Assets

Sometimes you want to link to assets (e.g. docx files, images...) directly from Markdown files, and it is convenient to co-locate the asset next to the Markdown file using it.

Let's imagine the following file structure:

```
# Your doc
/website/docs/myFeature.mdx

# Some assets you want to use
/website/docs/assets/docusaurus-asset-example-banner.png
/website/docs/assets/docusaurus-asset-example.docx
```

Images

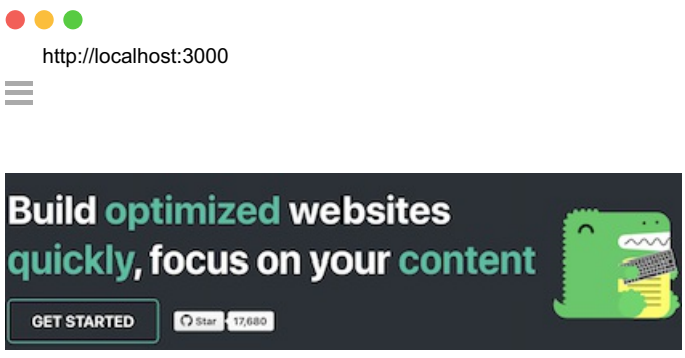
You can display images in three different ways: Markdown syntax, CJS require, or ES imports syntax.

Markdown syntax
CommonJS require
Import statement

Display images using simple Markdown syntax:

```
![Example banner](./assets/docusaurus-asset-example-banner.png)
```

All of the above result in displaying the image:



NOTE

If you are using @docusaurus/plugin-ideal-image, you need to use the dedicated image component, as documented.

Files

In the same way, you can link to existing assets by `require`'ing them and using the returned URL in `video`s, `a` anchor links, etc.

```
# My Markdown page

<a target="\_blank" href={require('./assets/docusaurus-asset-example.docx').default}> Download this docx </a>

or

[Download this docx using Markdown](./assets/docusaurus-asset-example.docx)
```

http://localhost:3000

Download this docx

Download this docx using Markdown

MARKDOWN LINKS ARE ALWAYS FILE PATHS

If you use the Markdown image or link syntax, all asset paths will be resolved as file paths by Docusaurus and automatically converted to `require()` calls. You don't need to use `require()` in Markdown unless you use the JSX syntax, which you do have to handle yourself.

Inline SVGs

Docusaurus supports inlining SVGs out of the box.

```
import DocusaurusSvg from './docusaurus.svg';

<DocusaurusSvg />;
```

http://localhost:3000

This can be useful if you want to alter the part of the SVG image via CSS. For example, you can change one of the SVG colors based on the current theme.

```
import DocusaurusSvg from './docusaurus.svg';

<DocusaurusSvg className="themedDocusaurus" />;
[data-theme='light'] .themedDocusaurus [fill='#FFFF50'] {
  fill: greenyellow;
}

[data-theme='dark'] .themedDocusaurus [fill='#FFFF50'] {
  fill: seagreen;
}
```

http://localhost:3000

Themed Images

Docusaurus supports themed images: the `ThemedImage` component (included in the themes) allows you to switch the image source based on the current theme.

```
import ThemedImage from '@theme/ThemedImage';

<ThemedImage
  alt="Docusaurus themed image"
  sources={{
    light: useBaseUrl('/img/docusaurus_light.svg'),
    dark: useBaseUrl('/img/docusaurus_dark.svg'),
  }}
/>;
```

http://localhost:3000

GitHub-style themed images

GitHub uses its own image theming approach with path fragments, which you can easily implement yourself.

To toggle the visibility of an image using the path fragment (for GitHub, it's `#gh-dark-mode-only` and `#gh-light-mode-only`), add the following to your custom CSS (you can also use your own suffix if you don't want to be coupled to GitHub):

src/css/custom.css

```
[data-theme='light'] img[src$='#gh-dark-mode-only'],
[data-theme='dark'] img[src$='#gh-light-mode-only'] {
  display: none;
}
```

```
![Docusaurus themed image](/img/docusaurus_keytar.svg#gh-light-mode-only)![Docusaurus themed image](/img/docu
```

http://localhost:3000

Static assets

If a Markdown link or image has an absolute path, the path will be seen as a file path and will be resolved from the static directories. For example, if you have configured static directories to be `['public', 'static']`, then for the following image:

my-doc.md
```
![An image from the static](/img/docusaurus.png)
```

Docusaurus will try to look for it in both `static/img/docusaurus.png` and `public/img/docusaurus.png`. The link will then be converted to a `require()` call instead of staying as a URL. This is desirable in two regards:

1. You don't have to worry about the base URL, which Docusaurus will take care of when serving the asset;
2. The image enters Webpack's build pipeline and its name will be appended by a hash, which enables browsers to aggressively cache the image and improves your site's performance.

If you intend to write URLs, you can use the `pathname://` protocol to disable automatic asset linking.

```
![banner](pathname:///img/docusaurus-asset-example-banner.png)
```

This link will be generated as `<img src="/img/docusaurus-asset-example-banner.png" alt="banner" />`, without any processing or file existence checking.

Changelog

GitHub

Twitter

Legal

Privacy

Terms

Data Policy

Cookie Policy