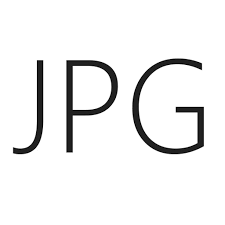
**Table of Contents**

# oldmark

[Online Image](https://pkg.go.dev/github.com/yuin/goldmark)test[Online Image](https://github.com/yuin/goldmark/actions?query=workflow:test)[Online Image](https://coveralls.io/github/yuin/goldmark)

[Online Image](https://goreportcard.com/report/github.com/yuin/goldmark)

A Markdown parser written in Go. Easy to extend, standards-compliant, well-structured.

goldmark is compliant with CommonMark 0.31.2.

* [goldmark playground](https://yuin.github.io/goldmark/playground/) : Try goldmark online. This playground is built with WASM(5-10MB).

# Motivation

This is a test

I needed a markdown parser for go that satisfies the following requirements:

* easy to extend.
  + markdown is poor in document expressions compared to other light markup languages such as restructuredtext.
  + we have extensions to the markdown syntax, e.g. php markdown extra, github flavored markdown.
* standards-compliant.
  + markdown has many dialects.
  + github-flavored markdown is widely used and is based upon commonmark, effectively mooting the question of whether or not commonmark is an ideal specification.
    - commonmark is complicated and hard to implement.
* well-structured.
  + ast-based; preserves source position of nodes.
* written in pure go.

[golang-commonmark](https://gitlab.com/golang-commonmark/markdown) may be a good choice, but it seems to be a copy of [markdown-it](https://github.com/markdown-it).

[blackfriday.v2](https://github.com/russross/blackfriday/tree/v2) is a fast and widely-used implementation, but is not commonmark-compliant and cannot be extended from outside of the package, since its ast uses structs instead of interfaces.

furthermore, its behavior differs from other implementations in some cases, especially regarding lists: [deep nested lists don't output correctly #329](https://github.com/russross/blackfriday/issues/329), [list block cannot have a second line #244](https://github.com/russross/blackfriday/issues/244), etc.

this behavior sometimes causes problems. if you migrate your markdown text from github to blackfriday-based wikis, many lists will immediately be broken.

as mentioned above, commonmark is complicated and hard to implement, so markdown parsers based on commonmark are few and far between.

## features

* **standards-compliant.** goldmark is fully compliant with the latest [commonmark](https://commonmark.org/) specification.
* **extensible.** do you want to add a @username mention syntax to markdown?you can easily do so in goldmark. you can add your ast nodes,parsers for block-level elements, parsers for inline-level elements,transformers for paragraphs, transformers for the whole ast structure, andrenderers.
* **performance.** goldmark's performance is on par with that of cmark,the commonmark reference implementation written in c.
* **robust.** goldmark is tested with go test --fuzz.
* **built-in extensions.** goldmark ships with common extensions like tables, strikethrough,task lists, and definition lists.
* **depends only on standard libraries.**

## installation

$ go get github.com/yuin/goldmark

## usage

import packages:

import (
  
 "bytes"
  
 "github.com/yuin/goldmark"
  
)

convert markdown documents with the commonmark-compliant mode:

var buf bytes.buffer
  
if err := goldmark.convert(source, &buf); err != nil {
  
 panic(err)
  
}

## with options

var buf bytes.buffer
  
if err := goldmark.convert(source, &buf, parser.withcontext(ctx)); err != nil {
  
 panic(err)
  
}

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| parser.withcontext | a parser.context | context for the parsing phase. |

## context options

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| parser.withids | a parser.ids | ids allows you to change logics that are related to element id(ex: auto heading id generation). |

## custom parser and renderer

import (
  
 "bytes"
  
 "github.com/yuin/goldmark"
  
 "github.com/yuin/goldmark/extension"
  
 "github.com/yuin/goldmark/parser"
  
 "github.com/yuin/goldmark/renderer/html"
  
)
  
  
md := goldmark.new(
  
 goldmark.withextensions(extension.gfm),
  
 goldmark.withparseroptions(
  
 parser.withautoheadingid(),
  
 ),
  
 goldmark.withrendereroptions(
  
 html.withhardwraps(),
  
 html.withxhtml(),
  
 ),
  
 )
  
var buf bytes.buffer
  
if err := md.convert(source, &buf); err != nil {
  
 panic(err)
  
}

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| goldmark.withparser | parser.parser | this option must be passed before goldmark.withparseroptions and goldmark.withextensions |
| goldmark.withrenderer | renderer.renderer | this option must be passed before goldmark.withrendereroptions and goldmark.withextensions |
| goldmark.withparseroptions | ...parser.option |  |
| goldmark.withrendereroptions | ...renderer.option |  |
| goldmark.withextensions | ...goldmark.extender |  |

## parser and renderer options

### parser options

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| parser.withblockparsers | a util.prioritizedslice whose elements are parser.blockparser | parsers for parsing block level elements. |
| parser.withinlineparsers | a util.prioritizedslice whose elements are parser.inlineparser | parsers for parsing inline level elements. |
| parser.withparagraphtransformers | a util.prioritizedslice whose elements are parser.paragraphtransformer | transformers for transforming paragraph nodes. |
| parser.withasttransformers | a util.prioritizedslice whose elements are parser.asttransformer | transformers for transforming an ast. |
| parser.withautoheadingid | - | enables auto heading ids. |
| parser.withattribute | - | enables custom attributes. currently only headings supports attributes. |

### html renderer options

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| html.withwriter | html.writer | html.writer for writing contents to an io.writer. |
| html.withhardwraps | - | render newlines as <br>. |
| html.withxhtml | - | render as xhtml. |
| html.withunsafe | - | by default, goldmark does not render raw html or potentially dangerous links. with this option, goldmark renders such content as written. |

### built-in extensions

* extension.table
  + [github flavored markdown: tables](https://github.github.com/gfm/#tables-extension-)
* extension.strikethrough
  + [github flavored markdown: strikethrough](https://github.github.com/gfm/#strikethrough-extension-)
* extension.linkify
  + [github flavored markdown: autolinks](https://github.github.com/gfm/#autolinks-extension-)
* extension.tasklist
  + [github flavored markdown: task list items](https://github.github.com/gfm/#task-list-items-extension-)
* extension.gfm
  + this extension enables table, strikethrough, linkify and tasklist.
  + this extension does not filter tags defined in [6.11: disallowed raw html (extension)](https://github.github.com/gfm/#disallowed-raw-html-extension-).if you need to filter html tags, see [security](#security).
  + if you need to parse github emojis, you can use [goldmark-emoji](https://github.com/yuin/goldmark-emoji) extension.
* extension.definitionlist
  + [php markdown extra: definition lists](https://michelf.ca/projects/php-markdown/extra/#def-list)
* extension.footnote
  + [php markdown extra: footnotes](https://michelf.ca/projects/php-markdown/extra/#footnotes)
* extension.typographer
  + this extension substitutes punctuations with typographic entities like [smartypants](https://daringfireball.net/projects/smartypants/).
* extension.cjk
  + this extension is a shortcut for cjk related functionalities.

### attributes

the parser.withattribute option allows you to define attributes on some elements.

currently only headings support attributes.

**attributes are being discussed in the**[commonmark forum](https://talk.commonmark.org/t/consistent-attribute-syntax/272).this syntax may possibly change in the future.

#### headings

## heading ## {#id .classname attrname=attrvalue class="class1 class2"}
  
  
## heading {#id .classname attrname=attrvalue class="class1 class2"}

heading {#id .classname attrname=attrvalue}
  
============

### table extension

the table extension implements [table(extension)](https://github.github.com/gfm/#tables-extension-), asdefined in [github flavored markdown spec](https://github.github.com/gfm/).

specs are defined for xhtml, so specs use some deprecated attributes for html5.

you can override alignment rendering method via options.

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| extension.withtablecellalignmethod | extension.tablecellalignmethod | option indicates how are table cells aligned. |

### typographer extension

the typographer extension translates plain ascii punctuation characters into typographic-punctuation html entities.

default substitutions are:

|  |  |
| --- | --- |
| punctuation | default entity |
| ' | &lsquo;, &rsquo; |
| " | &ldquo;, &rdquo; |
| -- | &ndash; |
| --- | &mdash; |
| ... | &hellip; |
| << | &laquo; |
| >> | &raquo; |

you can override the default substitutions via extensions.withtypographicsubstitutions:

markdown := goldmark.new(
  
 goldmark.withextensions(
  
 extension.newtypographer(
  
 extension.withtypographicsubstitutions(extension.typographicsubstitutions{
  
 extension.leftsinglequote: []byte("&sbquo;"),
  
 extension.rightsinglequote: nil, // nil disables a substitution
  
 }),
  
 ),
  
 ),
  
)

### linkify extension

the linkify extension implements [autolinks(extension)](https://github.github.com/gfm/#autolinks-extension-), asdefined in [github flavored markdown spec](https://github.github.com/gfm/).

since the spec does not define details about urls, there are numerous ambiguous cases.

you can override autolinking patterns via options.

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| extension.withlinkifyallowedprotocols | [][]byte | []string | list of allowed protocols such as []string{ "http:" } |
| extension.withlinkifyurlregexp | \*regexp.regexp | regexp that defines urls, including protocols |
| extension.withlinkifywwwregexp | \*regexp.regexp | regexp that defines url starting with www.. this pattern corresponds to [the extended www autolink](https://github.github.com/gfm/#extended-www-autolink) |
| extension.withlinkifyemailregexp | \*regexp.regexp | regexp that defines email addresses` |

example, using [xurls](https://github.com/mvdan/xurls):

import "mvdan.cc/xurls/v2"
  
  
markdown := goldmark.new(
  
 goldmark.withrendereroptions(
  
 html.withxhtml(),
  
 html.withunsafe(),
  
 ),
  
 goldmark.withextensions(
  
 extension.newlinkify(
  
 extension.withlinkifyallowedprotocols([]string{
  
 "http:",
  
 "https:",
  
 }),
  
 extension.withlinkifyurlregexp(
  
 xurls.strict(),
  
 ),
  
 ),
  
 ),
  
)

### footnotes extension

the footnote extension implements [php markdown extra: footnotes](https://michelf.ca/projects/php-markdown/extra/#footnotes).

this extension has some options:

|  |  |  |
| --- | --- | --- |
| functional option | type | description |
| extension.withfootnoteidprefix | []byte | string | a prefix for the id attributes. |
| extension.withfootnoteidprefixfunction | func(gast.node) []byte | a function that determines the id attribute for given node. |
| extension.withfootnotelinktitle | []byte | string | an optional title attribute for footnote links. |
| extension.withfootnotebacklinktitle | []byte | string | an optional title attribute for footnote backlinks. |
| extension.withfootnotelinkclass | []byte | string | a class for footnote links. this defaults to footnote-ref. |
| extension.withfootnotebacklinkclass | []byte | string | a class for footnote backlinks. this defaults to footnote-backref. |
| extension.withfootnotebacklinkhtml | []byte | string | a class for footnote backlinks. this defaults to &#x21a9;&#xfe0e;. |

some options can have special substitutions. occurrences of “^^” in the string will be replaced by the corresponding footnote number in the html output. occurrences of “%%” will be replaced by a number for the reference (footnotes can have multiple references).

extension.withfootnoteidprefix and extension.withfootnoteidprefixfunction are useful if you have multiple markdown documents displayed inside one html document to avoid footnote ids to clash each other.

extension.withfootnoteidprefix sets fixed id prefix, so you may write codes like the following:

for \_, path := range files {
  
 source := readall(path)
  
 prefix := getprefix(path)
  
  
 markdown := goldmark.new(
  
 goldmark.withextensions(
  
 newfootnote(
  
 withfootnoteidprefix(path),
  
 ),
  
 ),
  
 )
  
 var b bytes.buffer
  
 err := markdown.Convert(source, &b)
  
 if err != nil {
  
 t.Error(err.Error())
  
 }
  
}

extension.WithFootnoteIDPrefixFunction determines an id prefix by calling given function, so you may write codes like the following:

markdown := goldmark.New(
  
 goldmark.WithExtensions(
  
 NewFootnote(
  
 WithFootnoteIDPrefixFunction(func(n gast.Node) []byte {
  
 v, ok := n.OwnerDocument().Meta()["footnote-prefix"]
  
 if ok {
  
 return util.StringToReadOnlyBytes(v.(string))
  
 }
  
 return nil
  
 }),
  
 ),
  
 ),
  
)
  
  
for \_, path := range files {
  
 source := readAll(path)
  
 var b bytes.Buffer
  
  
 doc := markdown.Parser().Parse(text.NewReader(source))
  
 doc.Meta()["footnote-prefix"] = getPrefix(path)
  
 err := markdown.Renderer().Render(&b, source, doc)
  
}

You can use [goldmark-meta](https://github.com/yuin/goldmark-meta) to define a id prefix in the markdown document:

---
  
title: document title
  
slug: article1
  
footnote-prefix: article1
  
---
  
  
# My article

### CJK extension

CommonMark gives compatibilities a high priority and original markdown was designed by westerners. So CommonMark lacks considerations for languages like CJK.

This extension provides additional options for CJK users.

|  |  |  |
| --- | --- | --- |
| Functional option | Type | Description |
| extension.WithEastAsianLineBreaks | ...extension.EastAsianLineBreaksStyle | Soft line breaks are rendered as a newline. Some asian users will see it as an unnecessary space. With this option, soft line breaks between east asian wide characters will be ignored. This defaults to EastAsianLineBreaksStyleSimple. |
| extension.WithEscapedSpace | - | Without spaces around an emphasis started with east asian punctuations, it is not interpreted as an emphasis(as defined in CommonMark spec). With this option, you can avoid this inconvenient behavior by putting 'not rendered' spaces around an emphasis like 太郎は\ \*\*「こんにちわ」\*\*\ といった. |

#### Styles of Line Breaking

|  |  |
| --- | --- |
| Style | Description |
| EastAsianLineBreaksStyleSimple | Soft line breaks are ignored if both sides of the break are east asian wide character. This behavior is the same as east\_asian\_line\_breaks in Pandoc. |
| EastAsianLineBreaksCSS3Draft | This option implements CSS text level3 [Segment Break Transformation Rules](https://drafts.csswg.org/css-text-3/#line-break-transform) with [some enhancements](https://github.com/w3c/csswg-drafts/issues/5086). |

#### Example of EastAsianLineBreaksStyleSimple

Input Markdown:

私はプログラマーです。
  
東京の会社に勤めています。
  
GoでWebアプリケーションを開発しています。

Output:

<p>私はプログラマーです。東京の会社に勤めています。\nGoでWebアプリケーションを開発しています。</p>

#### Example of EastAsianLineBreaksCSS3Draft

Input Markdown:

私はプログラマーです。
  
東京の会社に勤めています。
  
GoでWebアプリケーションを開発しています。

Output:

<p>私はプログラマーです。東京の会社に勤めています。GoでWebアプリケーションを開発しています。</p>

## Security

By default, goldmark does not render raw HTML or potentially-dangerous URLs.If you need to gain more control over untrusted contents, it is recommended that youuse an HTML sanitizer such as [bluemonday](https://github.com/microcosm-cc/bluemonday).

## Benchmark

You can run this benchmark in the \_benchmark directory.

### against other golang libraries

blackfriday v2 seems to be the fastest, but as it is not CommonMark compliant, its performance cannot be directly compared to that of the CommonMark-compliant libraries.

goldmark, meanwhile, builds a clean, extensible AST structure, achieves full compliance withCommonMark, and consumes less memory, all while being reasonably fast.

* MBP 2019 13″(i5, 16GB), Go1.17

BenchmarkMarkdown/Blackfriday-v2-8 302 3743747 ns/op 3290445 B/op 20050 allocs/op
  
BenchmarkMarkdown/GoldMark-8 280 4200974 ns/op 2559738 B/op 13435 allocs/op
  
BenchmarkMarkdown/CommonMark-8 226 5283686 ns/op 2702490 B/op 20792 allocs/op
  
BenchmarkMarkdown/Lute-8 12 92652857 ns/op 10602649 B/op 40555 allocs/op
  
BenchmarkMarkdown/GoMarkdown-8 13 81380167 ns/op 2245002 B/op 22889 allocs/op

### against cmark (CommonMark reference implementation written in C)

* MBP 2019 13″(i5, 16GB), Go1.17

----------- cmark -----------
  
file: \_data.md
  
iteration: 50
  
average: 0.0044073057 sec
  
------- goldmark -------
  
file: \_data.md
  
iteration: 50
  
average: 0.0041611990 sec

As you can see, goldmark's performance is on par with cmark's.

## Extensions

### List of extensions

* [goldmark-meta](https://github.com/yuin/goldmark-meta): A YAML metadataextension for the goldmark Markdown parser.
* [goldmark-highlighting](https://github.com/yuin/goldmark-highlighting): A syntax-highlighting extensionfor the goldmark markdown parser.
* [goldmark-emoji](https://github.com/yuin/goldmark-emoji): An emojiextension for the goldmark Markdown parser.
* [goldmark-mathjax](https://github.com/litao91/goldmark-mathjax): Mathjax support for the goldmark markdown parser
* [goldmark-pdf](https://github.com/stephenafamo/goldmark-pdf): A PDF renderer that can be passed to goldmark.WithRenderer().
* [goldmark-hashtag](https://github.com/abhinav/goldmark-hashtag): Adds support for #hashtag-based tagging to goldmark.
* [goldmark-wikilink](https://github.com/abhinav/goldmark-wikilink): Adds support for [[wiki]]-style links to goldmark.
* [goldmark-anchor](https://github.com/abhinav/goldmark-anchor): Adds anchors (permalinks) next to all headers in a document.
* [goldmark-figure](https://github.com/mangoumbrella/goldmark-figure): Adds support for rendering paragraphs starting with an image to <figure> elements.
* [goldmark-frontmatter](https://github.com/abhinav/goldmark-frontmatter): Adds support for YAML, TOML, and custom front matter to documents.
* [goldmark-toc](https://github.com/abhinav/goldmark-toc): Adds support for generating tables-of-contents for goldmark documents.
* [goldmark-mermaid](https://github.com/abhinav/goldmark-mermaid): Adds support for rendering [Mermaid](https://mermaid-js.github.io/mermaid/) diagrams in goldmark documents.
* [goldmark-pikchr](https://github.com/jchenry/goldmark-pikchr): Adds support for rendering [Pikchr](https://pikchr.org/home/doc/trunk/homepage.md) diagrams in goldmark documents.
* [goldmark-embed](https://github.com/13rac1/goldmark-embed): Adds support for rendering embeds from YouTube links.
* [goldmark-latex](https://github.com/soypat/goldmark-latex): A $\LaTeX$ renderer that can be passed to goldmark.WithRenderer().
* [goldmark-fences](https://github.com/stefanfritsch/goldmark-fences): Support for pandoc-style [fenced divs](https://pandoc.org/MANUAL.html#divs-and-spans) in goldmark.
* [goldmark-d2](https://github.com/FurqanSoftware/goldmark-d2): Adds support for [D2](https://d2lang.com/) diagrams.
* [goldmark-katex](https://github.com/FurqanSoftware/goldmark-katex): Adds support for [KaTeX](https://katex.org/) math and equations.
* [goldmark-img64](https://github.com/tenkoh/goldmark-img64): Adds support for embedding images into the document as DataURL (base64 encoded).
* [goldmark-enclave](https://github.com/quail-ink/goldmark-enclave): Adds support for embedding youtube/bilibili video, X's [oembed tweet](https://publish.twitter.com/), [tradingview](https://www.tradingview.com/widget/)'s chart, [quail](https://quail.ink)'s widget into the document.
* [goldmark-wiki-table](https://github.com/movsb/goldmark-wiki-table): Adds support for embedding Wiki Tables.
* [goldmark-tgmd](https://github.com/Mad-Pixels/goldmark-tgmd): A Telegram markdown renderer that can be passed to goldmark.WithRenderer().

### Loading extensions at runtime

[goldmark-dynamic](https://github.com/yuin/goldmark-dynamic) allows you to write a goldmark extension in Lua and load it at runtime without re-compilation.

Please refer to [goldmark-dynamic](https://github.com/yuin/goldmark-dynamic) for details.

## goldmark internal(for extension developers)

### Overview

goldmark's Markdown processing is outlined in the diagram below.

<Markdown in []byte, parser.Context>
  
 |
  
 V
  
 +-------- parser.Parser ---------------------------
  
 | 1. Parse block elements into AST
  
 | 1. If a parsed block is a paragraph, apply
  
 | ast.ParagraphTransformer
  
 | 2. Traverse AST and parse blocks.
  
 | 1. Process delimiters(emphasis) at the end of
  
 | block parsing
  
 | 3. Apply parser.ASTTransformers to AST
  
 |
  
 V
  
 <ast.Node>
  
 |
  
 V
  
 +------- renderer.Renderer ------------------------
  
 | 1. Traverse AST and apply renderer.NodeRenderer
  
 | corespond to the node type
  
  
 |
  
 V
  
 <Output>

### Parsing

Markdown documents are read through text.Reader interface.

AST nodes do not have concrete text. AST nodes have segment information of the documents, represented by text.Segment .

text.Segment has 3 attributes: Start, End, Padding .

(TBC)

**TODO**

See extension directory for examples of extensions.

Summary:

* Define AST Node as a struct in which ast.BaseBlock or ast.BaseInline is embedded.
* Write a parser that implements parser.BlockParser or parser.InlineParser.
* Write a renderer that implements renderer.NodeRenderer.
* Define your goldmark extension that implements goldmark.Extender.

## Donation

BTC: 1NEDSyUmo4SMTDP83JJQSWi1MvQUGGNMZB

## License

MIT

## Author

Yusuke Inuzuka