Markdown style guide

Much of what makes Markdown great is the ability to write plain text, and get great formatted output as a result. To keep the slate clean for the next author, your Markdown should be simple and consistent with the whole corpus wherever possible.

We seek to balance three goals:

1. *Source text is readable and portable.*
2. *Markdown files are maintainable over time and across teams.*
3. *The syntax is simple and easy to remember.*

Contents:

Document layout

In general, most documents benefit from some variation of the following layout:

```
# Document Title

Short introduction.

[TOC]

## Topic
```

```
Content.

## See also

* https://link-to-more-info
```

1. `# Document Title`: The first heading should be a level one heading, and should ideally be the same or nearly the same as the filename. The first level one heading is used as the page `<title>`.
2. `author`: *Optional.* If you'd like to claim ownership of the document or if you are very proud of it, add yourself under the title. However, revision history generally suffices.
3. `Short introduction.` 1-3 sentences providing a high-level overview of the topic. Imagine yourself as a complete newbie, who landed on your "Extending Foo" doc and needs to know the most basic assumptions you take for granted. "What is Foo? Why would I extend it?"
4. `[TOC]`: if you use hosting that supports table of contents, such as Gitiles, put `[TOC]` after the short introduction. See `[TOC]` documentation.
5. `## Topic`: The rest of your headings should start from level 2.
6. `## See also`: Put miscellaneous links at the bottom for the user who wants to know more or didn't find what she needed.

Character line limit

Obey projects' character line limit wherever possible. Long URLs and tables are the usual suspects when breaking the rule. (Headings also can't be wrapped, but we encourage keeping them short). Otherwise, wrap your text:

```
Lorem ipsum dolor sit amet, nec eius volumus patrioque cu, nec et commodo
hendrerit, id nobis saperet fuisset ius.

*   Malorum moderatius vim eu. In vix dico persecuti. Te nam saperet percipitur
    interesset. See the [foo docs](https://gerrit.googlesource.com/gitiles/+/master/Document
```

Often, inserting a newline before a long link preserves readability while minimizing the overflow:

```
Lorem ipsum dolor sit amet. See the
[foo docs](https://gerrit.googlesource.com/gitiles/+/master/Documentation/markdown.md)
for details.
```

Trailing whitespace

Don't use trailing whitespace, use a trailing backslash.

The CommonMark spec decrees that two spaces at the end of a line should insert a `<br />` tag. However, many directories have a trailing whitespace presubmit check in place, and many IDEs will clean it up anyway.

Best practice is to avoid the need for a `<br />` altogether. Markdown creates paragraph tags for you simply with newlines: get used to that.

Headings

ATX-style headings

```
## Heading 2
```

Headings with `=` or `-` underlines can be annoying to maintain and don't fit with the rest of the heading syntax. The user has to ask: Does `---` mean H1 or H2?

```
Heading - do you remember what level? DO NOT DO THIS.
---------
```

Add spacing to headings

Prefer spacing after `#` and newlines before and after:

```
...text before.

# Heading 1

Text after...
```

Lack of spacing makes it a little harder to read in source:

```
...text before.

#Heading 1
Text after... DO NOT DO THIS.
```

Lists

Use lazy numbering for long lists

Markdown is smart enough to let the resulting HTML render your numbered lists correctly. For longer lists that may change, especially long nested lists, use "lazy" numbering:

```
1.  Foo.
1.  Bar.
    1.  Foofoo.
    1.  Barbar.
1.  Baz.
```

However, if the list is small and you don't anticipate changing it, prefer fully numbered lists, because it's nicer to read in source:

```
1.  Foo.
2.  Bar.
3.  Baz.
```

Nested list spacing

When nesting lists, use a 4 space indent for both numbered and bulleted lists:

```
1.  2 spaces after a numbered list.
    4 space indent for wrapped text.
2.  2 spaces again.

*   3 spaces after a bullet.
    4 space indent for wrapped text.
    1.  2 spaces after a numbered list.
        8 space indent for the wrapped text of a nested list.
    2.  Looks nice, don't it?
*   3 spaces after a bullet.
```

The following works, but it's very messy:

```
* One space,
with no indent for wrapped text.
     1. Irregular nesting... DO NOT DO THIS.
```

Even when there's no nesting, using the 4 space indent makes layout consistent for wrapped text:

```
*   Foo,
    wrapped.

1.  2 spaces
    and 4 space indenting.
2.  2 spaces again.
```

However, when lists are small, not nested, and a single line, one space can suffice for both kinds of lists:

```
* Foo
* Bar
* Baz.

1. Foo.
2. Bar.
```

Code

Inline

‘Backticks‘ designate `inline code`, and will render all wrapped content literally. Use them for short code quotations and field names:

```
You'll want to run `really_cool_script.sh arg`.

Pay attention to the `foo_bar_whammy` field in that table.
```

Use inline code when referring to file types in an abstract sense, rather than a specific file:

```
Be sure to update your `README.md`!
```

Backticks are the most common approach for “escaping” Markdown metacharacters; in most situations where escaping would be needed, code font just makes sense anyway.

Codeblocks

For code quotations longer than a single line, use a codeblock:

Declare the language

It is best practice to explicitly declare the language, so that neither the syntax highlighter nor the next editor must guess.

Indented codeblocks are sometimes cleaner

Four-space indenting is also interpreted as a codeblock. These can look cleaner and be easier to read in source, but there is no way to specify the language. We encourage their use when writing many short snippets:

```
You'll need to run:

    bazel run :thing -- --foo

And then:
    bazel run :another_thing -- --bar

And again:

    bazel run :yet_again -- --baz
```

Escape newlines

Because most commandline snippets are intended to be copied and pasted directly into a terminal, it's best practice to escape any newlines. Use a single backslash at the end of the line:

Nest codeblocks within lists

If you need a codeblock within a list, make sure to indent it so as to not break the list:

~~~
*   Bullet.

    ```c++
    int foo;
    ```

*   Next bullet.
~~~

You can also create a nested code block with 4 spaces. Simply indent 4 additional spaces from the list indentation:

```
*   Bullet.

        int foo;

*   Next bullet.
```

Links

Long links make source Markdown difficult to read and break the 80 character wrapping. **Wherever possible, shorten your links**.

Use informative Markdown link titles

Markdown link syntax allows you to set a link title, just as HTML does. Use it wisely.

Titling your links as “link” or “here” tells the reader precisely nothing when quickly scanning your doc and is a waste of space:

```
See the syntax guide for more info: [link](syntax_guide.md).
Or, check out the style guide [here](style_guide.md).
DO NOT DO THIS.
```

Instead, write the sentence naturally, then go back and wrap the most appropriate phrase with the link:

```
See the [syntax guide](syntax_guide.md) for more info.
Or, check out the [style guide](style_guide.md).
```

Images

Use images sparingly, and prefer simple screenshots. This guide is designed around the idea that plain text gets users down to the business of communication faster with less reader distraction and author procrastination. However, it's sometimes very helpful to show what you mean.

See image syntax.

Prefer lists to tables

Any tables in your Markdown should be small. Complex, large tables are difficult to read in source and most importantly, **a pain to modify later**.

```
Fruit | Attribute | Notes
--- | --- | --- | ---
Apple | [Juicy](https://example.com/SomeReallyReallyReallyReallyReallyReallyReallyReallyLong
Banana | [Convenient](https://example.com/SomeDifferentReallyReallyReallyReallyReallyReallyR

DO NOT DO THIS
```

Lists and subheadings usually suffice to present the same information in a slightly less compact, though much more edit-friendly way:

```
## Fruits

### Apple

* [Juicy](https://SomeReallyReallyReallyReallyReallyReallyReallyReallyReallyReallyReal
* Firm
* Sweet

Apples keep doctors away.

### Banana

* [Convenient](https://example.com/SomeDifferentReallyReallyReallyReallyReallyReallyReallyRe
* Soft
* Sweet

Contrary to popular belief, most apes prefer mangoes.
```

However, there are times when a small table is called for:

```
Transport | Favored by | Advantages
--- | --- | ---
Swallow | Coconuts | Otherwise unladen
Bicycle | Miss Gulch | Weatherproof
X-34 landspeeder | Whiny farmboys | Cheap since the X-38 came out
```

Strongly prefer Markdown to HTML

Please prefer standard Markdown syntax wherever possible and avoid HTML hacks. If you can't seem to accomplish what you want, reconsider whether you really need it. Except for big tables, Markdown meets almost all needs already.

Every bit of HTML or Javascript hacking reduces the readability and portability. This in turn limits the usefulness of integrations with other tools, which may either present the source as plain text or render it. See Philosophy.

Gitiles does not render HTML.