gatsby-remark-embed-snippet

Embeds the contents of specified files as code snippets.

Installation

Note: This plugin depends on gatsby-remark-prismjs and gatsby-transformer-remark plugins

```
npm install gatsby-remark-embed-snippet gatsby-remark-prismjs gatsby-transformer-remark prismjs
```

Configuration

> ***Important***: *You must add* `gatsby-remark-embed-snippet` *before* `gatsby-remark-prismjs` *in your plugins array! Otherwise, this plugin will not work because the code snippet files first need to be inlined before they can be transformed into code blocks. For more information, see gatsby-remark-prismjs.*

To use `gatsby-remark-embed-snippet` plugin:

```
// gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-transformer-remark`,
      options: {
        plugins: [
          {
            resolve: `gatsby-remark-embed-snippet`,
            options: {},
          },
          {
            resolve: `gatsby-remark-prismjs`,
            options: {},
          },
        ],
      },
    },
  ],
}
```

Plugin option

option	description
`directory`	Specify the directory where the code snippet files are located. If this option is omitted, this plugin will look for the code snippet files in the same directory as the markdown file.

Example 1: Specify that code snippet files are under the root directory

```
// In gatsby-config.js
{
  resolve: `gatsby-remark-embed-snippet`,
  options: {
    directory: `${__dirname}`
  }
},
```

Example 2: Specify that code snippet files are under a directory called `snippets`

```
// In gatsby-config.js
{
  resolve: `gatsby-remark-embed-snippet`,
  options: {
    directory: `${__dirname}/snippets/`
  }
},
```

Sample Usage I

Suppose you have the following file/folder structure and you want to embed `javascript-code.js` and `html-code.html` files as code snippets inside the Markdown file `index.md`.

```
.
├── content
│   └── my-first-post
│       ├── index.md
│       ├── javascript-code.js
│       └── html-code.html
```

Add the following syntax to the Markdown file `index.md` to embed `javascript-code.js` and `html-code.html` as code snippets:

`index.md`:

```
# Sample JavaScript

`embed:javascript-code.js`

# Sample HTML

`embed:html-code.html`
```

The resulting HTML generated from the Markdown file above would look something like this:

```
<h1>Sample JavaScript</h1>
<div class="gatsby-highlight">
  <pre class="language-jsx">
```

```
    <code>
      <!-- Embedded javascript-code.js content here ... -->
    </code>
  </pre>
</div>

<h1>Sample HTML</h1>
<div class="gatsby-highlight">
  <pre class="language-html">
    <code>
      <!-- Embedded html-code.html content here ... -->
    </code>
  </pre>
</div>
```

Sample Usage II

Suppose you have the following file/folder structure and you want to embed `javascript-code.js` and `html-code.html` files as code snippets inside the Markdown file `my-first-post.md`.

```
.
├── content
│   └── my-first-post.md
├── snippets
│   ├── javascript-code.js
│   └── html-code.html
```

Use `directory` plugin option to tell `gatsby-remark-embed-snippet` plugin that code snippet files are located under a directory called `snippets`:

```
// gatsby-config.js
{
  resolve: `gatsby-remark-embed-snippet`,
  options: {
    directory: `${__dirname}/snippets/`,
  },
},
```

Add the following syntax to the Markdown file `my-first-post.md` to embed `javascript-code.js` and `html-code.html` as code snippets:

`my-first-post.md`:

```
# Sample JavaScript 2

`embed:javascript-code.js`

# Sample HTML 2
```

```
`embed:html-code.html`
```

The resulting HTML generated from the markdown file above would look something like this:

```
<h1>Sample JavaScript 2</h1>
<div class="gatsby-highlight">
  <pre class="language-jsx">
    <code>
      <!-- Embedded javascript-code.js content here ... -->
    </code>
  </pre>
</div>

<h1>Sample HTML 2</h1>
<div class="gatsby-highlight">
  <pre class="language-html">
    <code>
      <!-- Embedded html-code.html content here ... -->
    </code>
  </pre>
</div>
```

Code snippet syntax highlighting

Highlighting Lines

You can specify specific lines for Prism to highlight using `highlight-line` and `highlight-next-line` comments. You can also specify a range of lines to highlight, relative to a `highlight-range` comment.

JavaScript example:

````
```jsx
import React from "react"
import ReactDOM from "react-dom"

const name = "Brian" // highlight-line

ReactDOM.render(
  <div>
    {/* highlight-range{1-3} */}
    <h1>Hello, ${name}!</h1>
  </div>,
  document.getElementById("root")
)
```
````

```
import React from "react"
import ReactDOM from "react-dom"
```

```
const name = "Brian" // highlight-line

ReactDOM.render(
  <div>
    {/* highlight-range{1-3} */}
    <h1>Hello, ${name}!</h1>
  </div>,
  document.getElementById("root")
)
```

CSS example:

````
```css
html {
  /* highlight-range{1-2} */
  height: 100%;
  width: 100%;
}

* {
  box-sizing: border-box; /* highlight-line */
}
```
````

```
html {
  /* highlight-range{1-2} */
  height: 100%;
  width: 100%;
}

* {
  box-sizing: border-box; /* highlight-line */
}
```

HTML example:

````
```html
<html>
  <body>
    <h1>highlight me</h1> <!-- highlight-line -->
    <p>
      <!-- highlight-next-line -->
      And me
    </p>
  </body>
</html>
```
````

```
<html>
  <body>
    <h1>highlight me</h1> <!-- highlight-line -->
    <p>
      <!-- highlight-next-line -->
      And me
    </p>
  </body>
</html>
```

YAML example:

````
```yaml
foo: "highlighted" # highlight-line
bar: "not highlighted"
# highlight-range{1-2}
baz: "highlighted"
quz: "highlighted"
```
````

```yaml
foo: "highlighted" # highlight-line
bar: "not highlighted"
# highlight-range{1-2}
baz: "highlighted"
quz: "highlighted"
```

Hide Lines

It's also possible to specify a range of lines to be hidden.

You can either specify line ranges in the embed using the syntax:

- #Lx - Embed one line from a file
- #Lx-y - Embed a range of lines from a file
- #Lx-y,a-b - Embed non-consecutive ranges of lines from a file

Markdown example:

````
This is the JSX of my app:

`embed:App.js#L6-8`
````

With this example snippet:

```
import React from "react"
import ReactDOM from "react-dom"

function App() {
  return (
    <div className="App">
```

```
      <h1>Hello world</h1>
    </div>
  )
}
```

Will produce something like this:

```
This is the JSX of my app:

    <div className="App">
      <h1>Hello world</h1>
    </div>
```

Omitting lines

It's also possible to specify ranges of lines to be hidden from an embedded file by adding `// hide-range` comments to your files.

JavaScript example:

```
// hide-range{1-2}
import React from "react"
import ReactDOM from "react-dom"

function App() {
  return (
    <div className="App">
      <ul>
        <li>Not hidden</li>
        <li>Not hidden</li>
        {/* hide-range{1-2} */}
        <li>Hidden</li>
        <li>Hidden</li>
        {/* hide-next-line */}
        <li>Hidden</li>
      </ul>
    </div>
  )
}

// hide-range{1-2}
const rootElement = document.getElementById("root")
ReactDOM.render(<App />, rootElement)
```

Will produce something like this:

```
function App() {
  return (
    <div className="App">
      <ul>
```

```
        <li>Not hidden</li>
        <li>Not hidden</li>
      </ul>
    </div>
  )
}
```

Specifying snippets by name

As an alternative to selecting a range of lines from a file, you can add `start-snippet{snippet-name}` and `end-snippet{snippet-name}` in comments in your files. The inclusion of a name for a snippet allows you to create an example file that contains multiple snippets that you reference from different places.

You can specify that you want to only include a named snippet from the embed by using the syntax `{snippet: "snippet-name"}`.

JavaScript example:

```
The function to use is:

`embed:api.js{snippet: "funcA"}`

And it is invoked via:

`embed:api.js{snippet: "invokeA"}`
```

With this example file `api.js`:

```
// start-snippet{funcA}
function factorial(x) {
    if (x <= 1) return 1
    else return x * factorial(x - 1)
}
// end-snippet{funcA}

function display() {
    let x = 5
    // start-snippet{invokeA}
    let xfact = factorial(x)
    // end-snippet{invokeA}
    println!(`{} factorial is {}`, x, xfact)
}
```

Will produce something like this:

```
The function to use is:

function factorial(x) {
if (x <= 1) return 1
else return x \* factorial(x - 1)
```

```
}

And it is invoked via:

let xfact = factorial(x)
```

Code snippet syntax highlighting

Highlighting Lines

You can specify specific lines for Prism to highlight using `highlight-line` and `highlight-next-line` comments. You can also specify a range of lines to highlight, relative to a `highlight-range` comment.

JavaScript example:

```
import React from "react"
import ReactDOM from "react-dom"

const name = "Brian" // highlight-line

ReactDOM.render(
  <div>
    {/* highlight-range{1-3} */}
    <h1>Hello, ${name}!</h1>
  </div>,
  document.getElementById("root")
)
```

CSS example:

```
html {
  /* highlight-range{1-2} */
  height: 100%;
  width: 100%;
}

* {
  box-sizing: border-box; /* highlight-line */
}
```

HTML example:

```
<html>
  <body>
    <h1>highlight me</h1> <!-- highlight-line -->
    <p>
      <!-- highlight-next-line -->
      And me
    </p>
```

```
  </body>
</html>
```

YAML example:

```yaml
foo: "highlighted" # highlight-line
bar: "not highlighted"
# highlight-range{1-2}
baz: "highlighted"
quz: "highlighted"
```