

**NAME**

Markdown – text formatting syntax

**DESCRIPTION**

Markdown is a text markup syntax for machine conversion to the more complex HTML or XHTML markup languages. It is intended to be easy to read and to write, with emphasis on readability. A Markdown-formatted document should be publishable as-is, in plain text, without the formatting distracting the reader.

The biggest source of inspiration for Markdown's syntax is the format of plain text email. The markup is comprised entirely of punctuation characters, chosen so as to look like what they mean. Asterisks around a word look like *\*emphasis\**. Markdown lists look like lists. Even blockquotes look like quoted passages of text, assuming the reader has used email.

**Block Elements****Paragraphs and Line Breaks**

A paragraph is one or more consecutive lines of text, separated by one or more blank lines. (A blank line is any line that looks like a blank line -- a line containing nothing but spaces or tabs is considered blank.) Normal paragraphs should not be indented with spaces or tabs.

Lines may be freely broken for readability; Markdown does not translate source line breaks to `<br />` tags. To request generation of `<br />` in the output, end a line with two or more spaces, then a newline.

**Headings**

Headings can be marked in two ways, called *setext* and *atx*.

Setext-style headings are “underlined” using equal signs (for first-level headings) and dashes (for second-level headings).

Atx-style headings use 1–6 hash characters at the start of the line, corresponding to HTML `<h^(1–6)>`. Optional closing hashes may follow the heading text.

**Blockquotes**

Lines beginning with `>` are output in blockquotes. Blockquotes can be nested by multiple levels of `>>`. Blockquotes can contain other Markdown elements, including headings, lists, and code blocks.

**Lists**

Markdown supports ordered (numbered) and unordered (bulleted) lists. List markers typically start at the left margin, but may be indented by up to three spaces. List markers must be followed by one or more spaces or a tab, then the list item text. A newline terminates each list item.

Unordered lists use asterisks, pluses, and hyphens interchangeably as list markers.

Ordered lists use integers followed by periods as list markers. The order of the integers is not interpreted, but the list should begin with 1.

If list items are separated by blank lines, Markdown will wrap each list item in `<p>` tags in the HTML output.

List items may consist of multiple paragraphs. Each subsequent paragraph within a list item must be indented by either 4 spaces or one tab. To put a blockquote within a list item, the blockquote's `>` marker needs to be indented. To put a code block within a list item, the code block needs to be indented *twice* -- 8 spaces or two tabs.

**Code Blocks**

To produce a code block, indent every line of the block by at least 4 spaces or 1 tab. A code block continues until it reaches a line that is not indented.

Rather than forming normal paragraphs, the lines of a code block are interpreted literally. Regular Markdown syntax is not processed within code blocks. Markdown wraps a code block in both `<pre>` and `<code>` tags. One level of indentation -- 4 spaces or 1 tab -- is removed from each line of the code block in the output.

**Horizontal Rules**

Produce a horizontal rule tag (`<hr />`) by placing three or more hyphens, asterisks, or

underscores on a line by themselves.

## Span Elements

### Links

Markdown supports two styles of links: *inline* and *reference*. In both styles, the link text is delimited by square brackets ([ ]). To create an inline link, use a set of regular parentheses immediately after the link text's closing square bracket. Inside the parentheses, put the link URL, along with an optional title for the link surrounded in double quotes. For example:

An [example](http://example.com/ "Title") inline link.

Reference-style links use a second set of square brackets, inside which you place a label of your choosing to identify the link:

An [example][id] reference-style link.

The label is then assigned a value on its own line, anywhere in the document:

[id]: http://example.com/ "Optional Title"

Link label names may consist of letters, numbers, spaces, and punctuation. Labels are not case sensitive. An empty label bracket set after a reference-style link implies the link label is equivalent to the link text. A URL value can then be assigned to the link by referencing the link text as the label name.

### Emphasis

Markdown treats asterisks (\*) and underscores (\_) as indicators of emphasis. Text surrounded with single asterisks or underscores will be wrapped with an HTML <em> tag. Double asterisks or underscores generate an HTML <strong> tag.

### Code

To indicate a span of code, wrap it with backtick quotes ('). Unlike a code block, a code span indicates code within a normal paragraph. To include a literal backtick character within a code span, you can use multiple backticks as the opening and closing delimiters:

‘‘There is a literal backtick (') here.’’

### Images

Markdown image syntax is intended to resemble that for links, allowing for two styles, once again *inline* and *reference*. The syntax is as for each respective style of link, described above, but prefixed with an exclamation mark character (!). Inline image syntax looks like this:

![Alt text](/path/to/img.jpg "Optional title")

That is: An exclamation mark; followed by a set of square brackets containing the 'alt' attribute text for the image; followed by a set of parentheses containing the URL or path to the image, and an optional 'title' attribute enclosed in double or single quotes.

Reference-style image syntax looks like this:

![Alt text][id]

Where *id* is a label used as for reference-style URL links, described above.

## Convenience

### Automatic Links

There is a shortcut style for creating "automatic" links for URLs and email addresses. Surround the URL or address with angle brackets.

### Backslash Escapes

Use backslash escapes to generate literal characters which would otherwise have special meaning in Markdown's formatting syntax.

### Inline HTML

For markup that is not covered by Markdown's syntax, simply use the HTML directly. The only restrictions are that block-level HTML elements -- <div>, <table>, <pre>, <p>, etc. -- must be separated from surrounding content by blank lines, and the start and end tags of the block should not be indented with tabs or spaces. Markdown formatting syntax is not processed within block-level HTML tags.

Span-level HTML tags -- e.g. `<span>`, `<cite>`, or `<del>` -- can be used anywhere in a Markdown paragraph, list item, or heading. It is permitted to use HTML tags instead of Markdown formatting; e.g. HTML `<a>` or `<img>` tags instead of Markdown's link or image syntax. Unlike block-level HTML tags, Markdown syntax *is* processed within the elements of span-level tags.

#### Automatic Special Character Escapes

To be displayed literally in a user agent, the characters `<` and `&` must appear as escaped entities in HTML source, e.g. `&lt;` and `&amp;`. Markdown allows natural use of these characters, taking care of the necessary escaping. The ampersand part of a directly-used HTML entity remains unchanged; otherwise it will be translated into `&amp;`. Inside code spans and blocks, angle brackets and ampersands are always encoded automatically. This makes it easy to use Markdown to write about HTML code.

#### Smarty Pants

The `markdown(1)` utility transforms a few plain text symbols into their typographically-fancier HTML entity equivalents. These are extensions to the standard Markdown syntax.

#### Punctuation

Input single- and double-quotes are transformed into "curly" quote entities in the output (e.g., `'text'` becomes `&lsquo;text&rsquo;`). Input double-dashes (`--`) and triple-dashes become en- and em-dashes, respectively, while a series of three dots (`. . .`) in the input becomes an ellipsis entity (`&hellip;`) in the HTML output.

#### Symbols

Three other transformations replace the common plain-text shorthands (`c`), (`℞`), and (`™`) from the input with their respective HTML entities. (As in (`c`) becoming `&copy;`, the Copyright symbol entity.)

#### Fractions

A small set of plain-text shorthands for fractions is recognized. `1/4` becomes `&frac14;`, for example. These fraction notations are replaced with their HTML entity equivalents: `1/4`, `1/2`, `3/4`. `1/4th` and `3/4ths` are replaced with their entity and the indicated ordinal suffix letters.

Like the basic Markdown syntax, none of the "Smarty Pants" extensions are processed inside code blocks or spans.

#### Discount Extensions

`Markdown(1)` recognizes some extensions to the Markdown format, many of them adopted or adapted from other Markdown interpreters or document formatting systems.

#### Pandoc Headers

If `markdown` was configured with `--enable-pandoc-header`, the markdown source can have a 3-line Pandoc header in the format of

```
% Title
% Author
% Date
```

whose data is available to the `mkd_doc_title`, `mkd_doc_author`, and `mkd_doc_date` (in `markdown(2)`) functions.

#### Embedded Stylesheets

Stylesheets may be defined and modified in a `<style>` block. A style block is parsed like any other block-level HTML; `<style>` starting on column 1, raw HTML (or, in this case, CSS) following it, and either ending with a `</style>` at the end of the line or at the beginning of a subsequent line.

Style blocks apply to the entire document regardless of where they are defined.

#### Image Dimensions

Image specification has been extended with an argument describing image dimensions: `=heightxwidth`. For an image 400 pixels high and 300 wide, the new syntax is:

```
![Alt text](/path/to/image.jpg =400x300 "Title")
```

### Pseudo-Protocols

Pseudo-protocols that may replace the common `http:` or `mailto:` have been added to the link syntax described above.

`abbr:` Text following is used as the `title` attribute of an `abbr` tag wrapping the link text. So `[LT](abbr:Link Text)` gives `<abbr title="Link Text">LT</abbr>`.

`id:` The link text is marked up and written to the output, wrapped with `<a id=text following>` and `</a>`.

`class:`

The link text is marked up and written to the output, wrapped with `<span class=text following>` and `</span>`.

`raw:` Text following is written to the output with no further processing. The link text is discarded.

### Alphabetic Lists

If `markdown` was configured with `--enable-alpha-list`,

```
a. this
b. is
c. an alphabetic
d. list
```

yields an HTML `ol` ordered list.

### Definition Lists

If configured with `--enable-dl-tag`, markup for definition lists is enabled. A definition list item is defined as

```
=term=
  definition
```

### Tables

Tables are specified with a pipe (`|`) and dash (`-`) marking. The markdown text

```
header0|header1
-----|-----
  textA|textB
  textC|textD
```

will produce an HTML table of two columns and three rows. A header row is designated by “underlining” with dashes. Declare a column’s alignment by affixing a colon (`:`) to the left or right end of the dashes underlining its header. In the output, this yields the corresponding value for the `align` attribute on each `td` cell in the column. A colon at both ends of a column’s header dashes indicates center alignment.

### Relaxed Emphasis

If configured with `--relaxed-emphasis`, the rules for emphasis are changed so that a single `_` will not count as an emphasis character in the middle of a word. This is useful for documenting some code where `_` appears frequently, and would normally require a backslash escape.

### SEE ALSO

*markdown(1)*, *markdown(2)*

<http://daringfireball.net/projects/markdown/syntax/>, “Markdown: Syntax”.

<http://daringfireball.net/projects/smartypants/>, “Smarty Pants”.

<http://michelf.com/projects/php-markdown/extra/#table>, “PHP Markdown Extra: Tables”.