```
{
  allMarkdownRemark {
    edges {
      node {
        excerpt(format: MARKDOWN)
      }
    }
  }
}
```

`gray-matter` options

`gatsby-transformer-remark` uses gray-matter to parse Markdown frontmatter, so you can specify any of the options mentioned here in the `gatsby-config.js` file.

Example: Excerpts

If you don't want to use `pruneLength` for excerpts but a custom separator, you can specify an `excerpt_separator` in the `gatsby-config.js` file:

```
{
  "resolve": `gatsby-transformer-remark`,
  "options": {
    "excerpt_separator": `<!-- end -->`
  }
}
```

Any file that does not have the given `excerpt_separator` will fall back to the default pruning method.

Troubleshooting

Excerpts for non-latin languages

By default, `excerpt` uses `underscore.string/prune` which doesn't handle non-latin characters (https://github.com/epeli/underscore.string/issues/418).

If that is the case, you can set `truncate` option on `excerpt` field, like:

```
{
  markdownRemark {
    excerpt(truncate: true)
  }
}
```

Excerpts for HTML embedded in Markdown files

If your Markdown file contains HTML, `excerpt` will not return a value.

In that case, you can set an `excerpt_separator` in the `gatsby-config.js` file:

```
{
  "resolve": `gatsby-transformer-remark`,
  "options": {
    "excerpt_separator": `<!-- endexcerpt -->`
  }
}
```

Edit your Markdown files to include that HTML tag after the text you'd like to appear in the excerpt:

```
---
title: "my little pony"
date: "2017-09-18T23:19:51.246Z"
---

<p>Where oh where is that pony?</p>
<!-- endexcerpt -->
<p>Is he in the stable or down by the stream?</p>
```

Then specify `MARKDOWN` as the format in your GraphQL query:

```
{
  markdownRemark {
    excerpt(format: MARKDOWN)
  }
}
```