Migrating Remark to MDX

For people who already have an existing blog using `gatsby-transformer-remark` but want to use MDX, you can swap out the Remark transformer plugin with `gatsby-plugin-mdx` and touch little code otherwise.

Prerequisites

- An existing Gatsby site that builds pages using `gatsby-transformer-remark` (gatsby-starter-blog will be used as a reference in this guide)

Adding in gatsby-plugin-mdx

Add the `gatsby-plugin-mdx` plugin (and its peer dependencies) to your `package.json` file and remove the `gatsby-transformer-remark` plugin.

```
npm install @mdx-js/mdx@v1 @mdx-js/react@v1 gatsby-plugin-mdx
npm remove gatsby-transformer-remark
```

Replacing gatsby-transformer-remark with gatsby-plugin-mdx

In your `gatsby-config.js` file, replace `gatsby-transformer-remark` with `gatsby-plugin-mdx`. Most sub-plugins of `gatsby-transformer-remark` can still work with `gatsby-plugin-mdx` by updating the plugins option to `gatsbyRemarkPlugins`.

```
{
- resolve: `gatsby-transformer-remark`
+ resolve: `gatsby-plugin-mdx`
  options: {
-   plugins: [
+   gatsbyRemarkPlugins: [
```

Update file extensions

Where your Markdown files live, changing their individual file extensions from `.md` to `.mdx` will pick them up with the new configuration.

Alternatively, you can tell `gatsby-plugin-mdx` to accept both `.md` and `.mdx` files by adding the `extensions` option in your `gatsby-config.js` entry.

```
  {
    resolve: `gatsby-plugin-mdx`,
    options: {
      extensions: [`.md`, `.mdx`], // highlight-line
    },
  },
```

Now with this addition, `gatsby-plugin-mdx` will see files that end with both `.mdx` or `.md`.

Update gatsby-node.js

In the `createPages` API call, when you query for `allMarkdownRemark`, replace it with `allMdx`.

```
const result = await graphql(
  `
    {
-     allMarkdownRemark(
+     allMdx(
        sort: { fields: [frontmatter___date], order: DESC }
        limit: 1000
      ) {
```

Don't forget to update the `posts` constant by replacing `allMarkdownRemark` with `allMdx`.

```
-const posts = result.data.allMarkdownRemark.nodes
+const posts = result.data.allMdx.nodes
```

Also, update `onCreateNode` which creates the blog post slugs to watch for the node type of `Mdx` instead of `MarkdownRemark`.

```
exports.onCreateNode = ({ node, actions, getNode }) => {
  const { createNodeField } = actions

-  if (node.internal.type === `MarkdownRemark`) {
+  if (node.internal.type === `Mdx`) {
```

Update usage in pages

Similar to `gatsby-node.js`, wherever you use `allMarkdownRemark` in a GraphQL query, change it to `allMdx`.

Then in your blogpost template, to render the MDX, pull in the `MDXRenderer` React component from `gatsby-plugin-mdx`.

```
import { MDXRenderer } from "gatsby-plugin-mdx"
```

And in the GraphQL query, change the `html` field in `mdx` to `body`.

```
mdx(fields: { slug: { eq: $slug } }) {
  id
  excerpt(pruneLength: 160)
  body // highlight-line
  frontmatter {
    ...
  }
}
```

And finally swap out the component with `dangerouslySetInnerHTML` to a `MDXRenderer` component:

```
const post = data.mdx

// ...

-<section dangerouslySetInnerHTML={{ __html: post.html }} />
+<MDXRenderer>{post.body}</MDXRenderer>
```

Update Markdown files that include HTML code

As MDX uses JSX instead of HTML, examine anywhere you put in HTML and make sure that it is valid JSX. There might be some reserved words that need to be modified, or attributes to convert to camelCase.

For instance, any HTML component with the `class` attribute needs to be changed to `className`.

```
-<span class="highlight">Hello World</span>
+<span className="highlight">Hello World</span>
```

Additional resources

- Follow Importing and Using Components in MDX to find out how you can insert React components in your MDX files.
- Follow Using MDX Plugins on how to add and use Gatsby Remark or Remark plugins to your MDX site.