- Vitaly Puzrin github/puzrin

markdown-it is the result of the decision of the authors who contributed to 99% of the *Remarkable* code to move to a project with the same authorship but new leadership (Vitaly and Alex). It's not a fork.

References / Thanks

Big thanks to John MacFarlane for his work on the CommonMark spec and reference implementations. His work saved us a lot of time during this project's development.

Related Links:

- https://github.com/jgm/CommonMark - reference CommonMark implementations in C & JS, also contains latest spec & online demo.
- http://talk.commonmark.org - CommonMark forum, good place to collaborate developers' efforts.

Ports

- motion-markdown-it - Ruby/RubyMotion
- markdown-it-py- Python

markdown-it

CI passing

Markdown parser done right. Fast and easy to extend.

Live demo

Follows the **CommonMark spec** + adds syntax extensions & sugar (URL autolinking, typographer).

Configurable syntax! You can add new rules and even replace existing ones.

High speed.

Safe by default.

Community-written **plugins** and other packages on npm.

Table of content

- Benchmark
- markdown-it for enterprise
- Authors
- References / Thanks

Install

node.js:

```
npm install markdown-it
```

browser (CDN):

- jsDeliver CDN
- cdnjs.com CDN

Usage examples

See also:

- **API documentation** - for more info and examples.
- Development info - for plugins writers.

```
> current x 743 ops/sec ±0.84%
        (97 runs sampled)
> current-commonmark x 1,568 ops/sec
        ±0.84% (98 runs sampled)
> marked x 1,587 ops/sec ±4.31% (93
        runs sampled)
```

Note. CommonMark version runs with simplified link normalizers for more “honest” compare. Difference is ≈1.5×.

As you can see, `markdown-it` doesn’t pay with speed for its flexibility. Slowdown of “full” version caused by additional features not available in other implementations.

markdown-it for enterprise

Available as part of the Tidelift Subscription.

The maintainers of `markdown-it` and thousands of other packages are working with Tidelift to deliver commercial support and maintenance for the open source dependencies you use to build your applications. Save time, reduce risk, and improve code health, while paying the maintainers of the exact dependencies you use. Learn more.

Authors

- Alex Kocharin github/rlidwka

```
  .enable(['link'])
  .enable('image');

// Enable everything
const md = markdownit({
  html: true,
  linkify: true,
  typographer: true,
});
```

You can find all rules in sources:

- parser_core.mjs
- parser_block.mjs
- parser_inline.mjs

Benchmark

Here is the result of readme parse at MB Pro Retina 2013 (2.4 GHz):

```
npm run benchmark-deps
benchmark/benchmark.mjs readme

Selected samples: (1 of 28)
 > README

Sample: README.md (7774 bytes)
 > commonmark-reference x 1,222 ops/sec
      ±0.96% (97 runs sampled)
```

Simple

```
// node.js
// can use `require('markdown-it')` for
        CJS
import markdownit from 'markdown-it'
const md = markdownit()
const result = md.render('# markdown-it
        rulezz!');

// browser with UMD build, added to
        "window" on script load
// Note, there is no dash in
        "markdownit".
const md = window.markdownit();
const result = md.render('# markdown-it
        rulezz!');
```

Single line rendering, without paragraph wrap:

```
import markdownit from 'markdown-it'
const md = markdownit()
const result =
        md.renderInline('__markdown-it__
        rulezz!');
```

Init with presets and options

(*) presets define combinations of active rules and options. Can be "commonmark", "zero" or "default" (if skipped). See API docs for more details.

```
import markdownit from 'markdown-it'

// commonmark mode
const md = markdownit('commonmark')

// default mode
const md = markdownit()

// enable everything
const md = markdownit({
  html: true,
  linkify: true,
  typographer: true
})

// full options list (defaults)
const md = markdownit({
  // Enable HTML tags in source
  html:         false,

  // Use '/' to close single tags (<br /
        >).
  // This is only for full CommonMark
        compatibility.
  xhtmlOut:     false,
```

Syntax extensions

Embedded (enabled by default):

- Tables (GFM)
- Strikethrough (GFM)

Via plugins:

- subscript
- superscript
- footnote
- definition list
- abbreviation
- emoji
- custom container
- insert
- mark
- … and others

Manage rules

By default all rules are enabled, but can be restricted by options. On plugin load all its rules are enabled automatically.

```
import markdownit from 'markdown-it'

// Activate/deactivate rules, with
        currying
const md = markdownit()
  .disable(['link', 'image'])
```

```
      } catch (__) {}
    }

    return '<pre><code class="hljs">' +
        md.utils.escapeHtml(str) + '</
        code></pre>';
  }
});
```

Linkify

`linkify: true` uses linkify-it. To configure linkify-it, access the linkify instance through `md.linkify`:

```
md.linkify.set({ fuzzyEmail:
        false });  // disables
        converting email to link
```

API

API documentation

If you are going to write plugins, please take a look at Development info.

```
// Convert '\n' in paragraphs into
      <br>
breaks:        false,

// CSS language prefix for fenced
      blocks. Can be
// useful for external highlighters.
langPrefix:   'language-',

// Autoconvert URL-like text to links
linkify:       false,

// Enable some language-neutral
      replacement + quotes
      beautification
// For the full list of replacements,
      see https://github.com/markdown-
      it/markdown-it/blob/master/lib/
      rules_core/replacements.mjs
typographer:  false,

// Double + single quotes replacement
      pairs, when typographer enabled,
// and smartquotes on. Could be either
      a String or an Array.
//
// For example, you can use '«»„“' for
      Russian, '„“‚‘' for German,
// and ['«\xA0', '\xA0»', '‹\xA0',
      '\xA0›'] for French (including
      nbsp).
quotes: '“”‘’',
  // Highlighter function. Should return
        escaped HTML,
  // or '' if the source string is not
        changed and should be escaped
        externally.
  // If result starts with <pre...
        internal wrapper is skipped.
  highlight: function (/*str, lang*/) {
        return ''; }
});
```

Plugins load

```
import markdownit from 'markdown-it'

const md = markdownit
  .use(plugin1)
  .use(plugin2, opts, ...)
  .use(plugin3);
```

Syntax highlighting

Apply syntax highlighting to fenced code blocks with the `highlight` option:

```
import markdownit from 'markdown-it'
import hljs from 'highlight.js' //
        https://highlightjs.org

// Actual default values
```

```
const md = markdownit({
  highlight: function (str, lang) {
    if (lang && hljs.getLanguage(lang))
        {
      try {
        return hljs.highlight(str, {
        language: lang }).value;
      } catch (__) {}
    }

    return ''; // use external default
        escaping
  }
});
```

Or with full wrapper override (if you need assign class to `<pre>` or `<code>`):

```
import markdownit from 'markdown-it'
import hljs from 'highlight.js' //
        https://highlightjs.org

// Actual default values
const md = markdownit({
  highlight: function (str, lang) {
    if (lang && hljs.getLanguage(lang))
        {
      try {
        return '<pre><code
        class="hljs">' +
              hljs.highlight(str, {
        language: lang, ignoreIllegals:
        true }).value +
              '</code></pre>';
```