






chjj / **marked**


 Watch 259


 Star 8,365


 Fork 1,186


 Code

 Issues 251

 Pull requests 126

 Wiki

 Pulse

 Graphs

A markdown parser and compiler. Built for speed.

 606 commits

 16 branches

 37 releases

 45 contributors

Branch: master

New pull request



















New fileFind file

HTTPS

https://github.com/chjj/




Download ZIP

 chjj v0.3.5	Latest commit 88ce4df on 31 Jul 2015
 bin	use --test for running tests from bin/marked. 3 years ago
 doc	fix doc formatting. 2 years ago
 lib	fix 2 failing tests. see #616. 6 months ago
 man	Add -mangle/-no-mangle to enable/disable mangling of email addresses. a year ago
 test	place new tests in test/new. 6 months ago
 .gitignore	1. fix benchmark for showdown 3 years ago
 .npmignore	add .npmignore. minor. 3 years ago
 .travis.yml	configure Travis CI services 3 years ago
 Gulpfile.js	Add a Gulpfile 11 months ago
 LICENSE	v0.3.1 2 years ago
 Makefile	add minified version of marked v0.3.2. fixes #293. 2 years ago
 README.md	Fix tiny typo 11 months ago
 bower.json	v0.3.4 6 months ago
 component.json	v0.3.4 6 months ago
 index.js	index 5 years ago
 marked.min.js	update marked.min.js. 6 months ago
 package.json	v0.3.5 6 months ago

README.md

marked

A full-featured markdown parser and compiler, written in JavaScript. Built for speed.

 npm package 0.3.5

Install

```
npm install marked --save
```

Usage

Minimal usage:

```
var marked = require('marked');
```

```
console.log(marked('I am using __markdown__.'));
// Outputs: <p>I am using <strong>markdown</strong>.</p>
```

Example setting options with default values:

```
var marked = require('marked');
marked.setOptions({
  renderer: new marked.Renderer(),
  gfm: true,
  tables: true,
  breaks: false,
  pedantic: false,
  sanitize: true,
  smartLists: true,
  smartypants: false
});

console.log(marked('I am using __markdown__.'));
```

Browser

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Marked in the browser</title>
  <script src="lib/marked.js"></script>
</head>
<body>
  <div id="content"></div>
  <script>
    document.getElementById('content').innerHTML =
      marked('# Marked in browser\n\nRendered by **marked**.');
  </script>
</body>
</html>
```

marked(markdownString [,options] [,callback])

markdownString

Type: string

String of markdown source to be compiled.

options

Type: object

Hash of options. Can also be set using the `marked.setOptions` method as seen above.

callback

Type: function

Function called when the `markdownString` has been fully parsed when using async highlighting. If the `options` argument is omitted, this can be used as the second argument.

Options

highlight

Type: function

A function to highlight code blocks. The first example below uses async highlighting with `node-pygmentize-bundled`, and the second is a synchronous example using `highlight.js`:

```
var marked = require('marked');

var markdownString = ``js\n console.log("hello"); \n``;

// Async highlighting with pygmentize-bundled
marked.setOptions({
  highlight: function (code, lang, callback) {
    require('pygmentize-bundled')({ lang: lang, format: 'html' }, code, function (err, result) {
      callback(err, result.toString());
    });
  }
});

// Using async version of marked
marked(markdownString, function (err, content) {
  if (err) throw err;
  console.log(content);
});

// Synchronous highlighting with highlight.js
marked.setOptions({
  highlight: function (code) {
    return require('highlight.js').highlightAuto(code).value;
  }
});

console.log(marked(markdownString));
```

highlight arguments

code

Type: string

The section of code to pass to the highlighter.

lang

Type: string

The programming language specified in the code block.

callback

Type: function

The callback function to call when using an async highlighter.

renderer

Type: object Default: `new Renderer()`

An object containing functions to render tokens to HTML.

Overriding renderer methods

The renderer option allows you to render tokens in a custom manner. Here is an example of overriding the default heading token rendering by adding an embedded anchor tag like on GitHub:

```

var marked = require('marked');
var renderer = new marked.Renderer();

renderer.heading = function (text, level) {
  var escapedText = text.toLowerCase().replace(/[^\w]+/g, '-');

  return '<h' + level + '><a name="' +
    escapedText +
    '" class="anchor" href="#' +
    escapedText +
    '"><span class="header-link"></span></a>' +
    text + '</h' + level + '>';
},

console.log(marked('# heading+', { renderer: renderer }));

```

This code will output the following HTML:

```

<h1>
  <a name="heading-" class="anchor" href="#heading-">
    <span class="header-link"></span>
  </a>
  heading+
</h1>

```

Block level renderer methods

- `code(string code, string language)`
- `blockquote(string quote)`
- `html(string html)`
- `heading(string text, number level)`
- `hr()`
- `list(string body, boolean ordered)`
- `listitem(string text)`
- `paragraph(string text)`
- `table(string header, string body)`
- `tablerow(string content)`
- `tablecell(string content, object flags)`

`flags` has the following properties:

```

{
  header: true || false,
  align: 'center' || 'left' || 'right'
}

```

Inline level renderer methods

- `strong(string text)`
- `em(string text)`
- `codespan(string code)`
- `br()`
- `del(string text)`
- `link(string href, string title, string text)`
- `image(string href, string title, string text)`

gfm

Type: boolean Default: true

Enable GitHub flavored markdown.

tables

Type: boolean Default: true

Enable GFM [tables](#). This option requires the `gfm` option to be true.

breaks

Type: boolean Default: false

Enable GFM [line breaks](#). This option requires the `gfm` option to be true.

pedantic

Type: boolean Default: false

Conform to obscure parts of `markdown.pl` as much as possible. Don't fix any of the original markdown bugs or poor behavior.

sanitize

Type: boolean Default: false

Sanitize the output. Ignore any HTML that has been input.

smartLists

Type: boolean Default: true

Use smarter list behavior than the original markdown. May eventually be default with the old behavior moved into `pedantic`.

smartypants

Type: boolean Default: false

Use "smart" typographic punctuation for things like quotes and dashes.

Access to lexer and parser

You also have direct access to the lexer and parser if you so desire.

```
var tokens = marked.lexer(text, options);
console.log(marked.parser(tokens));
```

```
var lexer = new marked.Lexer(options);
var tokens = lexer.lex(text);
console.log(tokens);
console.log(lexer.rules);
```

CLI

```
$ marked -o hello.html
hello world
^D
$ cat hello.html
<p>hello world</p>
```

Philosophy behind marked

The point of marked was to create a markdown compiler where it was possible to frequently parse huge chunks of markdown without having to worry about caching the compiled output somehow...or blocking for an unnecessarily long time.

marked is very concise and still implements all markdown features. It is also now fully compatible with the client-side.

marked more or less passes the official markdown test suite in its entirety. This is important because a surprising number of markdown compilers cannot pass more than a few tests. It was very difficult to get marked as compliant as it is. It could have cut corners in several areas for the sake of performance, but did not in order to be exactly what you expect in terms of a markdown rendering. In fact, this is why marked could be considered at a disadvantage in the benchmarks above.

Along with implementing every markdown feature, marked also implements [GFM features](#).

Benchmarks

node v0.8.x

```
$ node test --bench
marked completed in 3411ms.
marked (gfm) completed in 3727ms.
marked (pedantic) completed in 3201ms.
robotskirt completed in 808ms.
showdown (reuse converter) completed in 11954ms.
showdown (new converter) completed in 17774ms.
markdown-js completed in 17191ms.
```

Marked is now faster than Discount, which is written in C.

For those feeling skeptical: These benchmarks run the entire markdown test suite 1000 times. The test suite tests every feature. It doesn't cater to specific aspects.

Pro level

You also have direct access to the lexer and parser if you so desire.

```
var tokens = marked.lexer(text, options);
console.log(marked.parser(tokens));
```

```
var lexer = new marked.Lexer(options);
var tokens = lexer.lex(text);
console.log(tokens);
console.log(lexer.rules);
```

```
$ node
> require('marked').lexer('> i am using marked.')
[ { type: 'blockquote_start' },
  { type: 'paragraph',
    text: 'i am using marked.' },
  { type: 'blockquote_end' },
  links: {} ]
```

Running Tests & Contributing

If you want to submit a pull request, make sure your changes pass the test suite. If you're adding a new feature, be sure to add your own test.

The marked test suite is set up slightly strangely: `test/new` is for all tests that are not part of the original `markdown.pl` test suite (this is where your test should go if you make one). `test/original` is only for the original `markdown.pl` tests. `test/tests` houses both types of tests after they have been combined and moved/generated by running `node test --fix` or `marked --test --fix`.

In other words, if you have a test to add, add it to `test/new/` and then regenerate the tests with `node test --fix`. Commit the result. If your test uses a certain feature, for example, maybe it assumes GFM is *not* enabled, you can add `.nogfm` to the filename. So, `my-test.text` becomes `my-test.nogfm.text`. You can do this with any marked option. Say you want line breaks and smartypants enabled, your filename should be: `my-test.breaks.smartypants.text`.

To run the tests:

```
cd marked/  
node test
```

Contribution and License Agreement

If you contribute code to this project, you are implicitly allowing your code to be distributed under the MIT license. You are also implicitly verifying that all code is your original work. </legalese>

License

Copyright (c) 2011-2014, Christopher Jeffrey. (MIT License)

See LICENSE for more info.

