On this page

MDX Plugins

Sometimes, you may want to extend or tweak your Markdown syntax. For example:

- How do I embed youtube videos using the image syntax (`![](https://youtu.be/yKNxeF4KMsY)`)?
- How do I style links that are on their own lines differently, e.g., as a social card?
- How do I make every page start with a copyright notice?

And the answer is: create an MDX plugin! MDX has a built-in plugin system that can be used to customize how the Markdown files will be parsed and transformed to JSX. There are three typical use-cases of MDX plugins:

- Using existing remark plugins or rehype plugins;
- Creating remark/rehype plugins to transform the elements generated by existing MDX syntax;
- Creating remark/rehype plugins to introduce new syntaxes to MDX.

If you play with the MDX playground, you would notice that the MDX transpilation has two intermediate steps: Markdown AST (MDAST), and Hypertext AST (HAST), before arriving at the final JSX output. MDX plugins also come in two forms:

- **Remark**: processes the Markdown AST.
- **Rehype**: processes the Hypertext AST.

TIP

Use plugins to introduce shorter syntax for the most commonly used JSX elements in your project. The admonition syntax that we offer is also generated by a Remark plugin, and you could do the same for your own use case.

Default plugins

Docusaurus injects some default Remark plugins during Markdown processing. These plugins would:

- Generate the table of contents;
- Add anchor links to each heading;
- Transform images and links to `require()` calls.
- …

These are all typical use-cases of Remark plugins, which can also be a source of inspiration if you want to implement your own plugin.

Installing plugins

An MDX plugin is usually an npm package, so you install them like other npm packages using npm. Take the math plugins as an example.

npm
Yarn
pnpm

```
npm install --save remark-math@3 rehype-katex@4
```

NOTE

There's recently a trend in the Remark/Rehype ecosystem to migrate to ES Modules, a new JavaScript module system, which

Docusaurus doesn't support yet. Please make sure your installed plugin version is CommonJS-compatible before we officially support ESM. Alternatively, you can read about using dynamic `import()` as a workaround in the tutorial of installing `rehype-katex`.

- How are `remark-math` and `rehype-katex` different?

Next, add them to the plugin options through plugin or preset config in `docusaurus.config.js`:

docusaurus.config.js
```
const math = require('remark-math');
const katex = require('rehype-katex');

module.exports = {
  title: 'Docusaurus',
  tagline: 'Build optimized websites quickly, focus on your content',
  presets: [
    [
      '@docusaurus/preset-classic',
      {
        docs: {
          remarkPlugins: [math],
          rehypePlugins: [katex],
        },
      },
    ],
  ],
};
```

Configuring plugins

Some plugins can be configured and accept their own options. In that case, use the `[plugin, pluginOptions]` syntax, like this:

docusaurus.config.js
```
module.exports = {
  presets: [
    [
      '@docusaurus/preset-classic',
      {
        docs: {
          remarkPlugins: [math],
          rehypePlugins: [
            [katex, {strict: false}],
          ],
        },
      },
    ],
  ],
};
```

You should check your plugin's documentation for the options it supports.

Creating new rehype/remark plugins

If there isn't an existing package that satisfies your customization need, you can create your own MDX plugin.

NOTE

The writeup below is **not** meant to be a comprehensive guide to creating a plugin, but just an illustration of how to make it work with Docusaurus. Visit the Remark or Rehype documentation for a more in-depth explanation of how they work.

For example, let's make a plugin that visits every `h2` heading and adds a `Section X.` prefix. First, create your plugin source file anywhere—you can even publish it as a separate npm package and install it like explained above. We would put ours at `src/remark/section-prefix.js`. A remark/rehype plugin is just a function that receives the `options` and returns a `transformer` that operates on the AST.

```
const visit = require('unist-util-visit');

const plugin = (options) => {
  const transformer = async (ast) => {
    let number = 1;
    visit(ast, 'heading', (node) => {
      if (node.depth === 2 && node.children.length > 0) {
        node.children.unshift({
          type: 'text',
          value: `Section ${number}. `,
        });
        number++;
      }
    });
  };
  return transformer;
};

module.exports = plugin;
```

You can now import your plugin in `docusaurus.config.js` and use it just like an installed plugin!

docusaurus.config.js
```
const sectionPrefix = require('./src/remark/section-prefix');

module.exports = {
  presets: [
    [
      '@docusaurus/preset-classic',
      {
        docs: {
          remarkPlugins: [sectionPrefix],
        },
      },
    ],
  ],
};
```

TIP

The `transformer` has a second parameter `vfile` which is useful if you need to access the current Markdown file's path.

```
const plugin = (options) => {
  const transformer = async (ast, vfile) => {
    ast.children.unshift({
      type: 'text',
      value: `The current file path is ${vfile.path}`,
    });
  };
  return transformer;
};
```

Our `transformImage` plugin uses this parameter, for example, to transform relative image references to `require()` calls.

NOTE

The default plugins of Docusaurus would operate before the custom remark plugins, and that means the images or links have been converted to JSX with `require()` calls already. For example, in the example above, the table of contents generated is still the same even when all `h2` headings are now prefixed by `Section X.`, because the TOC-generating plugin is called before our custom plugin. If you need to process the MDAST before the default plugins do, use the `beforeDefaultRemarkPlugins` and `beforeDefaultRehypePlugins`.

docusaurus.config.js
```
module.exports = {
  presets: [
    [
      '@docusaurus/preset-classic',
      {
        docs: {
          beforeDefaultRemarkPlugins: [sectionPrefix],
        },
      },
    ],
  ],
};
```

This would make the table of contents generated contain the `Section X.` prefix as well.

Edit this page
*Last updated on **Mar 23, 2023** by **Sébastien Lorber***

Previous
« Markdown links

Next
Math Equations »