Markdown

The Gitiles source browser automatically renders `*.md` Markdown files into HTML for simplified documentation.

Contents

Access controls

Access controls for documentation is identical to source code.

Documentation stored with source files shares the same permissions. Documentation stored in a separate Git repository can use different access controls. If Gerrit Code Review is being used, branch level read permissions can be used to grant or restrict access to any documentation branches.

READMEs

Files named `README.md` are automatically displayed below the file's directory listing. For the top level directory this mirrors the standard GitHub presentation.

> README.md files are meant to provide orientation for developers browsing the repository, especially first-time users.

We recommend that Git repositories have an up-to-date top-level `README.md` file.

Markdown syntax

Gitiles supports the core Markdown syntax described in Markdown Basics. Additional extensions are supported to more closely match GitHub Flavored Markdown and simplify documentation writing.

Paragraphs

Paragraphs are one or more lines of consecutive text, followed by one or more blank lines. Line breaks within a paragraph are ignored by the parser, allowing authors to line-wrap text at any comfortable column width.

```
Documentation writing can be fun and profitable
by helping users to learn and solve problems.

After documentation is written, it needs to be published on the
web where it can be easily accessed for reading.
```

Headings

Headings can be indicated by starting the line with one or more `#` marks. The number of `#` used determines the depth of the heading in the document outline. Headings 1 through 6 (`######`) are supported.

```
# A level one (H1) heading
## A level two (H2) heading
### A level three (H3) heading
```

Headings can also use the less popular two line `======` and `------` forms to create H1 and H2 level headers:

```
A first level header
====================

A second level header
---------------------
```

This form is discouraged as maintaining the length of the `===` or `---` lines to match the preceding line can be tedious work that is unnecessary with the `#` headers.

Lists

A bullet list:

```
* Fruit
    * Orange
    * Pear
* Cake
```

will render into HTML as:

- Fruit
    - Orange
    - Pear

- Cake

The second level items (above Orange, Pear) must be indented with more spaces than the item they are nested under. Above 2 spaces were used. Additionally, the entire list must be preceded and followed by a blank line.

A numbered list:

```
1. Fruit
     1. Orange
     2. Pear
5. Cake
```

will render into HTML as:

1. Fruit

    1. Orange
    2. Pear

2. Cake

List items will be renumbered sequentially by the browser, which is why `5` above displays as `2`. Even with this feature it is a good idea to maintain list item numbers in the source Markdown to simplify reading the source file.

Like bullet lists, numbered lists can be nested by using more leading space than the prior list item.

Paragraphs can be properly nested within lists by indenting with at least 2 leading spaces:

```
1. Fruit

  In botany, a fruit is the seed-bearing structure in flowering
  plants.

  In common language usage, "fruit" normally means the fleshy
  seed-associated structures of a plant that are sweet or sour, and
  edible in the raw state, such as apples, bananas, grapes, lemons,
  oranges, and strawberries.

2. Cake

  The cake is a lie.
```

Tables

Requires `markdown.tables` to be true (default).

Simple tables are supported with column alignment. The first line is the header row and subsequent lines are data rows:

```
| Food  | Calories | Tasty? |
|-------|---------:|:------:|
| Apple |    95    | Yes    |
| Pear  |   102    | Yes    |
| Hay   |   977    |        |
```

will render as:

Food	Calories	Tasty?
Apple	95	Yes
Pear	102	Yes

Food	Calories	Tasty?
Hay	977	

Placing `:` in the separator line indicates how the column should be aligned. A colon on the left side is a **left-aligned** column; a colon on the right-most side is **right-aligned**; a colon on both sides is **center-aligned**. If no alignment is specified, the column is aligned with the default for HTML `<td>` tags (left-aligned by default unless overridden by css).

Empty table cells are indicated by whitespace between the column dividers (`| |`) while multiple column cells omit the whitespace.

Emphasis

Emphasize paragraph text with *italic* and **bold** styles. Either `_` or `*` can be used for italic (1 character) and bold text (2 characters). This allows styles to be mixed within a statement:

```
Emphasize paragraph text with *italic* and **bold** text.

**It is _strongly_ encouraged to review documentation for typos.**
```

It is *strongly* encouraged to review documentation for typos.

Emphasis within_words_is_ignored which helps write technical documentation. Literal *bold* can be forced by prefixing the opening `*` with \ such as `\*bold*`.

Strikethrough

Requires `markdown.strikethrough` to be true (default).

Text can be ~~struck out~~ within a paragraph:

```
Text can be ~~struck out~~ within a paragraph.
```

Note two tildes are required (`~~`) on either side of the struck out section of text.

Blockquotes

Blockquoted text can be used to stand off text obtained from another source:

```
Sir Winston Churchill once said:

> A lie gets halfway around the world before the truth has a
> chance to get its pants on.
```

renders as:

Sir Winston Churchill once said:

> A lie gets halfway around the world before the truth has a chance to get its pants on.

Code (inline)

Use `backticks` to markup inline code within a paragraph:

```
Use `backticks` to markup inline code within a paragraph.
```

Code (blocks)

Create a fenced code block using three backticks before and after a block of code, preceeded and followed by blank lines:

~~~
This is a simple hello world program in C:

``` c
#include <stdio.h>

int main() {
  printf("Hello, World.\n");
  return 0;
}
```

To compile it use `gcc hello.c`.
~~~

Text within a fenced code block is taken verbatim and is not processed for Markdown markup.

Syntax highlighting can optionally be enabled for common languages by adding the language name in lowercase on the opening line. Supported languages include:

Scripting

- bash, sh
- lua
- perl
- python, py
- ruby
- tcl

Web

- css
- dart
- html
- javascript, js
- json

Compiled

- basic, vb
- c
- cpp (C++)
- go
- java
- pascal
- scala

Markup

- tex, latex
- wiki
- xml
- xquery
- xsl
- yaml

Other

- clj (Clojure)
- erlang
- hs (Haskell)
- lisp
- llvm
- matlab
- ml (OCaml, SML, F#)
- r
- rd
- rust
- sql
- vhdl

Horizontal rules

If `markdown.ghthematicbreak` is true, a horizontal rule can be inserted using GitHub style `--` surrounded by blank lines. Alternatively repeating `-` or `*` and space on a line will also create a horizontal rule:

```
---

- - - -

* * * *
```

Links

Wrap text in `[brackets]` and the link destination in parens `(http://...)` such as:

```
Visit [this site](http://example.com/) for more examples.
```

Links can also use references to obtain URLs from elsewhere in the same document. This style can be useful if the same URL needs to be mentioned multiple times, is too long to cleanly place within text, or the link is within a table cell:

```
Search for [markdown style][1] examples.

[1]: https://www.google.com/?gws_rd=ssl#q=markdown+style+guide
```

References can be simplified if the text alone is sufficient:

```
Visit [Google] to search the web.

[Google]: https://www.google.com/
```

Automatic hyperlinking can be used where the link text should obviously also be the URL:

```
Use https://www.google.com/ to search the web.
```

Well formed URLs beginning with `https://`, `http://`, and `mailto:` are used as written for the link's destination. Malformed URLs may be replaced with `about:invalid#zSoyz` to prevent browser evaluation of dangerous content.

HTML escaping of URL characters such as `&` is handled internally by the parser/formatter. Documentation writers should insert the URL literally and allow the parser and formatter to make it HTML safe.

Relative links such as `../src/api.md` are resolved relative to the current markdown's file path within the Git repository. Absolute links such as `/src/api.md` are resolved relative to the top of the enclosing Git repository, but within the same branch or Git commit. Links may point to any file in the repository. A link to a `*.md` file will present the rendered markdown, while a link to a source file will display the syntax highlighted source.

Named anchors

Explicit anchors can be inserted anywhere in the document using `<a name="tag"></a>`, or `{#tag}` if `markdown.namedanchor` is true.

Implicit anchors are automatically created for each heading. For example `## Section 1` will have the anchor `Section-1`.

Anchor generation

- letters and digits, after removing accents (á → a)
- spaces are replaced with hyphens (`-`)
- other characters are replaced with underscores (`_`)
- runs of hyphens and underscores are collapsed

If a document contains the same named subsection under different parents the parent anchor will prefix the subsections to disambiguate. For example the following document will have anchors `Running-Format` and `Coding-Format` and `Libraries` as that subsection is unique:

```
## Running
### Format
```

```
## Coding
### Format
### Libraries
```

When placed in a section header the explicit anchor will override the automatic anchor. The following are identical and associate the anchor `live-examples` with the section header instead of the automaticly generated name `Examples`.

```
## Examples {#live-examples}
## Examples <a name="live-examples"></a>
```

Images

Similar to links but begin with `!` to denote an image reference:

```
![diffy the kung fu review cuckoo](http://commondatastorage.googleapis.com/gerrit-static/diffy-w20
```

For images the text in brackets will be included as the alt text for screen readers.

Well formed image URLs beginning with `https://` and `http://` will be used as written for the `<img src="...">` attribute. Malformed URLs will be replaced with a broken `data:` reference to prevent browsers from trying to load a bad destination.

Relative and absolute links to image files within the Git repository (such as `../images/banner.png`) are resolved during rendering by inserting the base64 encoding of the image using a `data:` URI. Only PNG (`*.png`), JPEG (`*.jpg` or `*.jpeg`), GIF (`*.gif`) and WebP (`*.webp`) image formats are supported when referenced from the Git repository.

Unsupported extensions or files larger than image size limit (default 256K) will display a broken image.

> *Inline image caching*
>
> Gitiles allows browsers to locally cache rendered markdown pages. Cache invalidation is triggered by the markdown file being modified and having a different SHA-1 in Git. Inlined images may need a documentation file update to be refreshed when viewed through unstable URLs like `/docs/+/master/index.md`.

HTML

Most HTML tags are not supported. HTML will be dropped on the floor by the parser with no warnings, and no output from that section of the document.

If `markdown.safehtml` is true there are small exceptions for `<br>`, `<hr>`, `<a name>` and `<iframe>`elements, see named anchor and HTML IFrame.

Markdown extensions

Gitiles includes additional extensions to the Markdown language that make documentation writing for the web easier without using raw HTML.

Table of contents

Requires `markdown.toc` to be true.

Place `[TOC]` surrounded by blank lines to insert a generated table of contents extracted from the H1, H2, and H3 headers used within the document:

```
# Title

[TOC]

## Section 1
Blah blah...

## Section 2
Go on...
```

H1 headers are omitted from the table of contents if there is only one level one header present. This allows H1 to be used as the document title without creating an unnecessary entry in the table of contents.

Anchors are automatically extracted from the headers, see named anchors.

Notification, aside, promotion blocks

Requires `markdown.blocknote` to be true.

Similar to fenced code blocks these blocks start and end with `***`, are surrounded by blank lines, and include the type of block on the opening line.

Note

```
*** note
**Warning:** watch out for nested formatting.
***
```

Warning: watch out for nested formatting.

Aside

```
*** aside
An aside can stand off less interesting text.
***
```

An aside can stand off less interesting text.

Promo

```
*** promo
Promotions can raise awareness of an important concept.
***
```

Promotions can raise awareness of an important concept.

Column layout

Requires `markdown.multicolumn` to be true.

Gitiles markdown includes support for up to 12 columns of text across the width of the page. By default space is divided equally between the columns.

Columns

can save space.

Prettify

the page layout.

Can be

trendy.

A column layout is denoted by a block starting and ending with the sequence `|||---|||`. Within the layout a new column is started for each header or note/promo/aside block and all text and formatting flow into that column until a new column is started.

```
|||---|||
#### Columns

can save space.

#### Prettify

the page layout.

*** promo
#### Can be

trendy.
***
|||---|||
```

Column spans can be specified on the first line as a comma separated list. In the example below the first column is 4 wide or 4/12ths of the page width, the second is 2 wide (or 2/12ths) and the final column is 6 wide (6/12ths or 50%) of the page.

```
|||---||| 4,2,6
```

An empty column can be inserted by prefixing its width with `:`, for example shifting content onto the right by padding 6 columns on the left:

```
|||---||| :6,3
#### Right
|||---|||
```

renders as:

Right

HTML IFrame

Requires `markdown.safehtml` to be true (default).

Although HTML is stripped the parser has special support for a limited subset of `<iframe>` elements:

```
<iframe src="https://example.com/embed" height="200px" width="400px"></iframe>
```

The parser allows whitespace including newlines between attributes, but strictly limits the supported attribute set to:

src : An `https://` or `http://` URL of the page to embed inside of an iframe at this position in the document. Malformed URLs will cause the iframe to be silently dropped. *(required)*

height : CSS pixel height such as `250px` defining the amount of vertical space to give to the embedded content. Only `px` units are supported; a malformed dimension will drop the iframe. *(required)*

width : CSS pixel width such as `250px` or a precentage such as `80%` defining the amount of horizontal space to give to the embedded content. Only `px` units or `%` are supported; a malformed dimension will drop the iframe. *(required)*

frameborder : By default a border is drawn around the iframe by the browser. The border can be hidden by adding `frameborder="0"` to the iframe tag. *(optional)*

Embedded source URLs must also be whitelisted by the Gitiles `markdown.allowiframe` configuration variable.

Site layout

Gitiles includes additional support to create functional documentation sites served directly from Git repositories.

Navigation bar

A top level navigation bar is automatically included on all pages if there is a `navbar.md` file present in the top of the repository. This file should be a simple bulleted list of links to include in the navigation bar.

```
* [Home](/index.md)
* [Markdown](/docs/markdown.md)
* [Configuration](/docs/configuration.md)
```

Links are resolved relative to the current page, not `navbar.md`. Links to other files within the repository should use absolute paths to ensure they are resolved correctly from any Markdown file within the repository.

Site title

A site-wide banner is displayed on all Markdown pages if `navbar.md` includes an H1 header. The text of the header is displayed on the left side of the banner.

```
# Gitiles

* [Home](/index.md)
```

Site logo

An optional logo image is displayed in the banner to the left of the site title if a `[logo]` reference exists in `navbar.md`. This image should be no taller than 45 px.

```
# Gitiles

[logo]: /images/site_logo.png
```

See images above for acceptable URLs and how repository relative paths are handled by inline data URIs.

Home link

Both the site logo (if present) and site title are wrapped in a link if the `[home]` reference is declared in `navbar.md`. This is typically also used in the outline for the navigation bar:

```
# Gitiles

* [Home][home]
* [Markdown](/docs/markdown.md)

[home]: /index.md
```

Page title

Titles for pages are extracted from the first H1 heading appearing in the document. This is traditionally placed on the first line of the markdown file, e.g.:

```
# Markdown
```

The title is included in the HTML `<title>` element and also in the right side of the banner if `navbar.md` defines a site title.

Configuration

The `gitiles.config` file supporting the site contains several configuration options that impact markdown rendering. Refer to the configuration documentation for details.