This is the title: it contains a colon

# Pandoc’s markdown

Pandoc understands an extended and slightly revised version of John Gruber’s [markdown](http://daringfireball.net/projects/markdown/) syntax. This document explains the syntax, noting differences from standard markdown. Except where noted, these differences can be suppressed by using the markdown\_strict format instead of markdown. An extensions can be enabled by adding +EXTENSION to the format name and disabled by adding -EXTENSION. For example, markdown\_strict+footnotes is strict markdown with footnotes enabled, while markdown-footnotes-pipe\_tables is pandoc’s markdown without footnotes or pipe tables.

## Philosophy

Markdown is designed to be easy to write, and, even more importantly, easy to read:

A Markdown-formatted document should be publishable as-is, as plain text, without looking like it’s been marked up with tags or formatting instructions. – [John Gruber](http://daringfireball.net/projects/markdown/syntax#philosophy)

This principle has guided pandoc’s decisions in finding syntax for tables, footnotes, and other extensions.

There is, however, one respect in which pandoc’s aims are different from the original aims of markdown. Whereas markdown was originally designed with HTML generation in mind, pandoc is designed for multiple output formats. Thus, while pandoc allows the embedding of raw HTML, it discourages it, and provides other, non-HTMLish ways of representing important document elements like definition lists, tables, mathematics, and footnotes.

## Paragraphs

A paragraph is one or more lines of text followed by one or more blank line. Newlines are treated as spaces, so you can reflow your paragraphs as you like. If you need a hard line break, put two or more spaces at the end of a line.

**Extension: escaped\_line\_breaks**

A backslash followed by a newline is also a hard line break. Note: in multiline and grid table cells, this is the only way to create a hard line break, since trailing spaces in the cells are ignored.

## Headers

There are two kinds of headers, Setext and atx.

### Setext-style headers

A setext-style header is a line of text “underlined” with a row of = signs (for a level one header) or - signs (for a level two header):

# A level-one header

## A level-two header

The header text can contain inline formatting, such as emphasis (see [Inline formatting](#inline-formatting), below).

### Atx-style headers

An Atx-style header consists of one to six # signs and a line of text, optionally followed by any number of # signs. The number of # signs at the beginning of the line is the header level:

## A level-two header

### A level-three header

As with setext-style headers, the header text can contain formatting:

# A level-one header with a [link](/url) and *emphasis*

**Extension: blank\_before\_header**

Standard markdown syntax does not require a blank line before a header. Pandoc does require this (except, of course, at the beginning of the document). The reason for the requirement is that it is all too easy for a # to end up at the beginning of a line by accident (perhaps through line wrapping). Consider, for example:

I like several of their flavors of ice cream: #22, for example, and #5.

### Header identifiers in HTML, LaTeX, and ConTeXt

**Extension: header\_attributes**

Headers can be assigned attributes using this syntax at the end of the line containing the header text:

{#identifier .class .class key=value key=value}

Although this syntax allows assignment of classes and key/value attributes, only identifiers currently have any affect in the writers (and only in some writers: HTML, LaTeX, ConTeXt, Textile, AsciiDoc). Thus, for example, the following headers will all be assigned the identifier foo:

# My header

## My header

## My other header

(This syntax is compatible with [PHP Markdown Extra](http://www.michelf.com/projects/php-markdown/extra/).)

Headers with the class unnumbered will not be numbered, even if --number-sections is specified. A single hyphen (-) in an attribute context is equivalent to .unnumbered, and preferable in non-English documents. So,

# My header

is just the same as

# My header

**Extension: auto\_identifiers**

A header without an explicitly specified identifier will be automatically assigned a unique identifier based on the header text. To derive the identifier from the header text,

* Remove all formatting, links, etc.
* Remove all footnotes.
* Remove all punctuation, except underscores, hyphens, and periods.
* Replace all spaces and newlines with hyphens.
* Convert all alphabetic characters to lowercase.
* Remove everything up to the first letter (identifiers may not begin with a number or punctuation mark).
* If nothing is left after this, use the identifier section.

Thus, for example,

|  |  |
| --- | --- |
| Header | Identifier |
| Header identifiers in HTML | header-identifiers-in-html |
| *Dogs*?–in *my* house? | dogs--in-my-house |
| [HTML](http://www.w3.org/TR/html40/), [S5](http://meyerweb.com/eric/tools/s5/), or [RTF](http://en.wikipedia.org/wiki/Rich_Text_Format)? | html-s5-or-rtf |
| 3. Applications | applications |
| 33 | section |

These rules should, in most cases, allow one to determine the identifier from the header text. The exception is when several headers have the same text; in this case, the first will get an identifier as described above; the second will get the same identifier with -1 appended; the third with -2; and so on.

These identifiers are used to provide link targets in the table of contents generated by the --toc|--table-of-contents option. They also make it easy to provide links from one section of a document to another. A link to this section, for example, might look like this:

See the section on [header identifiers](#header-identifiers-in-html-latex-and-context).

Note, however, that this method of providing links to sections works only in HTML, LaTeX, and ConTeXt formats.

If the --section-divs option is specified, then each section will be wrapped in a div (or a section, if --html5 was specified), and the identifier will be attached to the enclosing <div> (or <section>) tag rather than the header itself. This allows entire sections to be manipulated using javascript or treated differently in CSS.

**Extension: implicit\_header\_references**

Pandoc behaves as if reference links have been defined for each header. So, instead of

[header identifiers](#header-identifiers-in-html)

you can simply write

[header identifiers]

or

[header identifiers][]

or

[the section on header identifiers][header identifiers]

If there are multiple headers with identical text, the corresponding reference will link to the first one only, and you will need to use explicit links to link to the others, as described above.

Unlike regular reference links, these references are case-sensitive.

Note: if you have defined an explicit identifier for a header, then implicit references to it will not work.

## Block quotations

Markdown uses email conventions for quoting blocks of text. A block quotation is one or more paragraphs or other block elements (such as lists or headers), with each line preceded by a > character and a space. (The > need not start at the left margin, but it should not be indented more than three spaces.)

This is a block quote. This paragraph has two lines.

1. This is a list inside a block quote.
2. Second item.

A “lazy” form, which requires the > character only on the first line of each block, is also allowed:

This is a block quote. This paragraph has two lines.

1. This is a list inside a block quote.
2. Second item.

Among the block elements that can be contained in a block quote are other block quotes. That is, block quotes can be nested:

This is a block quote.

A block quote within a block quote.

**Extension: blank\_before\_blockquote**

Standard markdown syntax does not require a blank line before a block quote. Pandoc does require this (except, of course, at the beginning of the document). The reason for the requirement is that it is all too easy for a > to end up at the beginning of a line by accident (perhaps through line wrapping). So, unless the markdown\_strict format is used, the following does not produce a nested block quote in pandoc:

This is a block quote. > Nested.

## Verbatim (code) blocks

### Indented code blocks

A block of text indented four spaces (or one tab) is treated as verbatim text: that is, special characters do not trigger special formatting, and all spaces and line breaks are preserved. For example,

if (a > 3) {  
 moveShip(5 \* gravity, DOWN);  
}

The initial (four space or one tab) indentation is not considered part of the verbatim text, and is removed in the output.

Note: blank lines in the verbatim text need not begin with four spaces.

### Fenced code blocks

**Extension: fenced\_code\_blocks**

In addition to standard indented code blocks, Pandoc supports *fenced* code blocks. These begin with a row of three or more tildes (~) or backticks (`) and end with a row of tildes or backticks that must be at least as long as the starting row. Everything between these lines is treated as code. No indentation is necessary:

if (a > 3) {  
 moveShip(5 \* gravity, DOWN);  
}

Like regular code blocks, fenced code blocks must be separated from surrounding text by blank lines.

If the code itself contains a row of tildes or backticks, just use a longer row of tildes or backticks at the start and end:

~~~~~~~~~~  
code including tildes  
~~~~~~~~~~

Optionally, you may attach attributes to the code block using this syntax:

qsort [] = []  
qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++  
 qsort (filter (>= x) xs)

Here mycode is an identifier, haskell and numberLines are classes, and startFrom is an attribute with value 100. Some output formats can use this information to do syntax highlighting. Currently, the only output formats that uses this information are HTML and LaTeX. If highlighting is supported for your output format and language, then the code block above will appear highlighted, with numbered lines. (To see which languages are supported, do pandoc --version.) Otherwise, the code block above will appear as follows:

A shortcut form can also be used for specifying the language of the code block:

qsort [] = []

This is equivalent to:

qsort [] = []

To prevent all highlighting, use the --no-highlight flag. To set the highlighting style, use --highlight-style.

## Line blocks

**Extension: line\_blocks**

A line block is a sequence of lines beginning with a vertical bar (|) followed by a space. The division into lines will be preserved in the output, as will any leading spaces; otherwise, the lines will be formatted as markdown. This is useful for verse and addresses:

The limerick packs laughs anatomical  
In space that is quite economical.  
   But the good ones I’ve seen  
   So seldom are clean  
And the clean ones so seldom are comical

200 Main St.  
Berkeley, CA 94718

The lines can be hard-wrapped if needed, but the continuation line must begin with a space.

The Right Honorable Most Venerable and Righteous Samuel L. Constable, Jr.  
200 Main St.  
Berkeley, CA 94718

This syntax is borrowed from [reStructuredText](http://docutils.sourceforge.net/docs/ref/rst/introduction.html).

## Lists

### Bullet lists

A bullet list is a list of bulleted list items. A bulleted list item begins with a bullet (\*, +, or -). Here is a simple example:

* one
* two
* three

This will produce a “compact” list. If you want a “loose” list, in which each item is formatted as a paragraph, put spaces between the items:

* one
* two
* three

The bullets need not be flush with the left margin; they may be indented one, two, or three spaces. The bullet must be followed by whitespace.

List items look best if subsequent lines are flush with the first line (after the bullet):

* here is my first list item.
* and my second.

But markdown also allows a “lazy” format:

* here is my first list item.
* and my second.

### The four-space rule

A list item may contain multiple paragraphs and other block-level content. However, subsequent paragraphs must be preceded by a blank line and indented four spaces or a tab. The list will look better if the first paragraph is aligned with the rest:

* First paragraph.
* Continued.
* Second paragraph. With a code block, which must be indented eight spaces:
* { code }

List items may include other lists. In this case the preceding blank line is optional. The nested list must be indented four spaces or one tab:

* fruits
  + apples
    - macintosh
    - red delicious
  + pears
  + peaches
* vegetables
  + broccoli
  + chard

As noted above, markdown allows you to write list items “lazily,” instead of indenting continuation lines. However, if there are multiple paragraphs or other blocks in a list item, the first line of each must be indented.

* A lazy, lazy, list item.
* Another one; this looks bad but is legal.
* Second paragraph of second list item.

**Note:** Although the four-space rule for continuation paragraphs comes from the official [markdown syntax guide](http://daringfireball.net/projects/markdown/syntax#list), the reference implementation, Markdown.pl, does not follow it. So pandoc will give different results than Markdown.pl when authors have indented continuation paragraphs fewer than four spaces.

The [markdown syntax guide](http://daringfireball.net/projects/markdown/syntax#list) is not explicit whether the four-space rule applies to *all* block-level content in a list item; it only mentions paragraphs and code blocks. But it implies that the rule applies to all block-level content (including nested lists), and pandoc interprets it that way.

### Ordered lists

Ordered lists work just like bulleted lists, except that the items begin with enumerators rather than bullets.

In standard markdown, enumerators are decimal numbers followed by a period and a space. The numbers themselves are ignored, so there is no difference between this list:

1. one
2. two
3. three

and this one:

1. one
2. two
3. three

**Extension: fancy\_lists**

Unlike standard markdown, Pandoc allows ordered list items to be marked with uppercase and lowercase letters and roman numerals, in addition to arabic numerals. List markers may be enclosed in parentheses or followed by a single right-parentheses or period. They must be separated from the text that follows by at least one space, and, if the list marker is a capital letter with a period, by at least two spaces.[[1]](#footnote-56)

do not get treated as list items.

This rule will not prevent

(C) 2007 Joe Smith

from being interpreted as a list item. In this case, a backslash escape can be used:

(C\) 2007 Joe Smith

The fancy\_lists extension also allows ‘#’ to be used as an ordered list marker in place of a numeral:

1. one
2. two

**Extension: startnum**

Pandoc also pays attention to the type of list marker used, and to the starting number, and both of these are preserved where possible in the output format. Thus, the following yields a list with numbers followed by a single parenthesis, starting with 9, and a sublist with lowercase roman numerals:

1. Ninth
2. Tenth
3. Eleventh
   1. subone
   2. subtwo
   3. subthree

Pandoc will start a new list each time a different type of list marker is used. So, the following will create three lists:

1. Two
2. Three
3. Four

* Five

If default list markers are desired, use #.:

1. one
2. two
3. three

### Definition lists

**Extension: definition\_lists**

Pandoc supports definition lists, using a syntax inspired by [PHP Markdown Extra](http://www.michelf.com/projects/php-markdown/extra/) and [reStructuredText](http://docutils.sourceforge.net/docs/ref/rst/introduction.html):[[2]](#footnote-58)

Term 1

Definition 1

Term 2 with *inline markup*

Definition 2

{ some code, part of Definition 2 }

Third paragraph of definition 2.

Each term must fit on one line, which may optionally be followed by a blank line, and must be followed by one or more definitions. A definition begins with a colon or tilde, which may be indented one or two spaces. The body of the definition (including the first line, aside from the colon or tilde) should be indented four spaces. A term may have multiple definitions, and each definition may consist of one or more block elements (paragraph, code block, list, etc.), each indented four spaces or one tab stop.

If you leave space after the definition (as in the example above), the blocks of the definitions will be considered paragraphs. In some output formats, this will mean greater spacing between term/definition pairs. For a compact definition list, do not leave space between the definition and the next term:

Term 1

Definition 1

Term 2

Definition 2a

Definition 2b

### Numbered example lists

**Extension: example\_lists**

The special list marker @ can be used for sequentially numbered examples. The first list item with a @ marker will be numbered ‘1’, the next ‘2’, and so on, throughout the document. The numbered examples need not occur in a single list; each new list using @ will take up where the last stopped. So, for example:

1. My first example will be numbered (1).
2. My second example will be numbered (2).

Explanation of examples.

1. My third example will be numbered (3).

Numbered examples can be labeled and referred to elsewhere in the document:

1. This is a good example.

As (4) illustrates, …

The label can be any string of alphanumeric characters, underscores, or hyphens.

### Compact and loose lists

Pandoc behaves differently from Markdown.pl on some “edge cases” involving lists. Consider this source:

* First
* Second:
* Fee
* Fie
* Foe
* Third

Pandoc transforms this into a “compact list” (with no <p> tags around “First”, “Second”, or “Third”), while markdown puts <p> tags around “Second” and “Third” (but not “First”), because of the blank space around “Third”. Pandoc follows a simple rule: if the text is followed by a blank line, it is treated as a paragraph. Since “Second” is followed by a list, and not a blank line, it isn’t treated as a paragraph. The fact that the list is followed by a blank line is irrelevant. (Note: Pandoc works this way even when the markdown\_strict format is specified. This behavior is consistent with the official markdown syntax description, even though it is different from that of Markdown.pl.)

### Ending a list

What if you want to put an indented code block after a list?

* item one
* item two
* { my code block }

Trouble! Here pandoc (like other markdown implementations) will treat { my code block } as the second paragraph of item two, and not as a code block.

To “cut off” the list after item two, you can insert some non-indented content, like an HTML comment, which won’t produce visible output in any format:

* item one
* item two

{ my code block }

You can use the same trick if you want two consecutive lists instead of one big list:

1. one
2. two
3. three
4. uno
5. dos
6. tres

## Horizontal rules

A line containing a row of three or more \*, -, or \_ characters (optionally separated by spaces) produces a horizontal rule:

## Tables

Four kinds of tables may be used. The first three kinds presuppose the use of a fixed-width font, such as Courier. The fourth kind can be used with proportionally spaced fonts, as it does not require lining up columns.

### Simple tables

**Extension: simple\_tables, table\_captions**

Simple tables look like this:

|  |  |  |  |
| --- | --- | --- | --- |
| Right | Left | Center | Default |
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

Demonstration of simple table syntax.

The headers and table rows must each fit on one line. Column alignments are determined by the position of the header text relative to the dashed line below it:[[3]](#footnote-66)

* If the dashed line is flush with the header text on the right side but extends beyond it on the left, the column is right-aligned.
* If the dashed line is flush with the header text on the left side but extends beyond it on the right, the column is left-aligned.
* If the dashed line extends beyond the header text on both sides, the column is centered.
* If the dashed line is flush with the header text on both sides, the default alignment is used (in most cases, this will be left).

The table must end with a blank line, or a line of dashes followed by a blank line. A caption may optionally be provided (as illustrated in the example above). A caption is a paragraph beginning with the string Table: (or just :), which will be stripped off. It may appear either before or after the table.

The column headers may be omitted, provided a dashed line is used to end the table. For example:

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

When headers are omitted, column alignments are determined on the basis of the first line of the table body. So, in the tables above, the columns would be right, left, center, and right aligned, respectively.

### Multiline tables

**Extension: multiline\_tables, table\_captions**

Multiline tables allow headers and table rows to span multiple lines of text (but cells that span multiple columns or rows of the table are not supported). Here is an example:

|  |  |  |  |
| --- | --- | --- | --- |
| Centered Header | Default Aligned | Right Aligned | Left Aligned |
| First | row | 12.0 | Example of a row that spans multiple lines. |
| Second | row | 5.0 | Here’s another one. Note the blank line between rows. |

Here’s the caption. It, too, may span multiple lines.

These work like simple tables, but with the following differences:

* They must begin with a row of dashes, before the header text (unless the headers are omitted).
* They must end with a row of dashes, then a blank line.
* The rows must be separated by blank lines.

In multiline tables, the table parser pays attention to the widths of the columns, and the writers try to reproduce these relative widths in the output. So, if you find that one of the columns is too narrow in the output, try widening it in the markdown source.

Headers may be omitted in multiline tables as well as simple tables:

|  |  |  |  |
| --- | --- | --- | --- |
| First | row | 12.0 | Example of a row that spans multiple lines. |
| Second | row | 5.0 | Here’s another one. Note the blank line between rows. |

Here’s a multiline table without headers.

It is possible for a multiline table to have just one row, but the row should be followed by a blank line (and then the row of dashes that ends the table), or the table may be interpreted as a simple table.

### Grid tables

**Extension: grid\_tables, table\_captions**

Grid tables look like this:

|  |  |  |
| --- | --- | --- |
| Fruit | Price | Advantages |
| Bananas | $1.34 | * built-in wrapper * bright color |
| Oranges | $2.10 | * cures scurvy * tasty |

Sample grid table.

The row of =s separates the header from the table body, and can be omitted for a headerless table. The cells of grid tables may contain arbitrary block elements (multiple paragraphs, code blocks, lists, etc.). Alignments are not supported, nor are cells that span multiple columns or rows. Grid tables can be created easily using [Emacs table mode](http://table.sourceforge.net/).

### Pipe tables

**Extension: pipe\_tables, table\_captions**

Pipe tables look like this:

|  |  |  |  |
| --- | --- | --- | --- |
| Right | Left | Default | Center |
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

Demonstration of simple table syntax.

The syntax is [the same as in PHP markdown extra](http://michelf.ca/projects/php-markdown/extra/#table). The beginning and ending pipe characters are optional, but pipes are required between all columns. The colons indicate column alignment as shown. The header can be omitted, but the horizontal line must still be included, as it defines column alignments.

Since the pipes indicate column boundaries, columns need not be vertically aligned, as they are in the above example. So, this is a perfectly legal (though ugly) pipe table:

|  |  |
| --- | --- |
| fruit | price |
| apple | 2.05 |
| pear | 1.37 |
| orange | 3.09 |

The cells of pipe tables cannot contain block elements like paragraphs and lists, and cannot span multiple lines.

Note: Pandoc also recognizes pipe tables of the following form, as can produced by Emacs’ orgtbl-mode:

|  |  |
| --- | --- |
| One | Two |
| my | table |
| is | nice |

The difference is that + is used instead of |. Other orgtbl features are not supported. In particular, to get non-default column alignment, you’ll need to add colons as above.

## Title block

**Extension: pandoc\_title\_block**

If the file begins with a title block

% title % author(s) (separated by semicolons) % date

it will be parsed as bibliographic information, not regular text. (It will be used, for example, in the title of standalone LaTeX or HTML output.) The block may contain just a title, a title and an author, or all three elements. If you want to include an author but no title, or a title and a date but no author, you need a blank line:

% % Author

% My title % % June 15, 2006

The title may occupy multiple lines, but continuation lines must begin with leading space, thus:

% My title on multiple lines

If a document has multiple authors, the authors may be put on separate lines with leading space, or separated by semicolons, or both. So, all of the following are equivalent:

% Author One Author Two

% Author One; Author Two

% Author One; Author Two

The date must fit on one line.

All three metadata fields may contain standard inline formatting (italics, links, footnotes, etc.).

Title blocks will always be parsed, but they will affect the output only when the --standalone (-s) option is chosen. In HTML output, titles will appear twice: once in the document head – this is the title that will appear at the top of the window in a browser – and once at the beginning of the document body. The title in the document head can have an optional prefix attached (--title-prefix or -T option). The title in the body appears as an H1 element with class “title”, so it can be suppressed or reformatted with CSS. If a title prefix is specified with -T and no title block appears in the document, the title prefix will be used by itself as the HTML title.

The man page writer extracts a title, man page section number, and other header and footer information from the title line. The title is assumed to be the first word on the title line, which may optionally end with a (single-digit) section number in parentheses. (There should be no space between the title and the parentheses.) Anything after this is assumed to be additional footer and header text. A single pipe character (|) should be used to separate the footer text from the header text. Thus,

% PANDOC(1)

will yield a man page with the title PANDOC and section 1.

% PANDOC(1) Pandoc User Manuals

will also have “Pandoc User Manuals” in the footer.

% PANDOC(1) Pandoc User Manuals | Version 4.0

will also have “Version 4.0” in the header.

## YAML metadata block

**Extension: yaml\_metadata\_block**

A YAML metadata block is a valid YAML object, delimited by a line of three hyphens (---) at the top and a line of three hyphens (---) or three dots (...) at the bottom. A YAML metadata block may occur anywhere in the document, but if it is not at the beginning, it must be preceded by a blank line.

Metadata will be taken from the fields of the YAML object and added to any existing document metadata. Metadata can contain lists and objects (nested arbitrarily), but all string scalars will be interpreted as markdown. Fields with names ending in an underscore will be ignored by pandoc. (They may be given a role by external processors.)

A document may contain multiple metadata blocks. The metadata fields will be combined through a *left-biased union*: if two metadata blocks attempt to set the same field, the value from the first block will be taken.

Note that YAML escaping rules must be followed. Thus, for example, if a title contains a colon, it must be quoted. The pipe character (|) can be used to begin an indented block that will be interpreted literally, without need for escaping. This form is necessary when the field contains blank lines:

Template variables will be set automatically from the metadata. Thus, for example, in writing HTML, the variable abstract will be set to the HTML equivalent of the markdown in the abstract field:

This is the abstract.

It consists of two paragraphs.

Note: The author variable in the default templates expects a simple list or string. To use the structured authors in the example, you would need a custom template. For example:

()

## Backslash escapes

**Extension: all\_symbols\_escapable**

Except inside a code block or inline code, any punctuation or space character preceded by a backslash will be treated literally, even if it would normally indicate formatting. Thus, for example, if one writes

*\*hello\**

one will get

*hello*

instead of

hello

This rule is easier to remember than standard markdown’s rule, which allows only the following characters to be backslash-escaped:

`\*\_{}>#+-.!

(However, if the markdown\_strict format is used, the standard markdown rule will be used.)

A backslash-escaped space is parsed as a nonbreaking space. It will appear in TeX output as ~ and in HTML and XML as \&#160; or \&nbsp;.

A backslash-escaped newline (i.e. a backslash occurring at the end of a line) is parsed as a hard line break. It will appear in TeX output as \\ and in HTML as <br />. This is a nice alternative to markdown’s “invisible” way of indicating hard line breaks using two trailing spaces on a line.

Backslash escapes do not work in verbatim contexts.

## Smart punctuation

**Extension**

If the --smart option is specified, pandoc will produce typographically correct output, converting straight quotes to curly quotes, --- to em-dashes, -- to en-dashes, and ... to ellipses. Nonbreaking spaces are inserted after certain abbreviations, such as “Mr.”

Note: if your LaTeX template uses the csquotes package, pandoc will detect automatically this and use \enquote{...} for quoted text.

## Inline formatting

### Emphasis

To *emphasize* some text, surround it with \*s or \_, like this:

This text is *emphasized with underscores*, and this is *emphasized with asterisks*.

Double \* or \_ produces **strong emphasis**:

This is **strong emphasis** and **with underscores**.

A \* or \_ character surrounded by spaces, or backslash-escaped, will not trigger emphasis:

This is \* not emphasized \*, and \*neither is this\*.

**Extension: intraword\_underscores**

Because \_ is sometimes used inside words and identifiers, pandoc does not interpret a \_ surrounded by alphanumeric characters as an emphasis marker. If you want to emphasize just part of a word, use \*:

feas*ible*, not feas*able*.

### Strikeout

**Extension: strikeout**

To strikeout a section of text with a horizontal line, begin and end it with ~~. Thus, for example,

This ~~is deleted text.~~

### Superscripts and subscripts

**Extension: superscript, subscript**

Superscripts may be written by surrounding the superscripted text by ^ characters; subscripts may be written by surrounding the subscripted text by ~ characters. Thus, for example,

H2O is a liquid. 210 is 1024.

If the superscripted or subscripted text contains spaces, these spaces must be escaped with backslashes. (This is to prevent accidental superscripting and subscripting through the ordinary use of ~ and ^.) Thus, if you want the letter P with ‘a cat’ in subscripts, use P~a\ cat~, not P~a cat~.

### Verbatim

To make a short span of text verbatim, put it inside backticks:

What is the difference between >>= and >>?

If the verbatim text includes a backtick, use double backticks:

Here is a literal backtick `.

(The spaces after the opening backticks and before the closing backticks will be ignored.)

The general rule is that a verbatim span starts with a string of consecutive backticks (optionally followed by a space) and ends with a string of the same number of backticks (optionally preceded by a space).

Note that backslash-escapes (and other markdown constructs) do not work in verbatim contexts:

This is a backslash followed by an asterisk: \\*.

**Extension: inline\_code\_attributes**

Attributes can be attached to verbatim text, just as with [fenced code blocks](#fenced-code-blocks):

<$>

## Math

**Extension: tex\_math\_dollars**

Anything between two $ characters will be treated as TeX math. The opening $ must have a character immediately to its right, while the closing $ must have a character immediately to its left. Thus, $20,000 and $30,000 won’t parse as math. If for some reason you need to enclose text in literal $ characters, backslash-escape them and they won’t be treated as math delimiters.

TeX math will be printed in all output formats. How it is rendered depends on the output format:

Markdown, LaTeX, Org-Mode, ConTeXt

It will appear verbatim between $ characters.

reStructuredText

It will be rendered using an interpreted text role :math:, as described

[here](http://www.american.edu/econ/itex2mml/mathhack.rst).

AsciiDoc

It will be rendered as latexmath:[...].

Texinfo

It will be rendered inside a @math command.

groff man

It will be rendered verbatim without $’s.

MediaWiki

It will be rendered inside <math> tags.

Textile

It will be rendered inside <span class="math"> tags.

RTF, OpenDocument, ODT

It will be rendered, if possible, using unicode characters,

and will otherwise appear verbatim.

Docbook

If the --mathml flag is used, it will be rendered using mathml

in an inlineequation or informalequation tag. Otherwise it will be rendered, if possible, using unicode characters.

Docx

It will be rendered using OMML math markup.

FictionBook2

If the --webtex option is used, formulas are rendered as images

using Google Charts or other compatible web service, downloaded and embedded in the e-book. Otherwise, they will appear verbatim.

HTML, Slidy, DZSlides, S5, EPUB

The way math is rendered in HTML will depend on the

command-line options selected:

1. The default is to render TeX math as far as possible using unicode characters, as with RTF, DocBook, and OpenDocument output. Formulas are put inside a span with class="math", so that they may be styled differently from the surrounding text if needed.
2. If the --latexmathml option is used, TeX math will be displayed between $ or $$ characters and put in <span> tags with class LaTeX. The [LaTeXMathML] script will be used to render it as formulas. (This trick does not work in all browsers, but it works in Firefox. In browsers that do not support LaTeXMathML, TeX math will appear verbatim between $ characters.)
3. If the --jsmath option is used, TeX math will be put inside <span> tags (for inline math) or <div> tags (for display math) with class math. The [jsMath] script will be used to render it.
4. If the --mimetex option is used, the [mimeTeX] CGI script will be called to generate images for each TeX formula. This should work in all browsers. The --mimetex option takes an optional URL as argument. If no URL is specified, it will be assumed that the mimeTeX CGI script is at /cgi-bin/mimetex.cgi.
5. If the --gladtex option is used, TeX formulas will be enclosed in <eq> tags in the HTML output. The resulting htex file may then be processed by [gladTeX], which will produce image files for each formula and an html file with links to these images. So, the procedure is:

* pandoc -s --gladtex myfile.txt -o myfile.htex  
  gladtex -d myfile-images myfile.htex  
  # produces myfile.html and images in myfile-images

1. If the --webtex option is used, TeX formulas will be converted to <img> tags that link to an external script that converts formulas to images. The formula will be URL-encoded and concatenated with the URL provided. If no URL is specified, the Google Chart API will be used (http://chart.apis.google.com/chart?cht=tx&chl=).
2. If the --mathjax option is used, TeX math will be displayed between \(...\) (for inline math) or \[...\] (for display math) and put in <span> tags with class math. The [MathJax] script will be used to render it as formulas.

## Raw HTML

**Extension: raw\_html**

Markdown allows you to insert raw HTML (or DocBook) anywhere in a document (except verbatim contexts, where <, >, and & are interpreted literally). (Technically this is not an extension, since standard markdown allows it, but it has been made an extension so that it can be disabled if desired.)

The raw HTML is passed through unchanged in HTML, S5, Slidy, Slideous, DZSlides, EPUB, Markdown, and Textile output, and suppressed in other formats.

**Extension: markdown\_in\_html\_blocks**

Standard markdown allows you to include HTML “blocks”: blocks of HTML between balanced tags that are separated from the surrounding text with blank lines, and start and end at the left margin. Within these blocks, everything is interpreted as HTML, not markdown; so (for example), \* does not signify emphasis.

Pandoc behaves this way when the markdown\_strict format is used; but by default, pandoc interprets material between HTML block tags as markdown. Thus, for example, Pandoc will turn

*one*

[a link](http://google.com)

into

one

a link

whereas Markdown.pl will preserve it as is.

There is one exception to this rule: text between <script> and <style> tags is not interpreted as markdown.

This departure from standard markdown should make it easier to mix markdown with HTML block elements. For example, one can surround a block of markdown text with <div> tags without preventing it from being interpreted as markdown.

## Raw TeX

**Extension: raw\_tex**

In addition to raw HTML, pandoc allows raw LaTeX, TeX, and ConTeXt to be included in a document. Inline TeX commands will be preserved and passed unchanged to the LaTeX and ConTeXt writers. Thus, for example, you can use LaTeX to include BibTeX citations:

This result was proved in .

Note that in LaTeX environments, like

the material between the begin and end tags will be interpreted as raw LaTeX, not as markdown.

Inline LaTeX is ignored in output formats other than Markdown, LaTeX, and ConTeXt.

## LaTeX macros

**Extension: latex\_macros**

For output formats other than LaTeX, pandoc will parse LaTeX \newcommand and \renewcommand definitions and apply the resulting macros to all LaTeX math. So, for example, the following will work in all output formats, not just LaTeX:

In LaTeX output, the \newcommand definition will simply be passed unchanged to the output.

## Links

Markdown allows links to be specified in several ways.

### Automatic links

If you enclose a URL or email address in pointy brackets, it will become a link:

<http://google.com> [sam@green.eggs.ham](mailto:sam@green.eggs.ham)

### Inline links

An inline link consists of the link text in square brackets, followed by the URL in parentheses. (Optionally, the URL can be followed by a link title, in quotes.)

This is an [inline link](/url), and here’s [one with a title](http://fsf.org).

There can be no space between the bracketed part and the parenthesized part. The link text can contain formatting (such as emphasis), but the title cannot.

### Reference links

An *explicit* reference link has two parts, the link itself and the link definition, which may occur elsewhere in the document (either before or after the link).

The link consists of link text in square brackets, followed by a label in square brackets. (There can be space between the two.) The link definition consists of the bracketed label, followed by a colon and a space, followed by the URL, and optionally (after a space) a link title either in quotes or in parentheses.

Here are some examples:

The URL may optionally be surrounded by angle brackets:

The title may go on the next line:

Note that link labels are not case sensitive. So, this will work:

Here is [my link](/bar/baz)

In an *implicit* reference link, the second pair of brackets is empty, or omitted entirely:

See [my website](http://foo.bar.baz), or [my website](http://foo.bar.baz).

Note: In Markdown.pl and most other markdown implementations, reference link definitions cannot occur in nested constructions such as list items or block quotes. Pandoc lifts this arbitrary seeming restriction. So the following is fine in pandoc, though not in most other implementations:

My block [quote](/foo).

### Internal links

To link to another section of the same document, use the automatically generated identifier (see [Header identifiers in HTML, LaTeX, and ConTeXt](#header-identifiers-in-html-latex-and-context), below). For example:

See the [Introduction](#introduction).

or

See the [Introduction](#introduction).

Internal links are currently supported for HTML formats (including HTML slide shows and EPUB), LaTeX, and ConTeXt.

## Images

A link immediately preceded by a ! will be treated as an image. The link text will be used as the image’s alt text:

la lune

la lune

movie reel

movie reel

### Pictures with captions

**Extension: implicit\_figures**

An image occurring by itself in a paragraph will be rendered as a figure with a caption.[[4]](#footnote-101) (In LaTeX, a figure environment will be used; in HTML, the image will be placed in a div with class figure, together with a caption in a p with class caption.) The image’s alt text will be used as the caption.

This is the caption

This is the caption

If you just want a regular inline image, just make sure it is not the only thing in the paragraph. One way to do this is to insert a nonbreaking space after the image:

This image won’t be a figure

## Footnotes

**Extension: footnotes**

Pandoc’s markdown allows footnotes, using the following syntax:

Here is a footnote reference,[[5]](#footnote-103) and another.[[6]](#footnote-104)

This paragraph won’t be part of the note, because it isn’t indented.

The identifiers in footnote references may not contain spaces, tabs, or newlines. These identifiers are used only to correlate the footnote reference with the note itself; in the output, footnotes will be numbered sequentially.

The footnotes themselves need not be placed at the end of the document. They may appear anywhere except inside other block elements (lists, block quotes, tables, etc.).

**Extension: inline\_notes**

Inline footnotes are also allowed (though, unlike regular notes, they cannot contain multiple paragraphs). The syntax is as follows:

Here is an inline note.[[7]](#footnote-105)

Inline and regular footnotes may be mixed freely.

## Citations

**Extension: citations**

Using an external filter, pandoc-citeproc, pandoc can automatically generate citations and a bibliography in a number of styles. Basic usage is

pandoc –filter pandoc-citeproc myinput.txt

In order to use this feature, you will need to specify a bibliography file using the bibliography metadata field in a YAML metadata section. The bibliography may have any of these formats:

|  |  |
| --- | --- |
| Format | File extension |
| MODS | .mods |
| BibLaTeX | .bib |
| BibTeX | .bibtex |
| RIS | .ris |
| EndNote | .enl |
| EndNote XML | .xml |
| ISI | .wos |
| MEDLINE | .medline |
| Copac | .copac |
| JSON citeproc | .json |

Note that .bib can generally be used with both BibTeX and BibLaTeX files, but you can use .bibtex to force BibTeX.

Alternatively you can use a references field in the document’s YAML metadata. This should include an array of YAML-encoded references, for example:

(The program mods2yaml, which comes with pandoc-citeproc, can help produce these from a MODS reference collection.)

By default, pandoc-citeproc will use a Chicago author-date format for citations and references. To use another style, you will need to specify a [CSL] 1.0 style file in the csl metadata field. A primer on creating and modifying CSL styles can be found at <http://citationstyles.org/downloads/primer.html>. A repository of CSL styles can be found at <https://github.com/citation-style-language/styles>. See also <http://zotero.org/styles> for easy browsing.

Citations go inside square brackets and are separated by semicolons. Each citation must have a key, composed of ‘@’ + the citation identifier from the database, and may optionally have a prefix, a locator, and a suffix. The citation key must begin with a letter or \_, and may contain alphanumerics, \_, and internal punctuation characters (:.#$%&-+?<>~/). Here are some examples:

Blah blah [see @doe99, pp. 33-35; also @smith04, ch. 1].

Blah blah [@doe99, pp. 33-35, 38-39 and \*passim\*].

Blah blah [@smith04; @doe99].

A minus sign (-) before the @ will suppress mention of the author in the citation. This can be useful when the author is already mentioned in the text:

Smith says blah [-@smith04].

You can also write an in-text citation, as follows:

@smith04 says blah.

@smith04 [p. 33] says blah.

If the style calls for a list of works cited, it will be placed at the end of the document. Normally, you will want to end your document with an appropriate header:

last paragraph…

# References

The bibliography will be inserted after this header. Note that the unnumbered class will be added to this header, so that the section will not be numbered.

If you want to include items in the bibliography without actually citing them in the body text, you can define a dummy nocite metadata field and put the citations there:

@item3

In this example, the document will contain a citation for item3 only, but the bibliography will contain entries for item1, item2, and item3.

## Non-pandoc extensions

The following markdown syntax extensions are not enabled by default in pandoc, but may be enabled by adding +EXTENSION to the format name, where EXTENSION is the name of the extension. Thus, for example, markdown+hard\_line\_breaks is markdown with hard line breaks.

**Extension: lists\_without\_preceding\_blankline**  
Allow a list to occur right after a paragraph, with no intervening blank space.

**Extension: hard\_line\_breaks**  
Causes all newlines within a paragraph to be interpreted as hard line breaks instead of spaces.

**Extension: ignore\_line\_breaks**  
Causes newlines within a paragraph to be ignored, rather than being treated as spaces or as hard line breaks. This option is intended for use with East Asian languages where spaces are not used between words, but text is divided into lines for readability.

**Extension: tex\_math\_single\_backslash**  
Causes anything between \( and \) to be interpreted as inline TeX math, and anything between \[ and \] to be interpreted as display TeX math. Note: a drawback of this extension is that it precludes escaping ( and [.

**Extension: tex\_math\_double\_backslash**  
Causes anything between \\( and \\) to be interpreted as inline TeX math, and anything between \\[ and \\] to be interpreted as display TeX math.

**Extension: markdown\_attribute**  
By default, pandoc interprets material inside block-level tags as markdown. This extension changes the behavior so that markdown is only parsed inside block-level tags if the tags have the attribute markdown=1.

**Extension: mmd\_title\_block**  
Enables a [MultiMarkdown](http://fletcherpenney.net/multimarkdown/) style title block at the top of the document, for example:

Title: My title Author: John Doe Date: September 1, 2008 Comment: This is a sample mmd title block, with a field spanning multiple lines.

See the MultiMarkdown documentation for details. If pandoc\_title\_block or yaml\_metadata\_block is enabled, it will take precedence over mmd\_title\_block.

**Extension: abbreviations**  
Parses PHP Markdown Extra abbreviation keys, like

\*[HTML](http://www.w3.org/TR/html40/): Hyper Text Markup Language

Note that the pandoc document model does not support abbreviations, so if this extension is enabled, abbreviation keys are simply skipped (as opposed to being parsed as paragraphs).

**Extension: autolink\_bare\_uris**  
Makes all absolute URIs into links, even when not surrounded by pointy braces <...>.

**Extension: ascii\_identifiers**  
Causes the identifiers produced by auto\_identifiers to be pure ASCII. Accents are stripped off of accented latin letters, and non-latin letters are omitted.

**Extension: link\_attributes**  
Parses multimarkdown style key-value attributes on link and image references. Note that pandoc’s internal document model provides nowhere to put these, so they are presently just ignored.

**Extension: mmd\_header\_identifiers**  
Parses multimarkdown style header identifiers (in square brackets, after the header but before any trailing #s in an ATX header).

## Markdown variants

In addition to pandoc’s extended markdown, the following markdown variants are supported:

markdown\_phpextra (PHP Markdown Extra)

footnotes, pipe\_tables, raw\_html, markdown\_attribute,

fenced\_code\_blocks, definition\_lists, intraword\_underscores, header\_attributes, abbreviations.

markdown\_github (Github-flavored Markdown)

pipe\_tables, raw\_html, tex\_math\_single\_backslash,

fenced\_code\_blocks, fenced\_code\_attributes, auto\_identifiers, ascii\_identifiers, backtick\_code\_blocks, autolink\_bare\_uris, intraword\_underscores, strikeout, hard\_line\_breaks

markdown\_mmd (MultiMarkdown)

pipe\_tables raw\_html, markdown\_attribute, link\_attributes,

raw\_tex, tex\_math\_double\_backslash, intraword\_underscores, mmd\_title\_block, footnotes, definition\_lists, all\_symbols\_escapable, implicit\_header\_references, auto\_identifiers, mmd\_header\_identifiers

markdown\_strict (Markdown.pl)

raw\_html

## Extensions with formats other than markdown

Some of the extensions discussed above can be used with formats other than markdown:

* auto\_identifiers can be used with latex, rst, mediawiki, and textile input (and is used by default).
* tex\_math\_dollars, tex\_math\_single\_backslash, and tex\_math\_double\_backslash can be used with html input. (This is handy for reading web pages formatted using MathJax, for example.)

# Producing slide shows with Pandoc

You can use Pandoc to produce an HTML + javascript slide presentation that can be viewed via a web browser. There are five ways to do this, using [S5](http://meyerweb.com/eric/tools/s5/), [DZSlides](http://paulrouget.com/dzslides/), [Slidy](http://www.w3.org/Talks/Tools/Slidy/), [Slideous](http://goessner.net/articles/slideous/), or [reveal.js](http://lab.hakim.se/reveal-js/). You can also produce a PDF slide show using LaTeX [beamer](http://www.tex.ac.uk/CTAN/macros/latex/contrib/beamer).

Here’s the markdown source for a simple slide show, habits.txt:

% Habits % John Doe % March 22, 2005

# In the morning

## Getting up

* Turn off alarm
* Get out of bed

## Breakfast

* Eat eggs
* Drink coffee

# In the evening

## Dinner

* Eat spaghetti
* Drink wine

picture of spaghetti

picture of spaghetti

## Going to sleep

* Get in bed
* Count sheep

To produce an HTML/javascript slide show, simply type

pandoc -t FORMAT -s habits.txt -o habits.html

where FORMAT is either s5, slidy, slideous, dzslides, or revealjs.

For Slidy, Slideous, reveal.js, and S5, the file produced by pandoc with the -s/--standalone option embeds a link to javascripts and CSS files, which are assumed to be available at the relative path s5/default (for S5), slideous (for Slideous), reveal.js (for reveal.js), or at the Slidy website at w3.org (for Slidy). (These paths can be changed by setting the slidy-url, slideous-url, revealjs-url, or s5-url variables; see --variable, above.) For DZSlides, the (relatively short) javascript and css are included in the file by default.

With all HTML slide formats, the --self-contained option can be used to produce a single file that contains all of the data necessary to display the slide show, including linked scripts, stylesheets, images, and videos.

To produce a PDF slide show using beamer, type

pandoc -t beamer habits.txt -o habits.pdf

Note that a reveal.js slide show can also be converted to a PDF by printing it to a file from the browser.

## Structuring the slide show

By default, the *slide level* is the highest header level in the hierarchy that is followed immediately by content, and not another header, somewhere in the document. In the example above, level 1 headers are always followed by level 2 headers, which are followed by content, so 2 is the slide level. This default can be overridden using the --slide-level option.

The document is carved up into slides according to the following rules:

* A horizontal rule always starts a new slide.
* A header at the slide level always starts a new slide.
* Headers *below* the slide level in the hierarchy create headers *within* a slide.
* Headers *above* the slide level in the hierarchy create “title slides,” which just contain the section title and help to break the slide show into sections.
* A title page is constructed automatically from the document’s title block, if present. (In the case of beamer, this can be disabled by commenting out some lines in the default template.)

These rules are designed to support many different styles of slide show. If you don’t care about structuring your slides into sections and subsections, you can just use level 1 headers for all each slide. (In that case, level 1 will be the slide level.) But you can also structure the slide show into sections, as in the example above.

Note: in reveal.js slide shows, if slide level is 2, a two-dimensional layout will be produced, with level 1 headers building horizontally and level 2 headers building vertically. It is not recommended that you use deeper nesting of section levels with reveal.js.

## Incremental lists

By default, these writers produces lists that display “all at once.” If you want your lists to display incrementally (one item at a time), use the -i option. If you want a particular list to depart from the default (that is, to display incrementally without the -i option and all at once with the -i option), put it in a block quote:

* Eat spaghetti
* Drink wine

In this way incremental and nonincremental lists can be mixed in a single document.

## Inserting pauses

You can add “pauses” within a slide by including a paragraph containing three dots, separated by spaces:

# Slide with a pause

content before the pause

. . .

content after the pause

## Styling the slides

You can change the style of HTML slides by putting customized CSS files in $DATADIR/s5/default (for S5), $DATADIR/slidy (for Slidy), or $DATADIR/slideous (for Slideous), where $DATADIR is the user data directory (see --data-dir, above). The originals may be found in pandoc’s system data directory (generally $CABALDIR/pandoc-VERSION/s5/default). Pandoc will look there for any files it does not find in the user data directory.

For dzslides, the CSS is included in the HTML file itself, and may be modified there.

For reveal.js, themes can be used by setting the theme variable, for example:

-V theme=moon

Or you can specify a custom stylesheet using the --css option.

To style beamer slides, you can specify a beamer “theme” or “colortheme” using the -V option:

pandoc -t beamer habits.txt -V theme:Warsaw -o habits.pdf

Note that header attributes will turn into slide attributes (on a <div> or <section>) in HTML slide formats, allowing you to style individual slides. In Beamer, the only header attribute that affects slides is the allowframebreaks class, which sets the allowframebreaks option, causing multiple slides to be created if the content overfills the frame. This is recommended especially for bibliographies:

# References

## Speaker notes

reveal.js has good support for speaker notes. You can add notes to your markdown document thus:

This is my note.

* It can contain markdown
* like this list

To show the notes window, press s while viewing the presentation. Notes are not yet supported for other slide formats, but the notes will not appear on the slides themselves.

# EPUB Metadata

EPUB metadata may be specified using the --epub-metadata option, but if the source document is markdown, it is better to use a YAML metadata block. Here is an example:

The following fields are recognized:

identifier

Either a string value or an object with fields text and

scheme. Valid values for scheme are ISBN-10, GTIN-13, UPC, ISMN-10, DOI, LCCN, GTIN-14, ISBN-13, Legal deposit number, URN, OCLC, ISMN-13, ISBN-A, JP, OLCC. title ~ Either a string value, or an object with fields file-as and type, or a list of such objects. Valid values for type are main, subtitle, short, collection, edition, extended. creator ~ Either a string value, or an object with fields role, file-as, and text, or a list of such objects. Valid values for role are [marc relators](http://www.loc.gov/marc/relators/relaterm.html), but pandoc will attempt to translate the human-readable versions (like “author” and “editor”) to the appropriate marc relators. contributor ~ Same format as creator. date ~ A string value in YYYY-MM-DD format. (Only the year is necessary.) Pandoc will attempt to convert other common date formats. language ~ A string value in [RFC5646](http://tools.ietf.org/html/rfc5646) format. Pandoc will default to the local language if nothing is specified. subject ~ A string value or a list of such values. description ~ A string value. type ~ A string value. format ~ A string value. relation ~ A string value. coverage ~ A string value. rights ~ A string value. cover-image ~ A string value (path to cover image). stylesheet ~ A string value (path to CSS stylesheet).

# Literate Haskell support

If you append +lhs (or +literate\_haskell) to an appropriate input or output format (markdown, markdown\_strict, rst, or latex for input or output; beamer, html or html5 for output only), pandoc will treat the document as literate Haskell source. This means that

* In markdown input, “bird track” sections will be parsed as Haskell code rather than block quotations. Text between \begin{code} and \end{code} will also be treated as Haskell code.
* In markdown output, code blocks with classes haskell and literate will be rendered using bird tracks, and block quotations will be indented one space, so they will not be treated as Haskell code. In addition, headers will be rendered setext-style (with underlines) rather than atx-style (with ‘#’ characters). (This is because ghc treats ‘#’ characters in column 1 as introducing line numbers.)
* In restructured text input, “bird track” sections will be parsed as Haskell code.
* In restructured text output, code blocks with class haskell will be rendered using bird tracks.
* In LaTeX input, text in code environments will be parsed as Haskell code.
* In LaTeX output, code blocks with class haskell will be rendered inside code environments.
* In HTML output, code blocks with class haskell will be rendered with class literatehaskell and bird tracks.

Examples:

pandoc -f markdown+lhs -t html

reads literate Haskell source formatted with markdown conventions and writes ordinary HTML (without bird tracks).

pandoc -f markdown+lhs -t html+lhs

writes HTML with the Haskell code in bird tracks, so it can be copied and pasted as literate Haskell source.

# Custom writers

Pandoc can be extended with custom writers written in [lua](http://www.lua.org). (Pandoc includes a lua interpreter, so lua need not be installed separately.)

To use a custom writer, simply specify the path to the lua script in place of the output format. For example:

pandoc -t data/sample.lua

Creating a custom writer requires writing a lua function for each possible element in a pandoc document. To get a documented example which you can modify according to your needs, do

pandoc –print-default-data-file sample.lua

# Authors

Â© 2006-2013 John MacFarlane (jgm at berkeley dot edu). Released under the [GPL](http://www.gnu.org/copyleft/gpl.html), version 2 or greater. This software carries no warranty of any kind. (See COPYRIGHT for full copyright and warranty notices.) Other contributors include Recai OktaÅŸ, Paulo Tanimoto, Peter Wang, Andrea Rossato, Eric Kow, infinity0x, Luke Plant, shreevatsa.public, Puneeth Chaganti, Paul Rivier, rodja.trappe, Bradley Kuhn, thsutton, Nathan Gass, Jonathan Daugherty, JÃ©rÃ©my Bobbio, Justin Bogner, qerub, Christopher Sawicki, Kelsey Hightower, Masayoshi Takahashi, Antoine Latter, Ralf Stephan, Eric Seidel, B. Scott Michel, Gavin Beatty, Sergey Astanin, Arlo O’Keeffe, Denis Laxalde, Brent Yorgey, David Lazar, Jamie F. Olson, Matthew Pickering, Albert Krewinkel, mb21, Jesse Rosenthal.

1. The point of this rule is to ensure that normal paragraphs starting with people’s initials, like

   B. Russell was an English philosopher. [↑](#footnote-ref-56)
2. I have also been influenced by the suggestions of [David Wheeler](http://www.justatheory.com/computers/markup/modest-markdown-proposal.html). [↑](#footnote-ref-58)
3. This scheme is due to Michel Fortin, who proposed it on the [Markdown discussion list](http://six.pairlist.net/pipermail/markdown-discuss/2005-March/001097.html). [↑](#footnote-ref-66)
4. This feature is not yet implemented for RTF, OpenDocument, or ODT. In those formats, you’ll just get an image in a paragraph by itself, with no caption. [↑](#footnote-ref-101)
5. Here is the footnote. [↑](#footnote-ref-103)
6. Here’s one with multiple blocks.

   Subsequent paragraphs are indented to show that they belong to the previous footnote.

   { some.code }

   The whole paragraph can be indented, or just the first line. In this way, multi-paragraph footnotes work like multi-paragraph list items. [↑](#footnote-ref-104)
7. Inlines notes are easier to write, since you don’t have to pick an identifier and move down to type the note. [↑](#footnote-ref-105)