Marked Documentation

The `parse` function

```
import { marked } from 'marked';
marked.parse(markdownString [,options])
```

Argument	Type	Notes
markdownString	`string`	String of markdown source to be compiled.
options	`object`	Hash of options. Can also use `marked.use` to set global options

Alternative using reference

```
// Create reference instance
import { marked } from 'marked';

// Set options
marked.use({
  async: true,
  pedantic: false,
  gfm: true,
});

// Compile
console.log(marked.parse(markdownString));
```

Options

Member	Type	Default	Since	Notes
async	`boolean`	`false`	4.1.0	If true, `walkTokens` functions can be async and `marked.parse` will return a promise that resolves when all walk tokens functions resolve.
breaks	`boolean`	`false`	v0.2.7	If true, add ` ` on a single line break (copies GitHub behavior on comments, but not on rendered markdown files). Requires `gfm` be `true`.
gfm	`boolean`	`true`	v0.2.1	If true, use approved GitHub Flavored Markdown (GFM) specification.
pedantic	`boolean`	`false`	v0.2.1	If true, conform to the original `markdown.pl` as much as possible. Don't fix original markdown bugs or behavior. Turns off and overrides `gfm`.
renderer	`object`	`new Renderer()`	v0.3.0	An object containing functions to render tokens to HTML. See extensibility for more details.
silent	`boolean`	`false`	v0.2.7	If true, the parser does not throw any exception or log any warning. Any error will be returned as a string.
tokenizer	`object`	`new Tokenizer()`	v1.0.0	An object containing functions to create tokens from markdown. See extensibility for more details.
walkTokens	`function`	`null`	v1.1.0	A function which is called for every token. See extensibility for more details.
baseUrl (**deprecated**)	`string`	`null`	v0.3.9	Deprecated in v5.0.0 use `marked-base-url` to prefix url for any relative link.
headerIds (**deprecated**)	`boolean`	`true`	v0.4.0	Deprecated in v5.0.0 use `marked-gfm-heading-id` to include an `id` attribute when emitting headings (h1, h2, h3, etc).
headerPrefix (**deprecated**)	`string`	`''`	v0.3.0	Deprecated in v5.0.0 use `marked-gfm-heading-id` to add a string to prefix the `id` attribute when emitting headings (h1, h2, h3, etc).
highlight (**deprecated**)	`function`	`null`	v0.3.0	Deprecated in v5.0.0 use `marked-highlight` to add highlighting to code blocks.
langPrefix (**deprecated**)	`string`	`'language-'`	v0.3.0	Deprecated in v5.0.0 use `marked-highlight` to prefix the className in a `<code>` block. Useful for syntax highlighting.
mangle (**deprecated**)	`boolean`	`true`	v0.3.4	Deprecated in v5.0.0 use `marked-mangle` to mangle email addresses.
sanitize (**deprecated**)	`boolean`	`false`	v0.2.1	If true, sanitize the HTML passed into `markdownString` with the `sanitizer` function. **Warning**: This feature is deprecated and it should NOT be used as it cannot be considered secure. Instead use a sanitize library, like DOMPurify (recommended), sanitize-html or insane on the output HTML!
sanitizer (**deprecated**)	`function`	`null`	v0.3.4	A function to sanitize the HTML passed into `markdownString`.
smartypants (**deprecated**)	`boolean`	`false`	v0.2.9	Deprecated in v5.0.0 use `marked-smartypants` to use "smart" typographic punctuation for things like quotes and dashes.
xhtml (**deprecated**)	`boolean`	`false`	v0.3.2	Deprecated in v5.0.0 use `marked-xhtml` to emit self-closing HTML tags for void elements (, <img/>, etc.) with a "/" as required by XHTML.

Known Extensions

Marked can be extended using custom extensions. This is a list of extensions that can be used with `marked.use(extension)`.

Name	Package Name	Description
	`marked-`	

Admonition	`admonition-extension`	Admonition extension
Base URL	`marked-base-url`	Prefix relative urls with a base URL.
Bidi	`marked-bidi`	Add Bidirectional text support to the HTML
Custom Heading ID	`marked-custom-heading-id`	Specify a custom heading id in headings with the Markdown Extended Syntax `# heading {#custom-id}`
Emoji	`marked-emoji`	Add emoji support like on GitHub
Extended Tables	`marked-extended-tables`	Extends the standard Github-Flavored tables to support advanced features: Column Spanning, Row Spanning, Multi-row headers
GFM Heading ID	`marked-gfm-heading-id`	Use `github-slugger` to create the heading IDs and allow a custom prefix.
Highlight	`marked-highlight`	Highlight code blocks
Katex Code	`marked-katex-extension`	Render katex code
LinkifyIt	`marked-linkify-it`	Use linkify-it for urls
Mangle	`marked-mangle`	Mangle mailto links with HTML character references
Misskey-flavored Markdown	`marked-mfm`	Custom extension for Misskey-flavored Markdown.
Smartypants	`marked-smartypants`	Use smartypants to use "smart" typographic punctuation for things like quotes and dashes.
XHTML	`marked-xhtml`	Emit self-closing HTML tags for void elements (, <img/>, etc.) with a "/" as required by XHTML.

Inline Markdown

You can parse inline markdown by running markdown through `marked.parseInline`.

```
const blockHtml = marked.parse('**strong** _em_');
console.log(blockHtml); // '<p><strong>strong</strong> <em>em</em></p>'

const inlineHtml = marked.parseInline('**strong** _em_');
console.log(inlineHtml); // '<strong>strong</strong> <em>em</em>'
```

Highlighting

Use `marked-highlight` to highlight code blocks.

Workers

To prevent ReDoS attacks you can run marked on a worker and terminate it when parsing takes longer than usual.

Marked can be run in a worker thread on a node server, or a web worker in a browser.

Node Worker Thread

```
// markedWorker.js

import { marked } from 'marked';
import { parentPort } from 'worker_threads';

parentPort.on('message', (markdownString) => {
  parentPort.postMessage(marked.parse(markdownString));
});
```

```
// index.js

import { Worker } from 'worker_threads';
const markedWorker = new Worker('./markedWorker.js');

const markedTimeout = setTimeout(() => {
  markedWorker.terminate();
  throw new Error('Marked took too long!');
}, timeoutLimit);

markedWorker.on('message', (html) => {
  clearTimeout(markedTimeout);
  console.log(html);
  markedWorker.terminate();
});

markedWorker.postMessage(markdownString);
```

Web Worker

> **NOTE**: Web Workers send the payload from `postMessage` in an object with the payload in a `.data` property

```
// markedWorker.js

importScripts('path/to/marked.min.js');

onmessage = (e) => {
  const markdownString = e.data
  postMessage(marked.parse(markdownString));
};
```

```
// script.js

const markedWorker = new Worker('./markedWorker.js');

const markedTimeout = setTimeout(() => {
  markedWorker.terminate();
  throw new Error('Marked took too long!');
}, timeoutLimit);

markedWorker.onmessage = (e) => {
  clearTimeout(markedTimeout);
  const html = e.data;
  console.log(html);
  markedWorker.terminate();
};

markedWorker.postMessage(markdownString);
```

CLI Extensions

You can use extensions in the CLI by creating a new CLI that imports marked and the marked binary.

```
// file: myMarked
#!/usr/bin/node

import { marked } from 'marked';
import customHeadingId from 'marked-custom-heading-id';

marked.use(customHeadingId());

import 'marked/bin/marked';
```

```
$ ./myMarked -s "# heading {#custom-id}"
<h1 id="custom-id">heading</h1>
```