Pandoc - Pandoc User's Guide

Originalartikel

Backup

<html>

```
pandoc
```

[options] [input-file]… Pandoc is a Haskell library for converting from one markup format to another, and a command-line tool that uses this library. It can read Markdown, CommonMark, PHP Markdown Extra, GitHub-Flavored Markdown, MultiMarkdown, and (subsets of) Textile, reStructuredText, HTML, LaTeX, MediaWiki markup, TWiki markup, Haddock markup, OPML, Emacs Org mode, DocBook, txt2tags, EPUB, ODT and Word docx; and it can write plain text, Markdown, CommonMark, PHP Markdown Extra, GitHub-Flavored Markdown, MultiMarkdown, reStructuredText, XHTML, HTML5, LaTeX (including <a href=„https://ctan.org/pkg/beamer“>

```
beamer
```

 slide shows), ConTeXt, RTF, OPML, DocBook, OpenDocument, ODT, Word docx, GNU Texinfo, MediaWiki markup, DokuWiki markup, <a

href=„http://zim-wiki.org/manual/Help/Wiki_Syntax.html“>ZimWiki markup, Haddock markup, EPUB (v2 or v3), FictionBook2, Textile, groff man pages, Emacs Org mode, AsciiDoc, InDesign ICML, TEI Simple, and Slidy, Slideous, DZSlides, reveal.js or S5 HTML slide shows. It can also produce PDF output on systems where LaTeX, ConTeXt, or

```
wkhtmltopdf
```

is installed. Pandoc’s enhanced version of Markdown includes syntax for footnotes, tables, flexible ordered lists, definition lists, fenced code blocks, superscripts and subscripts, strikeout, metadata blocks, automatic tables of contents, embedded LaTeX math, citations, and Markdown inside HTML block elements. (These enhancements, described further under Pandoc’s Markdown, can be disabled using the

```
markdown_strict
```

input or output format.) In contrast to most existing tools for converting Markdown to HTML, which use regex substitutions, pandoc has a modular design: it consists of a set of readers, which parse text in a given format and produce a native representation of the document, and a set of writers, which convert this native representation into a target format. Thus, adding an input or output format requires only adding a reader or writer. Because pandoc’s intermediate representation of a document is less expressive than many of the formats it converts between, one should not expect perfect conversions between every format and every other. Pandoc attempts to preserve the structural elements of a document, but not formatting details such as margin size. And some document elements, such as complex tables, may not fit into pandoc’s simple document model. While conversions from pandoc’s Markdown to all formats aspire to be perfect, conversions from formats more expressive than pandoc’s Markdown can be expected to be lossy. <h2 id=„using-pandoc“>Using

```
pandoc
```

</h2> If no input-file is specified, input is read from stdin. Otherwise, the input-files are concatenated (with a blank line between each) and used as input. Output goes to stdout by default (though output to stdout is disabled for the

```
odt
```

,

```
docx
```

,

```
epub
```

, and

```
epub3
```

output formats). For output to a file, use the

```
-o
```

option: <pre>pandoc -o output.html input.txt</pre> By default, pandoc produces a document fragment, not a standalone document with a proper header and footer. To produce a standalone document, use the

```
-s
```

or

```
--standalone
```

flag: <pre>pandoc -s -o output.html input.txt</pre> For more information on how standalone documents are produced, see Templates, below. Instead of a file, an absolute URI may be given. In this case pandoc will fetch the content using HTTP: <pre>pandoc -f html -t markdown http://www.fsf.org</pre> If multiple input files are given,

```
pandoc
```

will concatenate them all (with blank lines between them) before parsing. This feature is disabled for binary input formats such as

```
EPUB
```

,

```
odt
```

, and

```
docx
```

. The format of the input and output can be specified explicitly using command-line options. The input format can be specified using the

```
-r/--read
```

or

```
-f/--from
```

options, the output format using the

```
-w/--write
```

or

```
-t/--to
```

options. Thus, to convert

```
hello.txt
```

from Markdown to LaTeX, you could type: <pre>pandoc -f markdown -t latex hello.txt</pre> To convert

```
hello.html
```

from HTML to Markdown: <pre>pandoc -f html -t markdown hello.html</pre> Supported output formats are listed below under the

```
-t/--to
```

option. Supported input formats are listed below under the

```
-f/--from
```

option. Note that the

```
rst
```

,

```
textile
```

,

```
latex
```

, and

```
html
```

readers are not complete; there are some constructs that they do not parse. If the input or output format is not specified explicitly,

```
pandoc
```

will attempt to guess it from the extensions of the input and output filenames. Thus, for example, <pre>pandoc -o hello.tex hello.txt</pre> will convert

```
hello.txt
```

from Markdown to LaTeX. If no output file is specified (so that output goes to stdout), or if the output file’s extension is unknown, the output format will default to HTML. If no input file is specified (so that input comes from stdin), or if the input files’ extensions are unknown, the input format will be assumed to be Markdown unless explicitly specified. Pandoc uses the UTF-8 character encoding for both input and output. If your local character encoding is not UTF-8, you should pipe input and output through <a href=„http://www.gnu.org/software/libiconv/“>

```
iconv
```

: <pre>iconv -t utf-8 input.txt | pandoc | iconv -f utf-8</pre> Note that in some output formats (such as HTML, LaTeX, ConTeXt, RTF, OPML, DocBook, and Texinfo), information about the character encoding is included in the document header, which will only be included if you use the

```
-s/--standalone
```

option. <h2 id=„creating-a-pdf“>Creating a PDF</h2> To produce a PDF, specify an output file with a

```
.pdf
```

extension. By default, pandoc will use LaTeX to convert it to PDF: <pre>pandoc test.txt -o test.pdf</pre> Production of a PDF requires that a LaTeX engine be installed (see

```
--latex-engine
```

, below), and assumes that the following LaTeX packages are available: <a href=„https://ctan.org/pkg/amsfonts“>

```
amsfonts
```

, <a href=„https://ctan.org/pkg/amsmath“>

```
amsmath
```

, <a href=„https://ctan.org/pkg/lm“>

```
lm
```

, <a href=„https://ctan.org/pkg/ifxetex“>

```
ifxetex
```

, <a href=„https://ctan.org/pkg/ifluatex“>

```
ifluatex
```

, <a href=„https://ctan.org/pkg/eurosym“>

```
eurosym
```

, <a href=„https://ctan.org/pkg/listings“>

```
listings
```

 (if the

```
--listings
```

option is used), <a href=„https://ctan.org/pkg/fancyvrb“>

```
fancyvrb
```

, <a href=„https://ctan.org/pkg/longtable“>

```
longtable
```

, <a href=„https://ctan.org/pkg/booktabs“>

```
booktabs
```

, <a href=„https://ctan.org/pkg/graphicx“>

```
graphicx
```

 and <a href=„https://ctan.org/pkg/grffile“>

```
grffile
```

 (if the document contains images),

`hyperref`

,

`ulem`

,

`geometry`

 (with the

`geometry`

variable set),

`setspace`

 (with

`linestretch`

), and

`babel`

 (with

`lang`

). The use of

`xelatex`

or

`lualatex`

as the LaTeX engine requires

`fontspec`

;

`xelatex`

uses

`mathspec`

, <a href=„https://ctan.org/pkg/polyglossia“>

```
polyglossia
```

 (with

```
lang
```

), <a href=„https://ctan.org/pkg/xecjk“>

```
xecjk
```

, and <a href=„https://ctan.org/pkg/bidi“>

```
bidi
```

 (with the

```
dir
```

variable set). The <a href=„https://ctan.org/pkg/upquote“>

```
upquote
```

 and <a href=„https://ctan.org/pkg/microtype“>

```
microtype
```

 packages are used if available, and <a href=„https://ctan.org/pkg/csquotes“>

```
csquotes
```

 will be used for smart punctuation if added to the template or included in any header file. The <a href=„https://ctan.org/pkg/natbib“>

```
natbib
```

, <a href=„https://ctan.org/pkg/biblatex“>

```
biblatex
```

, <a href=„https://ctan.org/pkg/bibtex“>

```
bibtex
```

, and <a href=„https://ctan.org/pkg/biber“>

```
biber
```

 packages can optionally be used for citation rendering. These are included with all recent versions of TeX Live. Alternatively, pandoc can use ConTeXt or

```
wkhtmltopdf
```

to create a PDF. To do this, specify an output file with a

```
.pdf
```

extension, as before, but add

```
-t context
```

or

```
-t html5
```

to the command line. PDF output can be controlled using variables for LaTeX (if LaTeX is used) and variables for ConTeXt (if ConTeXt is used). If

```
wkhtmltopdf
```

is used, then the variables

```
margin-left
```

,

```
margin-right
```

,

```
margin-top
```

,

```
margin-bottom
```

, and

```
papersize
```

will affect the output, as will

```
--css
```

. <h2 id=„general-options“>General options</h2> <dl readability=„63“><dt>

```
-f
```

FORMAT,

```
-r
```

FORMAT,

```
--from=
```

FORMAT,

```
--read=
```

FORMAT</dt> <dd readability=„38“> Specify input format. FORMAT can be

```
native
```

(native Haskell),

```
json
```

(JSON version of native AST),

```
markdown
```

(pandoc’s extended Markdown),

```
markdown_strict
```

(original unextended Markdown),

```
markdown_phpextra
```

(PHP Markdown Extra),

```
markdown_github
```

(GitHub-Flavored Markdown),

```
markdown_mmd
```

(MultiMarkdown),

```
commonmark
```

(CommonMark Markdown),

`textile`

(Textile),

`rst`

(reStructuredText),

`html`

(HTML),

`docbook`

(DocBook),

`t2t`

(txt2tags),

`docx`

(docx),

`odt`

(ODT),

`epub`

(EPUB),

`opml`

(OPML),

`org`

(Emacs Org mode),

`mediawiki`

(MediaWiki markup),

`twiki`

(TWiki markup),

```
haddock
```

(Haddock markup), or

```
latex
```

(LaTeX). If

```
+lhs
```

is appended to

```
markdown
```

,

```
rst
```

,

```
latex
```

, or

```
html
```

, the input will be treated as literate Haskell source: see Literate Haskell support, below. Markdown syntax extensions can be individually enabled or disabled by appending

```
+EXTENSION
```

or

```
-EXTENSION
```

to the format name. So, for example,

```
markdown_strict+footnotes+definition_lists
```

is strict Markdown with footnotes and definition lists enabled, and

```
markdown-pipe_tables+hard_line_breaks
```

is pandoc’s Markdown without pipe tables and with hard line breaks. See Pandoc’s Markdown, below, for a list of extensions and their names. See

```
--list-input-formats
```

and

```
--list-extensions
```

, below. </dd> <dt>

```
-t
```

FORMAT,

```
-w
```

FORMAT,

```
--to=
```

FORMAT,

```
--write=
```

FORMAT</dt> <dd readability=„59“> Specify output format. FORMAT can be

```
native
```

(native Haskell),

```
json
```

(JSON version of native AST),

```
plain
```

(plain text),

```
markdown
```

(pandoc’s extended Markdown),

```
markdown_strict
```

(original unextended Markdown),

```
markdown_phpextra
```

(PHP Markdown Extra),

```
markdown_github
```

(GitHub-Flavored Markdown),

```
markdown_mmd
```

(MultiMarkdown),

```
commonmark
```

(CommonMark Markdown),

```
rst
```

(reStructuredText),

```
html
```

(XHTML),

```
html5
```

(HTML5),

```
latex
```

(LaTeX),

```
beamer
```

(LaTeX beamer slide show),

```
context
```

(ConTeXt),

```
man
```

(groff man),

```
mediawiki
```

(MediaWiki markup),

```
dokuwiki
```

(DokuWiki markup),

```
zimwiki
```

(ZimWiki markup),

```
textile
```

(Textile),

```
org
```

(Emacs Org mode),

```
texinfo
```

(GNU Texinfo),

```
opml
```

(OPML),

```
docbook
```

(DocBook 4),

```
docbook5
```

(DocBook 5),

```
opendocument
```

(OpenDocument),

```
odt
```

(OpenOffice text document),

```
docx
```

(Word docx),

```
haddock
```

(Haddock markup),

```
rtf
```

(rich text format),

```
epub
```

(EPUB v2 book),

```
epub3
```

(EPUB v3),

```
fb2
```

(FictionBook2 e-book),

```
asciidoc
```

(AsciiDoc),

```
icml
```

(InDesign ICML),

```
tei
```

(TEI Simple),

```
slidy
```

(Slidy HTML and JavaScript slide show),

```
slideous
```

(Slideous HTML and JavaScript slide show),

```
dzslides
```

(DZSlides HTML5 + JavaScript slide show),

```
revealjs
```

(reveal.js HTML5 + JavaScript slide show),

```
s5
```

(S5 HTML and JavaScript slide show), or the path of a custom lua writer (see Custom writers, below). Note that

```
odt
```

,

```
epub
```

, and

```
epub3
```

output will not be directed to stdout; an output filename must be specified using the

```
-o/--output
```

option. If

```
+lhs
```

is appended to

```
markdown
```

,

```
rst
```

,

```
latex
```

,

```
beamer
```

,

```
html
```

, or

```
html5
```

, the output will be rendered as literate Haskell source: see Literate Haskell support, below. Markdown syntax extensions can be individually enabled or disabled by appending

```
+EXTENSION
```

or

```
-EXTENSION
```

to the format name, as described above under

```
-f
```

. See

```
--list-output-formats
```

and

```
--list-extensions
```

, below. </dd> <dt>

```
-o
```

FILE,

```
--output=
```

FILE</dt> <dd readability=„8“> Write output to FILE instead of stdout. If FILE is

```
-
```

, output will go to stdout. (Exception: if the output format is

```
odt
```

,

```
docx
```

,

```
epub
```

, or

```
epub3
```

, output to stdout is disabled.) </dd> <dt>

```
--data-dir=
```

DIRECTORY</dt> <dd readability=„18“> Specify the user data directory to search for pandoc data files. If this option is not specified, the default user data directory will be used. This is, in Unix: <pre>$HOME/.pandoc</pre> in Windows XP: <pre>C:\Documents And Settings\USERNAME\Application Data\pandoc</pre> and in Windows Vista or later: <pre>C:\Users\USERNAME\AppData\Roaming\pandoc</pre> You can find the default user data directory on your system by looking at the output of

```
pandoc --version
```

. A

```
reference.odt
```

,

```
reference.docx
```

,

```
epub.css
```

,

```
templates
```

,

```
slidy
```

,

```
slideous
```

, or

```
s5
```

directory placed in this directory will override pandoc’s normal defaults. </dd> <dt>

```
--bash-completion
```

</dt> <dd readability=„3“> Generate a bash completion script. To enable bash completion with pandoc, add this to your

```
.bashrc
```

: <pre> eval „$(pandoc –bash-completion)“</pre> </dd> <dt>

```
--verbose
```

</dt> <dd readability=„1“> Give verbose debugging output. Currently this only has an effect with PDF output. </dd> <dt>

```
--list-input-formats
```

</dt> <dd readability=„2“> List supported input formats, one per line. </dd> <dt>

```
--list-output-formats
```

</dt> <dd readability=„2“> List supported output formats, one per line. </dd> <dt>

```
--list-extensions
```

</dt> <dd readability=„5“> List supported Markdown extensions, one per line, followed by a

```
+
```

or

```
-
```

indicating whether it is enabled by default in pandoc’s Markdown. </dd> <dt>

```
--list-highlight-languages
```

</dt> <dd readability=„2“> List supported languages for syntax highlighting, one per line. </dd> <dt>

```
--list-highlight-styles
```

</dt> <dd readability=„2“> List supported styles for syntax highlighting, one per line. See

```
--highlight-style
```

. </dd> <dt>

```
-v
```

,

```
--version
```

</dt> <dd> Print version. </dd> <dt>

```
-h
```

,

```
--help
```

</dt> <dd> Show usage message. </dd> </dl><h2 id=„reader-options“>Reader options</h2> <dl readability=„55“><dt>

```
-R
```

,

```
--parse-raw
```

</dt> <dd readability=„21“> Parse untranslatable HTML codes and LaTeX environments as raw HTML or LaTeX, instead of ignoring them. Affects only HTML and LaTeX input. Raw HTML can be printed in Markdown, reStructuredText, Emacs Org mode, HTML, Slidy, Slideous, DZSlides, reveal.js, and S5 output; raw LaTeX can be printed in Markdown, reStructuredText, Emacs Org mode, LaTeX, and ConTeXt output. The default is for the readers to omit untranslatable HTML codes and LaTeX environments. (The LaTeX reader does pass through untranslatable LaTeX commands, even if

```
-R
```

is not specified.) </dd> <dt>

```
-S
```

,

```
--smart
```

</dt> <dd readability=„12“> 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: This option is selected automatically when the output format is

```
latex
```

or

```
context
```

, unless

```
--no-tex-ligatures
```

is used. It has no effect for

```
latex
```

input.) </dd> <dt>

```
--old-dashes
```

</dt> <dd readability=„4“> Selects the pandoc <= 1.8.2.1 behavior for parsing smart dashes:

```
-
```

before a numeral is an en-dash, and

```
--
```

is an em-dash. This option is selected automatically for

```
textile
```

input. </dd> <dt>

```
--base-header-level=
```

NUMBER</dt> <dd readability=„1“> Specify the base level for headers (defaults to 1). </dd> <dt>

```
--indented-code-classes=
```

CLASSES</dt> <dd readability=„5“> Specify classes to use for indented code blocks–for example,

```
perl,numberLines
```

or

```
haskell
```

. Multiple classes may be separated by spaces or commas. </dd> <dt>

```
--default-image-extension=
```

EXTENSION</dt> <dd readability=„4“> Specify a default extension to use when image paths/URLs have no extension. This allows you to use the same source for formats that require different kinds of images. Currently this option only affects the Markdown and LaTeX readers. </dd> <dt>

```
--file-scope
```

</dt> <dd readability=„8“> Parse each file individually before combining for multifile documents. This will allow footnotes in different files with the same identifiers to work as expected. If this option is set, footnotes and links will not work across files. Reading binary files (docx, odt, epub) implies

```
--file-scope
```

. </dd> <dt>

```
--filter=
```

PROGRAM</dt> <dd readability=„21“> Specify an executable to be used as a filter transforming the pandoc AST after the input is parsed and before the output is written. The executable should read JSON from stdin and write JSON to stdout. The JSON must be formatted like pandoc’s own JSON input and output. The name of the output format will be passed to the filter as the first argument. Hence, <pre>pandoc –filter ./caps.py -t latex</pre> is equivalent to <pre>pandoc -t json | ./caps.py latex | pandoc -f json -t latex</pre> The latter form may be useful for debugging filters. Filters may be written in any language.

```
Text.Pandoc.JSON
```

exports

```
toJSONFilter
```

to facilitate writing filters in Haskell. Those who would prefer to write filters in python can use the module <a href=„https://github.com/jgm/pandocfilters“>

```
pandocfilters
```

, installable from PyPI. There are also pandoc filter libraries in PHP, perl, and javascript/node.js. In order of preference, pandoc will look for filters in <ol type=„1“ readability=„1“><li readability=„1“> a specified full or relative path (executable or non-executable) <li readability=„1“>

```
$DATADIR/filters
```

(executable or non-executable) <li readability=„1“>

```
$PATH
```

(executable only) </dd> <dt>

```
-M
```

KEY[

```
=
```

VAL],

```
--metadata=
```

KEY[

```
:
```

VAL]</dt> <dd readability=„10“> Set the metadata field KEY to the value VAL. A value specified on the command line overrides a value specified in the document. Values will be parsed as YAML boolean or string values. If no value is specified, the value will be treated as Boolean true. Like

```
--variable
```

,

```
--metadata
```

causes template variables to be set. But unlike

```
--variable
```

,

```
--metadata
```

affects the metadata of the underlying document (which is accessible from filters and may be printed in some output formats). </dd> <dt>

```
--normalize
```

</dt> <dd readability=„4“> Normalize the document after reading: merge adjacent

```
Str
```

or

```
Emph
```

elements, for example, and remove repeated

```
Space
```

s. </dd> <dt>

```
-p
```

,

```
--preserve-tabs
```

</dt> <dd readability=„3“> Preserve tabs instead of converting them to spaces (the default). Note that this will only affect tabs in literal code spans and code blocks; tabs in regular text will be treated as spaces. </dd> <dt>

```
--tab-stop=
```

NUMBER</dt> <dd readability=„1“> Specify the number of spaces per tab (default is 4). </dd> <dt>

```
--track-changes=accept
```

|

```
reject
```

|

```
all
```

</dt> <dd readability=„20“> Specifies what to do with insertions, deletions, and comments produced by the MS Word “Track Changes” feature.

```
accept
```

(the default), inserts all insertions, and ignores all deletions.

```
reject
```

inserts all deletions and ignores insertions. Both

```
accept
```

and

```
reject
```

ignore comments.

```
all
```

puts in insertions, deletions, and comments, wrapped in spans with

```
insertion
```

,

```
deletion
```

,

```
comment-start
```

, and

```
comment-end
```

classes, respectively. The author and time of change is included.

```
all
```

is useful for scripting: only accepting changes from a certain reviewer, say, or before a certain date. This option only affects the docx reader. </dd> <dt>

```
--extract-media=
```

DIR</dt> <dd readability=„7“> Extract images and other media contained in a docx or epub container to the path DIR, creating it if necessary, and adjust the images references in the document so they point to the extracted files. This option only affects the docx and epub readers. </dd> </dl><h2 id=„general-writer-options“>General writer options</h2> <dl readability=„53“><dt>

```
-s
```

,

```
--standalone
```

</dt> <dd readability=„13“> Produce output with an appropriate header and footer (e.g. a standalone HTML, LaTeX, TEI, or RTF file, not a fragment). This option is set automatically for

```
pdf
```

,

```
epub
```

,

```
epub3
```

,

```
fb2
```

,

```
docx
```

, and

```
odt
```

output. </dd> <dt>

```
--template=
```

FILE</dt> <dd readability=„13“> Use FILE as a custom template for the generated document. Implies

```
--standalone
```

. See Templates, below, for a description of template syntax. If no extension is specified, an extension corresponding to the writer will be added, so that

```
--template=special
```

looks for

```
special.html
```

for HTML output. If the template is not found, pandoc will search for it in the

```
templates
```

subdirectory of the user data directory (see

```
--data-dir
```

). If this option is not used, a default template appropriate for the output format will be used (see

```
-D/--print-default-template
```

). </dd> <dt>

```
-V
```

KEY[

```
=
```

VAL],

```
--variable=
```

KEY[

```
:
```

VAL]</dt> <dd readability=„8“> Set the template variable KEY to the value VAL when rendering the document in standalone mode. This is generally only

useful when the

```
--template
```

option is used to specify a custom template, since pandoc automatically sets the variables used in the default templates. If no VAL is specified, the key will be given the value

```
true
```

. </dd> <dt>

```
-D
```

FORMAT,

```
--print-default-template=
```

FORMAT</dt> <dd readability=„3“> Print the system default template for an output FORMAT. (See

```
-t
```

for a list of possible FORMATs.) Templates in the user data directory are ignored. </dd> <dt>

```
--print-default-data-file=
```

FILE</dt> <dd readability=„1“> Print a system default data file. Files in the user data directory are ignored. </dd> <dt>

```
--dpi
```

=NUMBER</dt> <dd>Specify the dpi (dots per inch) value for conversion from pixels to inch/centimeters and vice versa. The default is 96dpi. Technically, the correct term would be ppi (pixels per inch). </dd> <dt>

```
--wrap=auto
```

|

```
none
```

|

```
preserve
```

</dt> <dd readability=„13“> Determine how text is wrapped in the output (the source code, not the rendered version). With

```
auto
```

(the default), pandoc will attempt to wrap lines to the column width specified by

```
--columns
```

(default 72). With

```
none
```

, pandoc will not wrap lines at all. With

```
preserve
```

, pandoc will attempt to preserve the wrapping from the source document (that is, where there are nonsemantic newlines in the source, there will be nonsemantic newlines in the output as well). Automatic wrapping does not currently work in HTML output. </dd> <dt>

```
--no-wrap
```

</dt> <dd readability=„1“> Deprecated synonym for

```
--wrap=none
```

. </dd> <dt>

```
--columns=
```

NUMBER</dt> <dd readability=„3“> Specify length of lines in characters. This affects text wrapping in the generated source code (see

```
--wrap
```

). It also affects calculation of column widths for plain text tables (see Tables below). </dd> <dt>

```
--toc
```

,

```
--table-of-contents
```

</dt> <dd readability=„16“> Include an automatically generated table of contents (or, in the case of

```
latex
```

,

`context`

,

`docx`

, and

`rst`

, an instruction to create one) in the output document. This option has no effect on

`man`

,

`docbook`

,

`docbook5`

,

`slidy`

,

`slideous`

,

`s5`

, or

`odt`

output. </dd> <dt>

`--toc-depth=`

NUMBER</dt> <dd readability=„5“> Specify the number of section levels to include in the table of contents. The default is 3 (which means that level 1, 2, and 3 headers will be listed in the contents). </dd> <dt>

`--no-highlight`

</dt> <dd readability=„2“> Disables syntax highlighting for code blocks and inlines, even when

a language attribute is given. </dd> <dt>

```
--highlight-style=
```

STYLE</dt> <dd readability=„12“> Specifies the coloring style to be used in highlighted source code. Options are

```
pygments
```

(the default),

```
kate
```

,

```
monochrome
```

,

```
breezeDark
```

,

```
espresso
```

,

```
zenburn
```

,

```
haddock
```

, and

```
tango
```

. For more information on syntax highlighting in pandoc, see Syntax highlighting, below. See also

```
--list-highlight-styles
```

. </dd> <dt>

```
-H
```

FILE,

```
--include-in-header=
```

FILE</dt> <dd readability=„9“> Include contents of FILE, verbatim, at the end of the header. This can be used, for example, to include special CSS or JavaScript in HTML documents. This option can be used repeatedly to include multiple files in the header. They will be included in the order specified. Implies

```
--standalone
```

. </dd> <dt>

```
-B
```

FILE,

```
--include-before-body=
```

FILE</dt> <dd readability=„9“> Include contents of FILE, verbatim, at the beginning of the document body (e.g. after the

```
&lt;body&gt;
```

tag in HTML, or the

```
\begin{document}
```

command in LaTeX). This can be used to include navigation bars or banners in HTML documents. This option can be used repeatedly to include multiple files. They will be included in the order specified. Implies

```
--standalone
```

. </dd> <dt>

```
-A
```

FILE,

```
--include-after-body=
```

FILE</dt> <dd readability=„8“> Include contents of FILE, verbatim, at the end of the document body (before the

```
&lt;/body&gt;
```

tag in HTML, or the

```
\end{document}
```

command in LaTeX). This option can be used repeatedly to include multiple files. They will be included in the order specified. Implies

```
--standalone
```

. </dd> </dl><h2 id=„options-affecting-specific-writers“>Options affecting specific writers</h2> <dl readability=„129“><dt>

```
--self-contained
```

</dt> <dd readability=„25“> Produce a standalone HTML file with no external dependencies, using

```
data:
```

URIs to incorporate the contents of linked scripts, stylesheets, images, and videos. The resulting file should be “self-contained,” in the sense that it needs no external files and no net access to be displayed properly by a browser. This option works only with HTML output formats, including

```
html
```

,

```
html5
```

,

```
html+lhs
```

,

```
html5+lhs
```

,

```
s5
```

,

```
slidy
```

,

```
slideous
```

,

```
dzslides
```

, and

```
revealjs
```

. Scripts, images, and stylesheets at absolute URLs will be downloaded; those at relative URLs will be sought relative to the working directory (if the first source file is local) or relative to the base URL (if the first source file is remote). Limitation: resources that are loaded dynamically through JavaScript cannot be incorporated; as a result,

```
--self-contained
```

does not work with

```
--mathjax
```

, and some advanced features (e.g. zoom or speaker notes) may not work in an offline “self-contained”

```
reveal.js
```

slide show. </dd> <dt>

```
--html-q-tags
```

</dt> <dd readability=„1“> Use

```
&lt;q&gt;
```

tags for quotes in HTML. </dd> <dt>

```
--ascii
```

</dt> <dd readability=„3“> Use only ASCII characters in output. Currently supported only for HTML output (which uses numerical entities instead of UTF-8 when this option is selected). </dd> <dt>

```
--reference-links
```

</dt> <dd readability=„6“> Use reference-style links, rather than inline links, in writing Markdown or reStructuredText. By default inline links are used. The placement of link references is affected by the

```
--reference-location
```

option. </dd> <dt>

```
--reference-location = block
```

|

```
section
```

|

```
document
```

</dt> <dd readability=„7“> Specify whether footnotes (and references, if

```
reference-links
```

is set) are placed at the end of the current (top-level) block, the current section, or the document. The default is

```
document
```

. Currently only affects the markdown writer. </dd> <dt>

```
--atx-headers
```

</dt> <dd readability=„4“> Use ATX-style headers in Markdown and AsciiDoc output. The default is to use setext-style headers for levels 1-2, and then ATX headers. </dd> <dt>

```
--chapters
```

</dt> <dd readability=„1“> Deprecated synonym for

```
--top-level-division=chapter
```

. </dd> <dt>

```
--top-level-division=[default|section|chapter|part]
```

</dt> <dd readability=„18“> Treat top-level headers as the given division type in LaTeX, ConTeXt, DocBook, and TEI output. The hierarchy order is part, chapter, then section; all headers are shifted such that the top-level header becomes the specified type. The default behavior is to determine the best division type via heuristics: unless other conditions apply,

```
section
```

is chosen. When the LaTeX document class is set to

```
report
```

,

```
book
```

, or

```
memoir
```

(unless the

```
article
```

option is specified),

```
chapter
```

is implied as the setting for this option. If

```
beamer
```

is the output format, specifying either

```
chapter
```

or

```
part
```

will cause top-level headers to become

```
\part{..}
```

, while second-level headers remain as their default type. </dd> <dt>

```
-N
```

,

```
--number-sections
```

</dt> <dd readability=„8“> Number section headings in LaTeX, ConTeXt, HTML, or EPUB output. By default, sections are not numbered. Sections with class

```
unnumbered
```

will never be numbered, even if

```
--number-sections
```

is specified. </dd> <dt>

```
--number-offset=
```

NUMBER[

```
,
```

NUMBER

```
,
```

…]</dt> <dd readability=„14“> Offset for section headings in HTML output (ignored in other output formats). The first number is added to the section number for top-level headers, the second for second-level headers, and so on. So, for example, if you want the first top-level header in your document to be numbered “6”, specify

```
--number-offset=5
```

. If your document starts with a level-2 header which you want to be numbered “1.5”, specify

```
--number-offset=1,4
```

. Offsets are 0 by default. Implies

```
--number-sections
```

. </dd> <dt>

```
--no-tex-ligatures
```

</dt> <dd readability=„22“> Do not use the TeX ligatures for quotation marks, apostrophes, and dashes (

```
`...'
```

,

```
``..''
```

,

```
--
```

,

```
---
```

) when writing or reading LaTeX or ConTeXt. In reading LaTeX, parse the characters

```
`
```

,

```
'
```

, and

```
-
```

literally, rather than parsing ligatures for quotation marks and dashes. In writing LaTeX or ConTeXt, print unicode quotation mark and dash characters literally, rather than converting them to the standard ASCII TeX ligatures. Note: normally

```
--smart
```

is selected automatically for LaTeX and ConTeXt output, but it must be specified explicitly if

```
--no-tex-ligatures
```

is selected. If you use literal curly quotes, dashes, and ellipses in your source, then you may want to use

```
--no-tex-ligatures
```

without

```
--smart
```

. </dd> <dt>

```
--listings
```

</dt> <dd readability=„1“> Use the <a href=„https://ctan.org/pkg/listings“>

```
listings
```

 package for LaTeX code blocks </dd> <dt>

```
-i
```

,

```
--incremental
```

</dt> <dd readability=„3“> Make list items in slide shows display incrementally (one by one). The default is for lists to be displayed all at once. </dd> <dt>

```
--slide-level=
```

NUMBER</dt> <dd readability=„9“> Specifies that headers with the specified level create slides (for

```
beamer
```

,

```
s5
```

,

```
slidy
```

,

```
slideous
```

,

```
dzslides
```

). Headers above this level in the hierarchy are used to divide the slide show into sections; headers below this level create subheads within a slide. The default is to set the slide level based on the contents of the document; see Structuring the slide show. </dd> <dt>

```
--section-divs
```

</dt> <dd readability=„5“> Wrap sections in

```
&lt;div&gt;
```

tags (or

```
&lt;section&gt;
```

tags in HTML5), and attach identifiers to the enclosing

```
&lt;div&gt;
```

(or

```
&lt;section&gt;
```

) rather than the header itself. See Header identifiers, below. </dd> <dt>

```
--email-obfuscation=none
```

|

```
javascript
```

|

```
references
```

</dt> <dd readability=„4“> Specify a method for obfuscating

```
mailto:
```

links in HTML documents.

```
none
```

leaves

```
mailto:
```

links as they are.

```
javascript
```

obfuscates them using JavaScript.

```
references
```

obfuscates them by printing their letters as decimal or hexadecimal character references. The default is

```
none
```

. </dd> <dt>

```
--id-prefix=
```

STRING</dt> <dd readability=„5“> Specify a prefix to be added to all automatically generated identifiers in HTML and DocBook output, and to footnote numbers in Markdown output. This is useful for preventing duplicate identifiers when generating fragments to be included in other pages. </dd> <dt>

```
-T
```

STRING,

```
--title-prefix=
```

STRING</dt> <dd readability=„3“> Specify STRING as a prefix at the beginning of the title that appears in the HTML header (but not in the title as it appears at the beginning of the HTML body). Implies

```
--standalone
```

. </dd> <dt>

```
-c
```

URL,

```
--css=
```

URL</dt> <dd readability=„3“> Link to a CSS style sheet. This option can be used repeatedly to include multiple files. They will be included in the order specified. </dd> <dt>

```
--reference-odt=
```

FILE</dt> <dd readability=„18“> Use the specified file as a style reference in producing an ODT. For best results, the reference ODT should be a modified version of an ODT produced using pandoc. The contents of the reference ODT are ignored, but its stylesheets are used in the new ODT. If no reference ODT is specified on the command line, pandoc will look for a file

```
reference.odt
```

in the user data directory (see

```
--data-dir
```

). If this is not found either, sensible defaults will be used. To produce a custom

```
reference.odt
```

, first get a copy of the default

```
reference.odt
```

:

```
pandoc --print-default-data-file reference.odt &gt; custom-reference.odt
```

. Then open

```
custom-reference.docx
```

in LibreOffice, modify the styles as you wish, and save the file. </dd> <dt>

```
--reference-docx=
```

FILE</dt> <dd readability=„54“> Use the specified file as a style reference in producing a docx file. For best results, the reference docx should be a modified version of a docx file produced using pandoc. The contents of the reference docx are ignored, but its stylesheets and document properties (including margins, page size, header, and footer) are used in the new docx. If no reference docx is specified on the command line, pandoc will look for a file

```
reference.docx
```

in the user data directory (see

```
--data-dir
```

). If this is not found either, sensible defaults will be used. To produce a custom

```
reference.docx
```

, first get a copy of the default

```
reference.docx
```

:

```
pandoc --print-default-data-file reference.docx &gt; custom-reference.docx
```

. Then open

```
custom-reference.docx
```

in Word, modify the styles as you wish, and save the file. For best results, do not make changes to this file other than modifying the styles used by pandoc: [paragraph] Normal, Body Text, First Paragraph, Compact, Title, Subtitle, Author, Date, Abstract, Bibliography, Heading 1, Heading 2, Heading 3, Heading 4, Heading 5, Heading 6, Block Text, Footnote Text, Definition Term, Definition, Caption, Table Caption, Image Caption, Figure, Figure With Caption, TOC Heading; [character] Default Paragraph Font, Body Text Char, Verbatim Char, Footnote Reference, Hyperlink; [table] Normal Table. </dd> <dt>

```
--epub-stylesheet=
```

FILE</dt> <dd readability=„6“> Use the specified CSS file to style the EPUB. If no stylesheet is specified, pandoc will look for a file

```
epub.css
```

in the user data directory (see

```
--data-dir
```

). If it is not found there, sensible defaults will be used. </dd> <dt>

```
--epub-cover-image=
```

FILE</dt> <dd readability=„6“> Use the specified image as the EPUB cover. It is recommended that the image be less than 1000px in width and height. Note that in a Markdown source document you can also specify

```
cover-image
```

in a YAML metadata block (see EPUB Metadata, below). </dd> <dt>

```
--epub-metadata=
```

FILE</dt> <dd readability=„22“> Look in the specified XML file for metadata for the EPUB. The file should contain a series of Dublin Core elements. For example: <pre> <dc:rights>Creative Commons</dc:rights> <dc:language>es-AR</dc:language></pre> By default, pandoc will include the following metadata elements:

```
&lt;dc:title&gt;
```

(from the document title),

```
&lt;dc:creator&gt;
```

(from the document authors),

```
&lt;dc:date&gt;
```

(from the document date, which should be in ISO 8601 format),

```
&lt;dc:language&gt;
```

(from the

```
lang
```

variable, or, if is not set, the locale), and

```
&lt;dc:identifier id="BookId"&gt;
```

(a randomly generated UUID). Any of these may be overridden by elements in the metadata file. Note: if the source document is Markdown, a YAML metadata block in the document can be used instead. See below under EPUB Metadata. </dd> <dt>

```
--epub-embed-font=
```

FILE</dt> <dd readability=„17“> Embed the specified font in the EPUB. This option can be repeated to embed multiple fonts. Wildcards can also be used: for example,

```
DejaVuSans-*.ttf
```

. However, if you use wildcards on the command line, be sure to escape them or put the whole

filename in single quotes, to prevent them from being interpreted by the shell. To use the embedded fonts, you will need to add declarations like the following to your CSS (see

```
--epub-stylesheet
```

): <pre>@font-face { font-family: DejaVuSans; font-style: normal; font-weight: normal; src:url(„DejaVuSans-Regular.ttf“); } @font-face { font-family: DejaVuSans; font-style: normal; font-weight: bold; src:url(„DejaVuSans-Bold.ttf“); } @font-face { font-family: DejaVuSans; font-style: italic; font-weight: normal; src:url(„DejaVuSans-Oblique.ttf“); } @font-face { font-family: DejaVuSans; font-style: italic; font-weight: bold; src:url(„DejaVuSans-BoldOblique.ttf“); } body { font-family: „DejaVuSans“; }</pre> </dd> <dt>

```
--epub-chapter-level=
```

NUMBER</dt> <dd readability=„10“> Specify the header level at which to split the EPUB into separate “chapter” files. The default is to split into chapters at level 1 headers. This option only affects the internal composition of the EPUB, not the way chapters and sections are displayed to users. Some readers may be slow if the chapter files are too large, so for large documents with few level 1 headers, one might want to use a chapter level of 2 or 3. </dd> <dt>

```
--latex-engine=pdflatex
```

|

```
lualatex
```

|

```
xelatex
```

</dt> <dd readability=„4“> Use the specified LaTeX engine when producing PDF output. The default is

```
pdflatex
```

. If the engine is not in your PATH, the full path of the engine may be specified here. </dd> <dt>

```
--latex-engine-opt=
```

STRING</dt> <dd readability=„4“> Use the given string as a command-line argument to the

```
latex-engine
```

. If used multiple times, the arguments are provided with spaces between them. Note that no check for duplicate options is done. </dd> </dl><h2 id=„citation-rendering“>Citation rendering</h2> <dl readability=„13“><dt>

```
--bibliography=
```

FILE</dt> <dd readability=„12“> Set the

```
bibliography
```

field in the document’s metadata to FILE, overriding any value set in the metadata, and process citations using

```
pandoc-citeproc
```

. (This is equivalent to

```
--metadata bibliography=FILE --filter pandoc-citeproc
```

.) If

```
--natbib
```

or

```
--biblatex
```

is also supplied,

```
pandoc-citeproc
```

is not used, making this equivalent to

```
--metadata bibliography=FILE
```

. If you supply this argument multiple times, each FILE will be added to bibliography. </dd> <dt>

```
--csl=
```

FILE</dt> <dd readability=„4“> Set the

```
csl
```

field in the document’s metadata to FILE, overriding any value set in the metadata. (This is equivalent to

```
--metadata csl=FILE
```

.) This option is only relevant with

```
pandoc-citeproc
```

. </dd> <dt>

```
--citation-abbreviations=
```

FILE</dt> <dd readability=„5“> Set the

```
citation-abbreviations
```

field in the document’s metadata to FILE, overriding any value set in the metadata. (This is equivalent to

```
--metadata citation-abbreviations=FILE
```

.) This option is only relevant with

```
pandoc-citeproc
```

. </dd> <dt>

```
--natbib
```

</dt> <dd readability=„4“> Use <a href=„https://ctan.org/pkg/natbib“>

```
natbib
```

 for citations in LaTeX output. This option is not for use with the

```
pandoc-citeproc
```

filter or with PDF output. It is intended for use in producing a LaTeX file that can be processed with <a href=„https://ctan.org/pkg/bibtex“>

```
bibtex
```

. </dd> <dt>

```
--biblatex
```

</dt> <dd readability=„4“> Use <a href=„https://ctan.org/pkg/biblatex“>

```
biblatex
```

 for citations in LaTeX output. This option is not for use with the

```
pandoc-citeproc
```

filter or with PDF output. It is intended for use in producing a LaTeX file that can be processed with <a href=„https://ctan.org/pkg/bibtex“>

```
bibtex
```

 or <a href=„https://ctan.org/pkg/biber“>

```
biber
```

. </dd> </dl><h2 id=„math-rendering-in-html“>Math rendering in HTML</h2> <dl readability=„26“><dt>

```
-m
```

[URL],

```
--latexmathml
```

[

```
=
```

URL]</dt> <dd readability=„12“> Use the LaTeXMathML script to display embedded TeX math in HTML output. To insert a link to a local copy of the

```
LaTeXMathML.js
```

script, provide a URL. If no URL is provided, the contents of the script will be inserted directly into the HTML header, preserving portability at the price of efficiency. If you plan to use math on several pages, it is much better to link to a copy of the script, so it can be cached. </dd> <dt>

```
--mathml
```

[

```
=
```

URL]</dt> <dd readability=„8“> Convert TeX math to MathML (in

```
docbook
```

,

```
docbook5
```

,

```
html
```

and

```
html5
```

). In standalone

```
html
```

output, a small JavaScript (or a link to such a script if a URL is supplied) will be inserted that allows the MathML to be viewed on some browsers. </dd> <dt>

```
--jsmath
```

[

```
=
```

URL]</dt> <dd readability=„9“> Use jsMath to display embedded TeX math in HTML output. The URL should point to the jsMath load script (e.g.

```
jsMath/easy/load.js
```

); if provided, it will be linked to in the header of standalone HTML documents. If a URL is not provided, no link to the jsMath load script will be inserted; it is then up to the author to provide such a link in the HTML template. </dd> <dt>

```
--mathjax
```

[

```
=
```

URL]</dt> <dd readability=„4“> Use MathJax to display embedded TeX math in HTML output. The URL should point to the

```
MathJax.js
```

load script. If a URL is not provided, a link to the MathJax CDN will be inserted. </dd> <dt>

```
--gladtex
```

</dt> <dd readability=„3“> Enclose TeX math in

```
&lt;eq&gt;
```

tags in HTML output. These can then be processed by <a

href=„http://ans.hsh.no/home/mgg/gladtex/“>gladTeX to produce links to images of the typeset formulas. </dd> <dt>

```
--mimetex
```

[

```
=
```

URL]</dt> <dd readability=„4“> Render TeX math using the mimeTeX CGI script. If URL is not specified, it is assumed that the script is at

```
/cgi-bin/mimetex.cgi
```

. </dd> <dt>

```
--webtex
```

[

```
=
```

URL]</dt> <dd readability=„8“> Render TeX formulas using an external script that converts TeX formulas to images. The formula will be concatenated with the URL provided. If URL is not specified, the CodeCogs will be used. Note: the

```
--webtex
```

option will affect Markdown output as well as HTML, which is useful if you’re targeting a version of Markdown without native math support. </dd> <dt>

```
--katex
```

[

```
=
```

URL]</dt> <dd readability=„6“> Use KaTeX to display embedded TeX math in HTML output. The URL should point to the

```
katex.js
```

load script. If a URL is not provided, a link to the KaTeX CDN will be inserted. Note: KaTeX seems to work best with

```
html5
```

output. </dd> <dt>

```
--katex-stylesheet=
```

URL</dt> <dd readability=„4“> The URL should point to the

```
katex.css
```

stylesheet. If this option is not specified, a link to the KaTeX CDN will be inserted. Note that this option does not imply

```
--katex
```

. </dd> </dl><h2 id=„options-for-wrapper-scripts“>Options for wrapper scripts</h2> <dl readability=„8“><dt>

```
--dump-args
```

</dt> <dd readability=„12“> Print information about command-line arguments to stdout, then exit. This option is intended primarily for use in wrapper scripts. The first line of output contains the name of the output file specified with the

```
-o
```

option, or

```
-
```

(for stdout) if no output file was specified. The remaining lines contain the command-line arguments, one per line, in the order they appear. These do not include regular pandoc options and their arguments, but do include any options appearing after a

```
--
```

separator at the end of the line. </dd> <dt>

```
--ignore-args
```

</dt> <dd readability=„6“> Ignore command-line arguments (for use in wrapper scripts). Regular pandoc options are not ignored. Thus, for example, <pre>pandoc –ignore-args -o foo.html -s foo.txt – -e latin1</pre> is equivalent to <pre>pandoc -o foo.html -s</pre> </dd> </dl> When the

```
-s/--standalone
```

option is used, pandoc uses a template to add header and footer material that is needed for a self-standing document. To see the default template that is used, just type <pre>pandoc -D *FORMAT*</pre> where FORMAT is the name of the output format. A custom template can be specified using the

```
--template
```

option. You can also override the system default templates for a given output format FORMAT by putting a file

```
templates/default.*FORMAT*
```

in the user data directory (see

```
--data-dir
```

, above). Exceptions:</p> For

```
odt
```

output, customize the

```
default.opendocument
```

template. For

```
pdf
```

output, customize the

```
default.latex
```

template (or the

```
default.beamer
```

template, if you use

```
-t beamer
```

, or the

```
default.context
```

template, if you use

```
-t context
```

).

```
docx
```

has no template (however, you can use

```
--reference-docx
```

to customize the output). Templates contain variables, which allow for the inclusion of arbitrary information at any point in the file. Variables may be set within the document using YAML metadata blocks. They may also be set at the command line using the

```
-V/--variable
```

option: variables set in this way override metadata fields with the same name. <h2 id=„variables-set-by-pandoc“>Variables set by pandoc</h2> Some variables are set automatically by pandoc. These vary somewhat depending on the output format, but include metadata fields as well as the following: <dl readability=„3“><dt>

```
title
```

,

```
author
```

,

```
date
```

</dt> <dd readability=„7“> allow identification of basic aspects of the document. Included in PDF metadata through LaTeX and ConTeXt. These can be set through a pandoc title block, which allows for multiple authors, or through a YAML metadata block: <pre>— author: - Aristotle - Peter Abelard …</pre> </dd> <dt>

```
subtitle
```

</dt> <dd>document subtitle, included in HTML, EPUB, LaTeX, ConTeXt, and Word docx; renders in LaTeX only when using a document class that supports

```
\subtitle
```

, such as

```
beamer
```

or the KOMA-Script series (

```
scrartcl
```

,

```
scrreprt
```

,

```
scrbook
```

). </dd> <dt>

```
institute
```

</dt> <dd>author affiliations (in LaTeX and Beamer only). Can be a list, when there are multiple authors. </dd> <dt>

```
abstract
```

</dt> <dd>document summary, included in LaTeX, ConTeXt, AsciiDoc, and Word docx </dd> <dt>

```
keywords
```

</dt> <dd>list of keywords to be included in HTML, PDF, and AsciiDoc metadata; may be repeated as for

```
author
```

, above </dd> <dt>

```
header-includes
```

</dt> <dd>contents specified by

```
-H/--include-in-header
```

(may have multiple values) </dd> <dt>

```
toc
```

</dt> <dd>non-null value if

```
--toc/--table-of-contents
```

was specified </dd> <dt>

```
toc-title
```

</dt> <dd>title of table of contents (works only with EPUB and docx) </dd> <dt>

```
include-before
```

</dt> <dd>contents specified by

```
-B/--include-before-body
```

(may have multiple values) </dd> <dt>

```
include-after
```

</dt> <dd>contents specified by

```
-A/--include-after-body
```

(may have multiple values) </dd> <dt>

```
body
```

</dt> <dd>body of document </dd> <dt>

```
meta-json
```

</dt> <dd>JSON representation of all of the document’s metadata </dd> </dl><h2 id=„language-variables“>Language variables</h2> <dl readability=„11“><dt>

```
lang
```

</dt> <dd readability=„11“> identifies the main language of the document, using a code according to BCP 47 (e.g.

```
en
```

or

```
en-GB
```

). For some output formats, pandoc will convert it to an appropriate format stored in the additional variables

```
babel-lang
```

,

```
polyglossia-lang
```

(LaTeX) and

```
context-lang
```

(ConTeXt). Native pandoc

```
span
```

s and

```
div
```

s with the lang attribute (value in BCP 47) can be used to switch the language in that range. </dd> <dt>

```
otherlangs
```

</dt> <dd>a list of other languages used in the document in the YAML metadata, according to BCP 47. For example:

```
otherlangs: [en-GB, fr]
```

. This is automatically generated from the

```
lang
```

attributes in all

```
span
```

s and

```
div
```

s but can be overridden. Currently only used by LaTeX through the generated

```
babel-otherlangs
```

and

```
polyglossia-otherlangs
```

variables. The LaTeX writer outputs polyglossia commands in the text but the

```
babel-newcommands
```

variable contains mappings for them to the corresponding babel. </dd> <dt>

```
dir
```

</dt> <dd readability=„12“> the base direction of the document, either

```
rtl
```

(right-to-left) or

```
ltr
```

(left-to-right). For bidirectional documents, native pandoc

```
span
```

s and

```
div
```

s with the

```
dir
```

attribute (value

```
rtl
```

or

```
ltr
```

) can be used to override the base direction in some output formats. This may not always be necessary if the final renderer (e.g. the browser, when generating HTML) supports the Unicode Bidirectional Algorithm. When using LaTeX for bidirectional documents, only the

```
xelatex
```

engine is fully supported (use

```
--latex-engine=xelatex
```

). </dd> </dl><h2 id=„variables-for-slides“>Variables for slides</h2> Variables are available for producing slide shows with pandoc, including all reveal.js configuration options. <dl><dt>

```
slidy-url
```

</dt> <dd>base URL for Slidy documents (defaults to

```
http://www.w3.org/Talks/Tools/Slidy2
```

) </dd> <dt>

```
slideous-url
```

</dt> <dd>base URL for Slideous documents (defaults to

```
slideous
```

) </dd> <dt>

`s5-url`

</dt> <dd>base URL for S5 documents (defaults to

`s5/default`

) </dd> <dt>

`revealjs-url`

</dt> <dd>base URL for reveal.js documents (defaults to

`reveal.js`

) </dd> <dt>

`theme`

,

`colortheme`

,

`fonttheme`

,

`innertheme`

,

`outertheme`

</dt> <dd>themes for LaTeX

`beamer`

 documents </dd> <dt>

`themeoptions`

</dt> <dd>options for LaTeX beamer themes (a list). </dd> <dt>

`navigation`

</dt> <dd>controls navigation symbols in

`beamer`

documents (default is

`empty`

for no navigation symbols; other valid values are

`frame`

,

`vertical`

, and

`horizontal`

). </dd> <dt>

`section-titles`

</dt> <dd>enables on “title pages” for new sections in

`beamer`

documents (default = true). </dd> <dt>

`beamerarticle`

</dt> <dd>when true, the

`beamerarticle`

package is loaded (for producing an article from beamer slides). </dd> <dt>

`colorlinks`

</dt> <dd>add color to link text; automatically enabled if any of

`linkcolor`

,

`citecolor`

,

`urlcolor`

, or

`toccolor`

are set (for beamer only). </dd> <dt>

`linkcolor`

,

`citecolor`

,

`urlcolor`

,

`toccolor`

</dt> <dd>color for internal links, citation links, external links, and links in table of contents: uses any of the predefined LaTeX colors (for beamer only). </dd> </dl><h2 id=„variables-for-latex“>Variables for LaTeX</h2> LaTeX variables are used when creating a PDF. <dl><dt>

`papersize`

</dt> <dd>paper size, e.g.

`letter`

,

`A4`

</dd> <dt>

`fontsize`

</dt> <dd>font size for body text (e.g.

`10pt`

,

`12pt`

) </dd> <dt>

`documentclass`

</dt> <dd>document class, e.g. <a href=„https://ctan.org/pkg/article“>

```
article
```

, <a href=„https://ctan.org/pkg/report“>

```
report
```

, <a href=„https://ctan.org/pkg/book“>

```
book
```

, <a href=„https://ctan.org/pkg/memoir“>

```
memoir
```

 </dd> <dt>

```
classoption
```

</dt> <dd>option for document class, e.g.

```
oneside
```

; may be repeated for multiple options </dd> <dt>

```
geometry
```

</dt> <dd>option for <a href=„https://ctan.org/pkg/geometry“>

```
geometry
```

 package, e.g.

```
margin=1in
```

; may be repeated for multiple options </dd> <dt>

```
margin-left
```

,

```
margin-right
```

,

```
margin-top
```

,

`margin-bottom`

</dt> <dd>sets margins, if

`geometry`

is not used (otherwise

`geometry`

overrides these) </dd> <dt>

`linestretch`

</dt> <dd>adjusts line spacing using the

`setspace`

 package, e.g.

`1.25`

,

`1.5`

</dd> <dt>

`fontfamily`

</dt> <dd>font package for use with

`pdflatex`

: TeX Live includes many options, documented in the LaTeX Font Catalogue. The default is Latin Modern. </dd> <dt>

`fontfamilyoptions`

</dt> <dd>options for package used as

`fontfamily`

: e.g.

`osf,sc`

with

`fontfamily`

set to

`mathpazo`

 provides Palatino with old-style figures and true small caps; may be repeated for multiple options </dd> <dt>

`mainfont`

,

`sansfont`

,

`monofont`

,

`mathfont`

,

`CJKmainfont`

</dt> <dd>font families for use with

`xelatex`

or

`lualatex`

: take the name of any system font, using the

`fontspec`

 package. Note that if

`CJKmainfont`

is used, the

`xecjk`

 package must be available. </dd> <dt>

`mainfontoptions`

,

`sansfontoptions`

,

`monofontoptions`

,

`mathfontoptions`

,

`CJKoptions`

</dt> <dd>options to use with

`mainfont`

,

`sansfont`

,

`monofont`

,

`mathfont`

,

`CJKmainfont`

in

`xelatex`

and

`lualatex`

. Allow for any choices available through

`fontspec`

, such as the OpenType features

```
Numbers=OldStyle,Numbers=Proportional
```

. May be repeated for multiple options. </dd> <dt>

```
fontenc
```

</dt> <dd>allows font encoding to be specified through

```
fontenc
```

package (with

```
pdflatex
```

); default is

```
T1
```

(see guide to LaTeX font encodings) </dd> <dt>

```
microtypeoptions
```

</dt> <dd>options to pass to the microtype package </dd> <dt>

```
colorlinks
```

</dt> <dd>add color to link text; automatically enabled if any of

```
linkcolor
```

,

```
citecolor
```

,

```
urlcolor
```

, or

```
toccolor
```

are set </dd> <dt>

```
linkcolor
```

,

```
citecolor
```

,

```
urlcolor
```

,

```
toccolor
```

</dt> <dd>color for internal links, citation links, external links, and links in table of contents: uses any of the predefined LaTeX colors </dd> <dt>

```
links-as-notes
```

</dt> <dd>causes links to be printed as footnotes </dd> <dt>

```
indent
```

</dt> <dd>uses document class settings for indentation (the default LaTeX template otherwise removes indentation and adds space between paragraphs) </dd> <dt>

```
subparagraph
```

</dt> <dd>disables default behavior of LaTeX template that redefines (sub)paragraphs as sections, changing the appearance of nested headings in some classes </dd> <dt>

```
thanks
```

</dt> <dd>specifies contents of acknowledgments footnote after document title. </dd> <dt>

```
toc
```

</dt> <dd>include table of contents (can also be set using

```
--toc/--table-of-contents
```

) </dd> <dt>

```
toc-depth
```

</dt> <dd>level of section to include in table of contents </dd> <dt>

```
secnumdepth
```

</dt> <dd>numbering depth for sections, if sections are numbered </dd> <dt>

```
lof
```

,

```
lot
```

</dt> <dd>include list of figures, list of tables </dd> <dt>

```
bibliography
```

</dt> <dd>bibliography to use for resolving references </dd> <dt>

```
biblio-style
```

</dt> <dd>bibliography style, when used with

```
--natbib
```

and

```
--biblatex
```

. </dd> <dt>

```
biblio-title
```

</dt> <dd>bibliography title, when used with

```
--natbib
```

and

```
--biblatex
```

. </dd> <dt>

```
biblatexoptions
```

</dt> <dd>list of options for biblatex. </dd> </dl><h2 id=„variables-for-context“>Variables for ConTeXt</h2> <dl><dt>

```
papersize
```

</dt> <dd>paper size, e.g.

```
letter
```

,

```
A4
```

,

```
landscape
```

(see ConTeXt Paper Setup); may be repeated for multiple options </dd> <dt>

```
layout
```

</dt> <dd>options for page margins and text arrangement (see ConTeXt Layout); may be repeated for multiple options </dd> <dt>

```
margin-left
```

,

```
margin-right
```

,

```
margin-top
```

,

```
margin-bottom
```

</dt> <dd>sets margins, if

```
layout
```

is not used (otherwise

```
layout
```

overrides these) </dd> <dt>

```
fontsize
```

</dt> <dd>font size for body text (e.g.

```
10pt
```

,

```
12pt
```

) </dd> <dt>

```
mainfont
```

,

```
sansfont
```

,

```
monofont
```

,

```
mathfont
```

</dt> <dd>font families: take the name of any system font (see ConTeXt Font Switching) </dd> <dt>

```
linkcolor
```

,

```
contrastcolor
```

</dt> <dd>color for links outside and inside a page, e.g.

```
red
```

,

```
blue
```

(see ConTeXt Color) </dd> <dt>

```
linkstyle
```

</dt> <dd>typeface style for links, e.g.

```
normal
```

,

```
bold
```

,

```
slanted
```

,

```
boldslanted
```

,

```
type
```

,

```
cap
```

,

```
small
```

</dd> <dt>

```
indenting
```

</dt> <dd>controls indentation of paragraphs, e.g.

```
yes,small,next
```

(see ConTeXt Indentation); may be repeated for multiple options </dd> <dt>

```
whitespace
```

</dt> <dd>spacing between paragraphs, e.g.

```
none
```

,

```
small
```

(using <a href=„http://wiki.contextgarden.net/Command/setupwhitespace“>

```
setupwhitespace
```

) </dd> <dt>

```
interlinespace
```

</dt> <dd>adjusts line spacing, e.g.

```
4ex
```

(using <a href=„http://wiki.contextgarden.net/Command/setupinterlinespace“>

```
setupinterlinespace
```

); may be repeated for multiple options </dd> <dt>

```
headertext
```

,

```
footertext
```

</dt> <dd>text to be placed in running header or footer (see ConTeXt Headers and Footers); may be repeated up to four times for different placement </dd> <dt>

```
pagenumbering
```

</dt> <dd>page number style and location (using <a href=„http://wiki.contextgarden.net/Command/setuppagenumbering“>

```
setuppagenumbering
```

); may be repeated for multiple options </dd> <dt>

```
toc
```

</dt> <dd>include table of contents (can also be set using

```
--toc/--table-of-contents
```

) </dd> <dt>

```
lof
```

,

```
lot
```

</dt> <dd>include list of figures, list of tables </dd> </dl><h2 id=„variables-for-man-pages“>Variables for man pages</h2> <dl><dt>

```
section
```

</dt> <dd>section number in man pages </dd> <dt>

```
header
```

</dt> <dd>header in man pages </dd> <dt>

```
footer
```

</dt> <dd>footer in man pages </dd> <dt>

```
adjusting
```

</dt> <dd>adjusts text to left (

```
l
```

), right (

```
r
```

), center (

```
c
```

), or both (

```
b
```

) margins </dd> <dt>

```
hyphenate
```

</dt> <dd>if

```
true
```

(the default), hyphenation will be used </dd> </dl><h2 id=„using-variables-in-templates“>Using variables in templates</h2> Variable names are sequences of alphanumerics,

```
-
```

, and

```
_
```

, starting with a letter. A variable name surrounded by

```
$
```

signs will be replaced by its value. For example, the string

```
$title$
```

in <pre><title>$title$</title></pre> will be replaced by the document title. To write a literal

```
$
```

in a template, use

```
$$
```

. Templates may contain conditionals. The syntax is as follows: <pre>$if(variable)$ X $else$ Y $endif$</pre> This will include

```
X
```

in the template if

```
variable
```

has a non-null value; otherwise it will include

```
Y
```

.

```
X
```

and

```
Y
```

are placeholders for any valid template text, and may include interpolated variables or other conditionals. The

```
$else$
```

section may be omitted. When variables can have multiple values (for example,

```
author
```

in a multi-author document), you can use the

```
$for$
```

keyword: <pre>$for(author)$ <meta name=„author“ content=„$author$“ /> $endfor$</pre> You can optionally specify a separator to be used between consecutive items: <pre>$for(author)$$author$$sep$, $endfor$</pre> A dot can be used to select a field of a variable that takes an object as its value. So, for example: <pre>$author.name$ ($author.affiliation$)</pre> If you use custom templates, you may need to revise them as pandoc changes. We recommend tracking the changes in the default templates, and modifying your custom templates accordingly. An easy way to do this is to fork the pandoc-templates repository and merge in changes after each pandoc release. Pandoc understands an extended and slightly revised

version of John Gruber’s 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. <h2 id=„philosophy“>Philosophy</h2> Markdown is designed to be easy to write, and, even more importantly, easy to read: <blockquote readability=„7“> 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 </blockquote> 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. <h2 id=„paragraphs“>Paragraphs</h2> A paragraph is one or more lines of text followed by one or more blank lines. 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. <h4 id=„extension-escaped_line_breaks“>Extension:

```
escaped_line_breaks
```

</h4> 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. There are two kinds of headers: Setext and ATX. 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): <pre>A level-one header

A level-two header ——————</pre> The header text can contain inline formatting, such as emphasis (see Inline formatting, below). 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: <pre>## A level-two header ### A level-three header ###</pre> As with setext-style headers, the header text can contain formatting: <pre># A level-one header with a [link](/url) and *emphasis*</pre> 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: <pre>I like several of their flavors of ice cream: #22, for example, and #5.</pre> Headers can be assigned attributes using this syntax at the end of the line containing the header text: <pre>{#identifier .class .class key=value key=value}</pre> Thus, for example, the following headers will all be assigned the identifier

```
foo
```

: <pre># My header {#foo} ## My header ## {#foo} My other header {#foo} ——————</pre> (This syntax is compatible with PHP Markdown Extra.) Note that although this syntax allows assignment of classes and key/value attributes, writers generally don’t use all of this information. Identifiers, classes, and key/value attributes are used in HTML and HTML-based formats such as EPUB and slidy. Identifiers are used for labels and link anchors in the LaTeX, ConTeXt, Textile, and AsciiDoc writers. 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, <pre># My header {-}</pre> is just the same as <pre># My header {.unnumbered}</pre> <h4 id=„extension-auto_identifiers“>Extension:

```
auto_identifiers
```

</h4> 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, <table><thead/><tbody readability=„3“><tr class=„odd“ readability=„2“><td>

```
Header identifiers in HTML
```

</td> <td>

```
header-identifiers-in-html
```

</td> </tr><tr class=„even“ readability=„1“><td>

```
*Dogs*?--in *my* house?
```

</td> <td>

```
dogs--in-my-house
```

</td> </tr><tr class=„odd“ readability=„3“><td>

```
[HTML], [S5], or [RTF]?
```

</td> <td>

```
html-s5-or-rtf
```

</td> </tr><tr class=„even“><td>

```
3. Applications
```

</td> <td>

```
applications
```

</td> </tr><tr class=„odd“><td>

```
33
```

</td> <td>

```
section
```

</td> </tr></tbody></table> 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: <pre>See the section on [header identifiers](#header-identifiers-in-html-latex-and-context).</pre> 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

```
&lt;div&gt;
```

(or

```
&lt;section&gt;
```

) tag rather than the header itself. This allows entire sections to be manipulated using JavaScript or treated differently in CSS. Pandoc behaves as if reference links have been defined for each header. So, to link to a header <pre># Header identifiers in HTML</pre> you can simply write <pre>[Header identifiers in HTML]</pre> or <pre>[Header identifiers in HTML][]</pre> or <pre>[the section on header identifiers][header identifiers in HTML]</pre> instead of giving the identifier explicitly: <pre>[Header identifiers in HTML](#header-identifiers-in-html)</pre> 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. Like regular reference links, these references are case-insensitive. Explicit link reference definitions always take priority over implicit header references. So, in the following example, the link will point to

```
bar
```

, not to

```
#foo
```

: <pre># Foo [foo]: bar See [foo]</pre> <h2 id=„block-quotations“>Block quotations</h2> 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

```
&gt;
```

character and an optional space. (The

```
&gt;
```

need not start at the left margin, but it should not be indented more than three spaces.) <pre>> This is a block quote. This > paragraph has two lines. > > 1. This is a list inside a block quote. > 2. Second item.</pre> A “lazy” form, which requires the

```
&gt;
```

character only on the first line of each block, is also allowed: <pre>> This is a block quote. This paragraph has two lines. > 1. This is a list inside a block quote. 2. Second item.</pre> Among the block elements that can be contained in a block quote are other block quotes. That is, block quotes can be nested: <pre>> This is a block quote. > > > A block quote within

a block quote.</pre> If the

```
&gt;
```

character is followed by an optional space, that space will be considered part of the block quote marker and not part of the indentation of the contents. Thus, to put an indented code block in a block quote, you need five spaces after the

```
&gt;
```

: <pre>> code</pre> <h4 id=„extension-blank_before_blockquote“>Extension:

```
blank_before_blockquote
```

</h4> 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

```
&gt;
```

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: <pre>> This is a block quote. >> Nested.</pre> <h2 id=„verbatim-code-blocks“>Verbatim (code) blocks</h2> <h3 id=„indented-code-blocks“>Indented code blocks</h3> 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, <pre> if (a > 3) {

```
    moveShip(5 * gravity, DOWN);
  }</pre>
```

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. <h3 id=„fenced-code-blocks“>Fenced code blocks</h3> <h4 id=„extension-fenced_code_blocks“>Extension:

```
fenced_code_blocks
```

</h4> In addition to standard indented code blocks, pandoc supports fenced code blocks. These begin with a row of three or more tildes (

```
~
```

) and end with a row of tildes that must be at least as long as the starting row. Everything between these lines is treated as code. No indentation is necessary: <pre>~~~~~~~~ if (a > 3) {

```
moveShip(5 * gravity, DOWN);
```

} ~~~~~~~~</pre> 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: <pre>~~~~~~~~~~~~~~~~ ~~~~~~~~~~ code including tildes ~~~~~~~~~~ ~~~~~~~~~~~~~~~~</pre> <h4 id=„extension-backtick_code_blocks“>Extension:

```
backtick_code_blocks
```

</h4> Same as

```
fenced_code_blocks
```

, but uses backticks (

```
`
```

) instead of tildes (

```
~
```

). <h4 id=„extension-fenced_code_attributes“>Extension:

```
fenced_code_attributes
```

</h4> Optionally, you may attach attributes to fenced or backtick code block using this syntax: <pre>~~~~ {#mycode .haskell .numberLines startFrom=„100“} qsort [] = [] qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++

```
            qsort (filter (&gt;= x) xs)
```

~~~~~~</pre> <p>Here

```
mycode
```

is an identifier,

```
haskell
```

and

```
numberLines
```

are classes, and

```
startFrom
```

is an attribute with value
~~~~~~

<p>This syntax is borrowed from reStructuredText.</p> <h2 id="lists">Lists</h2> <h3 id="bullet-lists">Bullet lists</h3> <p>A bullet list is a list of bulleted list items. A bulleted list item begins with a bullet (

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

The Right Honorable Most Venerable and Righteous Samuel L.

```
100: Constable, Jr.
```

```
| 200 Main St.
```

*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, type

```
+
pandoc --list-highlight-languages
```

, or

.) Otherwise, the code block above will appear as follows:</p> <pre><pre id="mycode" class="haskell numberLines" startFrom="100">

```
&lt;code&gt;
```

). Here is a simple example:</p> <pre>* one * two * three</pre> <p>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:</p> <pre>* one * two * three</pre> <p>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.</p> <p>List items look best if subsequent lines are flush with the first line (after the bullet):</p> <pre>* here is my first

```
&lt;/code&gt;
```

</pre></pre> <p>A shortcut form can also be used for specifying the language of the code block:</p> <pre>```haskell qsort [] = [] ```</pre> <p>This is equivalent to:</p> <pre>``` {.haskell} qsort [] = [] ```</pre> <p>If the

```
list item.
fenced_code_attributes
```

* and my second.</pre> <p>But Markdown also allows a “lazy” format:</p> <pre>* here is my first list item. * and my second.</pre> <h3 id="the-four-space-rule">The four-space rule</h3> <p>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:</p> <pre> * First paragraph.

extension is disabled, but input contains class attribute(s) for the code block, the first class attribute will be printed after the opening fence as a bare word.</p> <p>To prevent all highlighting, use the

```
--no-highlight
```

flag. To set the highlighting style, use

```
    Continued.
* Second paragraph. With a code block, which must be indented
--highlight-style
    eight spaces:
      { code }</pre>
```

. For more information on highlighting, see Syntax highlighting, below.</p> <h2 id="line-blocks">Line blocks</h2> <h4 id="extension-line_blocks">Extension:

<p>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:</p> <pre>* fruits

```
line_blocks
  + apples
      - macintosh
      - red delicious
  + pears
  + peaches
|
```

</h4> <p>A line block is a sequence of lines beginning with a vertical bar (

* vegetables

) 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:</p> <pre>| The limerick packs laughs anatomical

```
  + broccoli
  + chard</pre>
```

<p>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.</p> <pre>+ A lazy, lazy, list item. + Another one; this looks

bad but is legal.

```
  Second paragraph of second
```

list item.</pre> Note: Although the four-space rule for continuation paragraphs comes from the official Markdown syntax guide, 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 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. <h3 id=„ordered-lists“>Ordered lists</h3> 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: <pre>1. one 2. two 3. three</pre> and this one: <pre>5. one 7. two 1. three</pre> <h4 id=„extension-fancy_lists“>Extension:

```
fancy_lists
```

</h4> 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. The

```
fancy_lists
```

extension also allows ‘

```
#
```

’ to be used as an ordered list marker in place of a numeral: <pre>#. one #. two</pre> <h4 id=„extension-startnum“>Extension:

```
startnum
```

</h4> 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: <pre> 9) Ninth 10) Tenth 11) Eleventh

```
     i. subone
    ii. subtwo
   iii. subthree</pre>
```

Pandoc will start a new list each time a different type of list marker is used. So, the following will create three lists: <pre>(2) Two (5) Three 1. Four * Five</pre> If default list markers are desired, use

```
#.
```

: <pre>#. one #. two #. three</pre> <h3 id=„definition-lists“>Definition lists</h3> <h4 id=„extension-definition_lists“>Extension:

```
definition_lists
```

</h4> Pandoc supports definition lists, using the syntax of PHP Markdown Extra with some extensions. <pre>Term 1 : Definition 1 Term 2 with *inline markup* : Definition 2

```
      { some code, part of Definition 2 }
  Third paragraph of definition 2.</pre>
```

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. 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. The body of the definition (including the first line, aside from the colon or tilde) should be indented four spaces. However, as with other Markdown lists, you can “lazily” omit indentation except at the beginning of a paragraph or other block element: <pre>Term 1 : Definition with lazy continuation.

```
  Second paragraph of the definition.</pre>
```

If you leave space before the definition (as in the example above), the text of the definition will be treated as a paragraph. In some output formats, this will mean greater spacing between term/definition pairs. For a more compact definition list, omit the space before the definition: <pre>Term 1

```
~ Definition 1
```

Term 2

```
~ Definition 2a
~ Definition 2b</pre>
```

Note that space between items in a definition list is required. (A variant that loosens this requirement, but disallows “lazy” hard wrapping, can be activated with

```
compact_definition_lists
```

: see Non-pandoc extensions, below.) <h3 id=„numbered-example-lists“>Numbered example lists</h3> <h4 id=„extension-example_lists“>Extension:

```
example_lists
```

</h4> 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: <pre>(@) My first example will be numbered (1). (@) My second example will be numbered (2). Explanation of examples. (@) My third example will be numbered (3).</pre> Numbered examples can be labeled and referred to elsewhere in the document: <pre>(@good) This is a good example. As (@good) illustrates, …</pre> The label can be any string of alphanumeric characters, underscores, or hyphens. <h3 id=„compact-and-loose-lists“>Compact and loose lists</h3> Pandoc behaves differently from

```
Markdown.pl
```

on some “edge cases” involving lists. Consider this source: <pre>+ First + Second:

1. Fee
2. Fie
3. Foe

+ Third</pre> Pandoc transforms this into a “compact list” (with no

```
&lt;p&gt;
```

tags around “First”, “Second”, or “Third”), while Markdown puts

```
&lt;p&gt;
```

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
```

.) <h3 id=„ending-a-list“>Ending a list</h3> What if you want to put an indented code block after a list? <pre>- item one - item two

```
  { my code block }</pre>
```

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: <pre>- item one - item two <!– end of list –>

```
  { my code block }</pre>
```

You can use the same trick if you want two consecutive lists instead of one big list: <pre>1. one 2. two 3. three <!– –> 1. uno 2. dos 3. tres</pre> <h2 id=„horizontal-rules“>Horizontal rules</h2> A line containing a row of three or more

```
*
```

,

```
-
```

, or

```
_
```

characters (optionally separated by spaces) produces a horizontal rule: <pre>* * * * ———–</pre> <h2 id=„tables“>Tables</h2> 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. <h4 id=„extension-table_captions“>Extension:

```
table_captions
```

</h4> A caption may optionally be provided with all 4 kinds of tables (as illustrated in the examples below). 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. <h4 id="extension-simple_tables">Extension:

```
simple_tables
```

</h4> Simple tables look like this: <pre> Right Left Center Default ——- —— ———- ——-

```
  12     12        12            12
 123     123       123          123
   1     1          1             1
```

Table: Demonstration of simple table syntax.</pre> 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: 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. The column headers may be omitted, provided a dashed line is used to end the table. For example: <pre>——- —— ———- ——-

```
  12     12        12            12
 123     123       123          123
   1     1          1             1
```

——- —— ———- ——-</pre> 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. <h4 id="extension-multiline_tables">Extension:

```
multiline_tables
```

</h4> 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: <pre>————————————————————- Centered Default Right Left

```
Header    Aligned         Aligned 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.
```

————————————————————————- Table: Here's the caption. It, too, may span multiple lines.</pre> 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: <pre>————– ——- ——— ———-

```
 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.</pre> 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. <h4 id=„extension-grid_tables“>Extension:

```
grid_tables
```

</h4> Grid tables look like this: <pre>: Sample grid table.
+———+———-+———–+

Fruit	Price	Advantages

+===============+===============+====================+

Bananas	$1.34	- built-in wrapper
		- bright color

+———+———-+———–+

Oranges	$2.10	- cures scurvy
		- tasty

+———+———-+———–+</pre> 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.). Cells that span multiple columns or rows are not supported. Grid tables can be created easily using Emacs table mode. Alignments can be specified as with pipe tables, by putting colons at the boundaries of the separator line after the header: <pre>+———+———-+———–+

Right	Left	Centered

+==============:+:==============+:==================:+

Bananas	$1.34	built-in wrapper

+——————+——————+——————-+</pre> For headerless tables, the colons go on the top line instead: <pre>+———-:+:———-+:———-:+

Right	Left	Centered

+——————+——————+——————-+</pre> <h4 id=„extension-pipe_tables“>Extension:

```
pipe_tables
```

</h4> Pipe tables look like this: <pre>| Right | Left | Default | Center |

—-:	:—-	———-	:—-:
12	12	12	12
123	123	123	123
1	1	1	1

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: <pre>fruit| price —–|—-: apple|2.05 pear|1.37 orange|3.09</pre> The cells of pipe tables cannot contain block elements like paragraphs and lists, and cannot span multiple lines. If a pipe table contains a row whose printable content is wider than the column width (see

```
--columns
```

), then the cell contents will wrap, with the relative cell widths determined by the widths of the separator lines. Note: pandoc also recognizes pipe tables of the following form, as can be produced by Emacs’ orgtbl-mode: <pre>| 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. <h4 id=„extension-pandoc_title_block“>Extension:

```
pandoc_title_block
```

</h4> If the file begins with a title block <pre>% title % author(s) (separated by

semicolons) % date</pre> 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: <pre>% % Author % My title % % June 15, 2006</pre> The title may occupy multiple lines, but continuation lines must begin with leading space, thus: <pre>% My title

```
on multiple lines</pre>
```

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: <pre>% Author One

```
Author Two
```

% Author One; Author Two % Author One;

```
Author Two</pre>
```

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, <pre>% PANDOC(1)</pre> will yield a man page with the title

```
PANDOC
```

and section 1. <pre>% PANDOC(1) Pandoc User Manuals</pre> will also have “Pandoc User Manuals” in the footer. <pre>% PANDOC(1) Pandoc User Manuals | Version 4.0</pre> will also have “Version 4.0” in the header. 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. (Note that, because of the way pandoc concatenates input files when several are provided, you may also keep the metadata in a separate YAML file and pass it to pandoc as an argument, along with your Markdown files: <pre>pandoc chap1.md chap2.md chap3.md metadata.yaml -s -o book.html</pre> Just be sure that the YAML file begins with

```
---
```

and ends with

```
---
```

or

```
...
```

.) 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. When pandoc is used with

```
-t markdown
```

to create a Markdown document, a YAML metadata block will be produced only if the

```
-s/--standalone
```

option is used. All of the metadata will appear in a single block at the beginning of the document. 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: <pre>— title: 'This is the title: it contains a colon' author: - Author One - Author Two tags: [nothing, nothingness] abstract: |

```
This is the abstract.
It consists of two paragraphs.
```

…</pre> 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: <pre><p>This is the abstract.</p> <p>It consists of two paragraphs.</p></pre> Variables can contain arbitrary YAML structures, but the template must match this structure. The

```
author
```

variable in the default templates expects a simple list or string, but can be changed to support more complicated structures. The following combination, for example, would add an affiliation to the author if one is given: <pre>— title: The document title author: - name: Author One

```
affiliation: University of Somewhere
```

- name: Author Two

```
affiliation: University of Nowhere
```

…</pre> To use the structured authors in the example above, you would need a custom template: <pre>$for(author)$ $if(author.name)$ $author.name$$if(author.affiliation)$ ($author.affiliation$)$endif$ $else$ $author$ $endif$ $endfor$</pre> <h2 id=„backslash-escapes“>Backslash escapes</h2> <h4 id=„extension-all_symbols_escapable“>Extension:

```
all_symbols_escapable
```

</h4> 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 <pre>**hello**</pre> ** one will get <pre>*hello*</pre> instead of <pre>hello</pre> This rule is easier to remember than standard Markdown’s rule, which allows only the following characters to be backslash-escaped: <pre>\`*_{}[]()>#+-.!</pre> (However, if the <code>markdown_strict</code> 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 <code>~</code> and in HTML and XML as <code>\ </code> or <code>\ </code>. 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 <code>\\</code> and in HTML as <code>
</code>. 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. <h2 id="smart-punctuation">Smart punctuation</h2> <h4 id="extension">Extension</h4> If the <code>-smart</code> option is specified, pandoc will produce typographically correct output, converting straight quotes to curly quotes, <code>---</code> to em-dashes, <code>--</code> to en-dashes, and <code>...</code> to ellipses. Nonbreaking spaces are inserted after certain abbreviations, such as “Mr.” Note: if your LaTeX template or any included header file call for the <code>csquotes</code> package, pandoc will detect this automatically and use <code>\enquote{...}</code> for quoted text. <h2 id="inline-formatting">Inline formatting</h2> <h3 id="emphasis">Emphasis</h3> To emphasize some text, surround it with <code>*</code>s or <code>_</code>, like this: <pre>This text is _emphasized with underscores_, and this is *emphasized with asterisks*.</pre> Double <code>*</code> or <code>_</code> produces strong emphasis: <pre>This is** strong emphasis** and with underscores.</pre> A

```
*
```

or

```
_
```

character surrounded by spaces, or backslash-escaped, will not trigger emphasis: <pre>This is * not emphasized *, and *neither is this*.</pre> <h4 id="extension-intraword_underscores">Extension:

```
intraword_underscores
```

</h4> 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

```
*
```

: <pre>feas*ible*, not feas*able*.</pre> <h3 id=„strikeout“>Strikeout</h3> <h4 id=„extension-strikeout“>Extension:

```
strikeout
```

</h4> To strikeout a section of text with a horizontal line, begin and end it with

```
~~
```

. Thus, for example, <pre>This ~~is deleted text.~~</pre> 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, <pre>H~2~O is a liquid. 2^10^ is 1024.</pre> 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~
```

. <h3 id=„verbatim“>Verbatim</h3> To make a short span of text verbatim, put it inside backticks: <pre>What is the difference between `>>=` and `>>`?</pre> If the verbatim text includes a backtick, use double backticks: <pre>Here is a literal backtick `` ` ``.</pre> (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: <pre>This is a backslash followed by an asterisk: `\*`.</pre> <h4 id=„extension-inline_code_attributes“>Extension:

```
inline_code_attributes
```

</h4> Attributes can be attached to verbatim text, just as with fenced code blocks: <pre>`<$>`{.haskell}</pre> <h3 id=„small-caps“>Small caps</h3> To write small caps, you can use an HTML span tag: <pre>Small caps</pre> (The semicolon is optional and there may be space after the colon.) This will work in all output formats that support small caps. Alternatively, you can also use the new

```
bracketed_spans
```

syntax: <pre>[Small caps]{style=„font-variant:small-caps;“}</pre> <h2 id=„math“>Math</h2> <h4 id=„extension-tex_math_dollars“>Extension:

```
tex_math_dollars
```

</h4> Anything between two

```
$
```

characters will be treated as TeX math. The opening

```
$
```

must have a non-space character immediately to its right, while the closing

```
$
```

must have a non-space character immediately to its left, and must not be followed immediately by a digit. 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: <dl readability=„0“><dt>Markdown, LaTeX, Emacs Org mode, ConTeXt, ZimWiki</dt> <dd>It will appear verbatim between

```
$
```

characters. </dd> <dt>reStructuredText</dt> <dd>It will be rendered using an <a href=„http://docutils.sourceforge.net/docs/ref/rst/roles.html#math“>interpreted text role

```
:math:
```

. </dd> <dt>AsciiDoc</dt> <dd>It will be rendered as

```
latexmath:[...]
```

. </dd> <dt>Texinfo</dt> <dd>It will be rendered inside a

```
@math
```

command. </dd> <dt>groff man</dt> <dd>It will be rendered verbatim without

```
$
```

’s. </dd> <dt>MediaWiki, DokuWiki</dt> <dd>It will be rendered inside

```
&lt;math&gt;
```

tags. </dd> <dt>Textile</dt> <dd>It will be rendered inside

```
&lt;span class="math"&gt;
```

tags. </dd> <dt>RTF, OpenDocument, ODT</dt> <dd>It will be rendered, if possible, using Unicode characters, and will otherwise appear verbatim. </dd> <dt>DocBook</dt> <dd>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. </dd> <dt>Docx</dt> <dd>It will be rendered using OMML math markup. </dd> <dt>FictionBook2</dt> <dd>If the

```
--webtex
```

option is used, formulas are rendered as images using CodeCogs or other compatible web service, downloaded and embedded in the e-book. Otherwise, they will appear verbatim. </dd> <dt>HTML, Slidy, DZSlides, S5, EPUB</dt> <dd readability=„1“> The way math is rendered in HTML will depend on the command-line options selected: <ol type=„1“ readability=„24“><li readability=„9“> 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. <li readability=„10“> If the

```
--latexmathml
```

option is used, TeX math will be displayed between

```
$
```

or

```
$$
```

characters and put in

```
&lt;span&gt;
```

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.) <li readability=„4“> If the

```
--jsmath
```

option is used, TeX math will be put inside

```
&lt;span&gt;
```

tags (for inline math) or

```
&lt;div&gt;
```

tags (for display math) with class

```
math
```

. The jsMath script will be used to render it. <li readability=„8“> 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
```

. <li readability=„10“> If the

```
--gladtex
```

option is used, TeX formulas will be enclosed in

```
&lt;eq&gt;
```

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: <pre>pandoc -s –gladtex myfile.txt -o myfile.htex gladtex -d myfile-images myfile.htex # produces myfile.html and images in myfile-images</pre> <li readability=„8“> If the

```
--webtex
```

option is used, TeX formulas will be converted to

```
&lt;img&gt;
```

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 CodeCogs will be used (

```
https://latex.codecogs.com/png.latex?
```

). <li readability=„5“> If the

```
--mathjax
```

option is used, TeX math will be displayed between

```
\(...\)
```

(for inline math) or

```
\[...\]
```

(for display math) and put in

```
&lt;span&gt;
```

tags with class

```
math
```

. The MathJax script will be used to render it as formulas. </dd> </dl><h2 id=„raw-html“>Raw HTML</h2> <h4 id=„extension-raw_html“>Extension:

```
raw_html
```

</h4> Markdown allows you to insert raw HTML (or DocBook) anywhere in a document (except verbatim contexts, where

```
&lt;
```

,

```
&gt;
```

, and

```
&amp;
```

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, Emacs Org mode, and Textile output, and suppressed in other formats. <h4 id=„extension-markdown_in_html_blocks“>Extension:

```
markdown_in_html_blocks
```

</h4> 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 <pre><table> <tr> <td>*one*</td> <td>[a link](http://google.com)</td> </tr> </table></pre> into <pre><table> <tr> <td>one</td> <td>a link</td> </tr> </table></pre> whereas

```
Markdown.pl
```

will preserve it as is. There is one exception to this rule: text between

```
&lt;script&gt;
```

and

```
&lt;style&gt;
```

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

```
&lt;div&gt;
```

tags without preventing it from being interpreted as Markdown. <h4 id=„extension-native_divs“>Extension:

```
native_divs
```

</h4> Use native pandoc

```
Div
```

blocks for content inside

```
&lt;div&gt;
```

tags. For the most part this should give the same output as

```
markdown_in_html_blocks
```

, but it makes it easier to write pandoc filters to manipulate groups of blocks. <h4 id=„extension-native_spans“>Extension:

```
native_spans
```

</h4> Use native pandoc

```
Span
```

blocks for content inside

```
&lt;span&gt;
```

tags. For the most part this should give the same output as

```
raw_html
```

, but it makes it easier to write pandoc filters to manipulate groups of inlines. <h2 id=„raw-tex“>Raw TeX</h2> <h4 id=„extension-raw_tex“>Extension:

```
raw_tex
```

</h4> 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: <pre>This result was proved in \cite{jones.1967}.</pre> Note that in LaTeX environments, like
<pre>\begin{tabular}{|l|l|}\hline Age & Frequency
\hline 18–25 & 15
26–35 & 33
36–45 & 22
\hline \end{tabular}</pre> 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, Emacs Org mode, and ConTeXt. <h2 id=„latex-macros“>LaTeX macros</h2> <h4 id=„extension-latex_macros“>Extension:

```
latex_macros
```

</h4> 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: <pre>\newcommand{\tuple}[1]{\langle #1 \rangle}
$\tuple{a, b, c}$</pre> In LaTeX output, the

```
\newcommand
```

definition will simply be passed unchanged to the output. Markdown allows links to be specified in several ways. <h3 id=„automatic-links“>Automatic links</h3> If you enclose a URL or email address in pointy brackets, it will become a link: <pre><http://google.com>
<sam@green.eggs.ham></pre> <h3 id=„inline-links“>Inline links</h3> 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.) <pre>This is an [inline link](/url), and here's [one with a title](http://fsf.org „click here for a good time!“).</pre> 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. Email addresses in inline links are not autodetected, so they have to be prefixed with

```
mailto
```

: <pre>[Write me!](mailto:sam@green.eggs.ham)</pre> <h3 id=„reference-links“>Reference links</h3> 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. The label must not be parseable as a citation (assuming the

```
citations
```

extension is enabled): citations take precedence over link labels. Here are some examples: <pre>[my label 1]: /foo/bar.html „My title, optional“ [my label 2]: /foo [my label 3]: http://fsf.org (The free software foundation) [my label 4]: /bar#special 'A title in single quotes'</pre> The URL may optionally be surrounded by angle brackets: <pre>[my label 5]: <http://foo.bar.baz></pre> The title may go on the next line: <pre>[my label 3]: http://fsf.org

```
"The free software foundation"</pre>
```

Note that link labels are not case sensitive. So, this will work: <pre>Here is [my link][FOO] [Foo]: /bar/baz</pre> In an implicit reference link, the second pair of brackets is empty: <pre>See [my website][]. [my website]: http://foo.bar.baz</pre> 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: <pre>> My block [quote]. > > [quote]: /foo</pre> <h4 id=„extension-shortcut_reference_links“>Extension:

```
shortcut_reference_links
```

</h4> In a shortcut reference link, the second pair of brackets may be omitted entirely: <pre>See [my website]. [my website]: http://foo.bar.baz</pre> <h3 id=„internal-links“>Internal links</h3> To link to another section of the same document, use the automatically generated identifier (see Header identifiers). For example: <pre>See the [Introduction](#introduction).</pre> or <pre>See the [Introduction]. [Introduction]: #introduction</pre> Internal links are currently supported for HTML formats (including HTML slide shows and EPUB), LaTeX, and ConTeXt. <h2 id=„images“>Images</h2> A link immediately preceded by a

```
!
```

will be treated as an image. The link text will be used as the image’s alt text: <pre>![la lune](lalune.jpg „Voyage to the moon“) ![movie reel] [movie reel]: movie.gif</pre> <h4 id=„extension-implicit_figures“>Extension:

```
implicit_figures
```

</h4> An image occurring by itself in a paragraph will be rendered as a figure with a caption. (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. <pre>![This is the caption](/url/of/image.png)</pre> 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: <pre>![This image won't be a figure](/url/of/image.png)\ </pre> <h4 id=„extension-link_attributes“>Extension:

```
link_attributes
```

</h4> Attributes can be set on links and images: <pre>An inline ![image](foo.jpg){#id .class width=30 height=20px} and a reference ![image][ref] with attributes. [ref]: foo.jpg „optional title“ {#id .class key=val key2=„val 2“}</pre> (This syntax is compatible with PHP Markdown Extra when only

```
#id
```

and

```
.class
```

are used.) For HTML and EPUB, all attributes except

```
width
```

and

```
height
```

(but including

```
srcset
```

and

```
sizes
```

) are passed through as is. The other writers ignore attributes that are not supported by their output format. The

```
width
```

and

```
height
```

attributes on images are treated specially. When used without a unit, the unit is assumed to be pixels. However, any of the following unit identifiers can be used:

```
px
```

,

```
cm
```

,

```
mm
```

,

```
in
```

,

```
inch
```

and

```
%
```

. There must not be any spaces between the number and the unit. For example: <pre>{ width=50% }</pre> Dimensions are converted to inches for output in page-based formats like LaTeX. Dimensions are converted to pixels for output in HTML-like formats. Use the

```
--dpi
```

option to specify the number of pixels per inch. The default is 96dpi. The

```
%
```

unit is generally relative to some available space. For example the above example will render to

```
&lt;img href="file.jpg" style="width: 50%;" /&gt;
```

(HTML),

```
\includegraphics[width=0.5\textwidth]{file.jpg}
```

(LaTeX), or

```
\externalfigure[file.jpg][width=0.5\textwidth]
```

(ConTeXt). Some output formats have a notion of a class (ConTeXt) or a unique identifier (LaTeX

```
\caption
```

), or both (HTML). When no

```
width
```

or

```
height
```

attributes are specified, the fallback is to look at the image resolution and the dpi metadata embedded in the image file. <h2 id=„spans“>Spans</h2> <h4 id=„extension-bracketed_spans“>Extension:

```
bracketed_spans
```

</h4> A bracketed sequence of inlines, as one would use to begin a link, will be treated as a span with attributes if it is followed immediately by attributes: <pre>[This is *some text*]{.class key=„val“}</pre> Pandoc’s Markdown allows footnotes, using the following syntax: <pre>Here is a footnote reference,[^1] and another.[^longnote] [^1]: Here is the footnote. [^longnote]: 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.
```

This paragraph won't be part of the note, because it isn't indented.</pre> 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.). Each footnote should be separated from surrounding content (including other footnotes) by blank lines. <h4 id=„extension-inline_notes“>Extension:

```
inline_notes
```

</h4> Inline footnotes are also allowed (though, unlike regular notes, they cannot contain multiple paragraphs). The syntax is as follows: <pre>Here is an inline note.^[Inlines notes are easier to write, since you don't have to pick an identifier and move down to type the note.]</pre> Inline and regular footnotes may be mixed freely. <h2 id=„citations“>Citations</h2> <h4 id=„extension-citations“>Extension:

```
citations
```

</h4> Using an external filter,

```
pandoc-citeproc
```

, pandoc can automatically generate citations and a bibliography in a number of styles. Basic usage is <pre>pandoc –filter pandoc-citeproc myinput.txt</pre> In order to use this feature, you will need to specify a bibliography file using the

```
bibliography
```

metadata field in a YAML metadata section, or

```
--bibliography
```

command line argument. You can supply multiple

```
--bibliography
```

arguments or set

```
bibliography
```

metadata field to YAML array, if you want to use multiple bibliography files. The bibliography may have any of these formats: <table><thead/><tbody><tr class=„odd“><td>BibLaTeX</td> <td>.bib</td> </tr><tr class=„even“><td>BibTeX</td> <td>.bibtex</td> </tr><tr class=„odd“><td>Copac</td> <td>.copac</td> </tr><tr class=„even“><td>CSL JSON</td> <td>.json</td> </tr><tr class=„odd“><td>CSL YAML</td> <td>.yaml</td> </tr><tr class=„even“><td>EndNote</td> <td>.enl</td> </tr><tr class=„odd“><td>EndNote XML</td> <td>.xml</td> </tr><tr class=„even“><td>ISI</td> <td>.wos</td> </tr><tr class=„odd“><td>MEDLINE</td> <td>.medline</td> </tr><tr class=„even“><td>MODS</td> <td>.mods</td> </tr><tr class=„odd“><td>RIS</td> <td>.ris</td> </tr></tbody></table> Note that

```
.bib
```

can be used with both BibTeX and BibLaTeX files; use

```
.bibtex
```

to force BibTeX. Note that

```
pandoc-citeproc --bib2json
```

and

```
pandoc-citeproc --bib2yaml
```

can produce

```
.json
```

and

```
.yaml
```

files from any of the supported formats. In-field markup: In BibTeX and BibLaTeX databases, pandoc-citeproc parses a subset of LaTeX markup; in CSL YAML databases, pandoc Markdown; and in CSL JSON databases, an HTML-like markup: <dl><dt>

```
&lt;i&gt;...&lt;/i&gt;
```

</dt> <dd>italics </dd> <dt>

```
&lt;b&gt;...&lt;/b&gt;
```

</dt> <dd>bold </dd> <dt>

```
&lt;span style="font-variant:small-caps;"&gt;...&lt;/span&gt;
```

or

```
&lt;sc&gt;...&lt;/sc&gt;
```

</dt> <dd>small capitals </dd> <dt>

```
&lt;sub&gt;...&lt;/sub&gt;
```

</dt> <dd>subscript </dd> <dt>

```
&lt;sup&gt;...&lt;/sup&gt;
```

</dt> <dd>superscript </dd> <dt>

```
&lt;span class="nocase"&gt;...&lt;/span&gt;
```

</dt> <dd>prevent a phrase from being capitalized as title case </dd> </dl>

```
pandoc-citeproc -j
```

and

```
-y
```

interconvert the CSL JSON and CSL YAML formats as far as possible. As an alternative to specifying a bibliography file using

```
--bibliography
```

or the YAML metadata field

```
bibliography
```

, you can include the citation data directly in the

```
references
```

field of the document’s YAML metadata. The field should contain an array of YAML-encoded references, for example: <pre>— references: - type: article-journal

```
id: WatsonCrick1953
author:
- family: Watson
  given: J. D.
- family: Crick
  given: F. H. C.
issued:
  date-parts:
  - - 1953
    - 4
    - 25
title: 'Molecular structure of nucleic acids: a structure for deoxyribose
  nucleic acid'
title-short: Molecular structure of nucleic acids
container-title: Nature
volume: 171
issue: 4356
page: 737-738
DOI: 10.1038/171737a0
URL: http://www.nature.com/nature/journal/v171/n4356/abs/171737a0.html
language: en-GB
```

...</pre> (

```
pandoc-citeproc --bib2yaml
```

can produce these from a bibliography file in one of the supported formats.) Citations and references can be formatted using any style supported by the Citation Style Language, listed in the Zotero Style Repository. These files are specified using the

```
--csl
```

option or the

```
csl
```

metadata field. By default,

```
pandoc-citeproc
```

will use the Chicago Manual of Style author-date format. The CSL project provides further information on finding and editing styles. To make your citations hyperlinks to the corresponding bibliography entries, add

```
link-citations: true
```

to your YAML metadata. 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, digit, or

```
_
```

, and may contain alphanumerics,

```
_
```

, and internal punctuation characters (

```
:.#$%&amp;-+?&lt;&gt;~/
```

). Here are some examples: <pre>Blah blah [see @doe99, pp. 33-35; also @smith04, chap. 1]. Blah blah [@doe99, pp. 33-35, 38-39 and *passim*]. Blah blah [@smith04; @doe99].</pre>

```
pandoc-citeproc
```

detects locator terms in the CSL locale files. Either abbreviated or unabbreviated forms are accepted. In the

```
en-US
```

locale, locator terms can be written in either singular or plural forms, as

```
book
```

,

```
bk.
```

/

```
bks.
```

;

```
chapter
```

,

```
chap.
```

/

```
chaps.
```

;

```
column
```

,

```
col.
```

/

```
cols.
```

;

```
figure
```

,

```
fig.
```

/

```
figs.
```

;

```
folio
```

,

```
fol.
```

/

```
fols.
```

;

```
number
```

,

```
no.
```

/

```
nos.
```

;

```
line
```

,

```
l.
```

/

```
ll.
```

;

```
note
```

,

```
n.
```

/

```
nn.
```

;

```
opus
```

,

```
op.
```

/

```
opp.
```

;

```
page
```

,

```
p.
```

/

```
pp.
```

;

```
paragraph
```

,

```
para.
```

/

```
paras.
```

;

```
part
```

,

```
pt.
```

/

```
pts.
```

;

```
section
```

,

```
sec.
```

/

```
secs.
```

;

```
sub verbo
```

,

```
s.v.
```

/

```
s.vv.
```

;

```
verse
```

,

```
v.
```

/

```
vv.
```

;

```
volume
```

,

```
vol.
```

/

```
vols.
```

;

```
&#182;
```

/

```
&#182;&#182;
```

;

```
&#167;
```

/

```
&#167;&#167;
```

. If no locator term is used, “page” is assumed. 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: <pre>Smith says blah [-@smith04].</pre> You can also write an in-text citation, as follows: <pre>@smith04 says blah. @smith04 [p. 33] says blah.</pre> 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: <pre>last paragraph… # References</pre> 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: <pre>— nocite: |

```
@item1, @item2
```

… @item3</pre> In this example, the document will contain a citation for

```
item3
```

only, but the bibliography will contain entries for

```
item1
```

,

```
item2
```

, and

```
item3
```

. It is possible to create a bibliography with all the citations, whether or not they appear in the document, by using a wildcard: <pre>— nocite: |

```
@*
```

...</pre> For LaTeX or PDF output, you can also use <a href=„https://ctan.org/pkg/natbib“>

```
natbib
```

 or <a href=„https://ctan.org/pkg/biblatex“>

```
biblatex
```

 to render bibliography. In order to do so, specify bibliography files as outlined above, and add

```
--natbib
```

or

```
--biblatex
```

argument to

```
pandoc
```

invocation. Bear in mind that bibliography files have to be in respective format (either BibTeX or BibLaTeX). For more information, see the pandoc-citeproc man page. <h2 id=„non-pandoc-extensions“>Non-pandoc extensions</h2> 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. <h4 id=„extension-angle_brackets_escapable“>Extension:

```
angle_brackets_escapable
```

</h4> Allow

```
&lt;
```

and

```
&gt;
```

to be backslash-escaped, as they can be in GitHub flavored Markdown but not original Markdown. This is implied by pandoc’s default

```
all_symbols_escapable
```

. <h4 id=„extension-lists_without_preceding_blankline“>Extension:

```
lists_without_preceding_blankline
```

</h4> Allow a list to occur right after a paragraph, with no intervening blank space. <h4 id=„extension-hard_line_breaks“>Extension:

```
hard_line_breaks
```

</h4> Causes all newlines within a paragraph to be interpreted as hard line breaks instead of spaces. 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. <h4 id=„extension-east_asian_line_breaks“>Extension:

```
east_asian_line_breaks
```

</h4> Causes newlines within a paragraph to be ignored, rather than being treated as spaces or as hard line breaks, when they occur between two East Asian wide characters. This is a better choice than

```
ignore_line_breaks
```

for texts that include a mix of East Asian wide characters and other characters. <h5 id=„extension-emoji“>Extension:

```
emoji
```

</h5> Parses textual emojis like

```
:smile:
```

as Unicode emoticons. <h4 id=„extension-tex_math_single_backslash“>Extension:

```
tex_math_single_backslash
```

</h4> 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

```
[
```

. <h4 id=„extension-tex_math_double_backslash“>Extension:

```
tex_math_double_backslash
```

</h4> Causes anything between

```
\\(
```

and

```
\\)
```

to be interpreted as inline TeX math, and anything between

```
\\[
```

and

```
\\]
```

to be interpreted as display TeX math. <h4 id=„extension-markdown_attribute“>Extension:

```
markdown_attribute
```

</h4> 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
```

. <h4 id=„extension-mmd_title_block“>Extension:

```
mmd_title_block
```

</h4> Enables a MultiMarkdown style title block at the top of the document, for example: <pre>Title: My title Author: John Doe Date: September 1, 2008 Comment: This is a sample mmd title block, with

```
       a field spanning multiple lines.</pre>
```

See the MultiMarkdown documentation for details. If

```
pandoc_title_block
```

or

```
yaml_metadata_block
```

is enabled, it will take precedence over

```
mmd_title_block
```

. <h4 id=„extension-abbreviations“>Extension:

```
abbreviations
```

</h4> Parses PHP Markdown Extra abbreviation keys, like <pre>*[HTML]: Hypertext Markup Language</pre> 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). <h4 id=„extension-autolink_bare_uris“>Extension:

```
autolink_bare_uris
```

</h4> Makes all absolute URIs into links, even when not surrounded by pointy braces

```
&lt;...&gt;
```

. <h4 id=„extension-ascii_identifiers“>Extension:

```
ascii_identifiers
```

</h4> 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. <h4 id=„extension-mmd_link_attributes“>Extension:

```
mmd_link_attributes
```

</h4> Parses multimarkdown style key-value attributes on link and image references. This extension should not be confused with the <a href=„http://pandoc.org/MANUAL.html#extension-link_attributes“>

```
link_attributes
```

 extension. <pre>This is a reference ![image][ref] with multimarkdown attributes. [ref]: http://path.to/image „Image title“ width=20px height=30px

```
     id=myId class="myClass1 myClass2"</pre>
```

Parses multimarkdown style header identifiers (in square brackets, after the header but before any trailing

```
#
```

s in an ATX header). <h4 id=„extension-compact_definition_lists“>Extension:

```
compact_definition_lists
```

</h4> Activates the definition list syntax of pandoc 1.12.x and earlier. This syntax differs from the one described above under Definition lists in several respects: No blank line is required between consecutive items of the definition list. To get a “tight” or “compact” list, omit space between consecutive items; the space between a term and its definition does not affect anything. Lazy wrapping of paragraphs is not allowed: the entire definition must be indented four spaces. <h2 id=„markdown-variants“>Markdown variants</h2> In addition to pandoc’s extended Markdown, the following Markdown variants are supported: <dl><dt>

```
markdown_phpextra
```

(PHP Markdown Extra)</dt> <dd>

```
footnotes
```

,

```
pipe_tables
```

,

```
raw_html
```

,

```
markdown_attribute
```

,

```
fenced_code_blocks
```

,

```
definition_lists
```

,

```
intraword_underscores
```

,

```
header_attributes
```

,

```
link_attributes
```

,

```
abbreviations
```

,

```
shortcut_reference_links
```

. </dd> <dt>

```
markdown_github
```

(GitHub-Flavored Markdown)</dt> <dd>

```
pipe_tables
```

,

```
raw_html
```

,

```
fenced_code_blocks
```

,

```
auto_identifiers
```

,

`ascii_identifiers`

,

`backtick_code_blocks`

,

`autolink_bare_uris`

,

`intraword_underscores`

,

`strikeout`

,

`hard_line_breaks`

,

`emoji`

,

`shortcut_reference_links`

,

`angle_brackets_escapable`

. </dd> <dt>

`markdown_mmd`

(MultiMarkdown)</dt> <dd>

`pipe_tables`

,

`raw_html`

,

`markdown_attribute`

,

```
mmd_link_attributes
```

,

```
tex_math_double_backslash
```

,

```
intraword_underscores
```

,

```
mmd_title_block
```

,

```
footnotes
```

,

```
definition_lists
```

,

```
all_symbols_escapable
```

,

```
implicit_header_references
```

,

```
auto_identifiers
```

,

```
mmd_header_identifiers
```

,

```
shortcut_reference_links
```

. </dd> <dt>

```
markdown_strict
```

(Markdown.pl)</dt> <dd>

```
raw_html
```

</dd> </dl><h2 id=„extensions-with-formats-other-than-markdown“>Extensions with formats other than Markdown</h2> Some of the extensions discussed above can be used with formats other than Markdown: <ul readability=„5“><li readability=„5“>

```
auto_identifiers
```

can be used with

```
latex
```

,

```
rst
```

,

```
mediawiki
```

, and

```
textile
```

input (and is used by default). <li readability=„6“>

```
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.)   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, DZSlides, Slidy, Slideous, or reveal.js. You can also produce a PDF slide show using LaTeX <a href=„https://ctan.org/pkg/beamer“>

```
beamer
```

. Here’s the Markdown source for a simple slide show,

```
habits.txt
```

: <pre>% 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](images/spaghetti.jpg) ## Going to sleep - Get in bed - Count sheep</pre> To produce an HTML/JavaScript slide show, simply type <pre>pandoc -t FORMAT -s habits.txt -o habits.html</pre> 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 JavaScript 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 Variables for slides, 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 <pre>pandoc -t beamer habits.txt -o habits.pdf</pre> Note that a reveal.js slide show can also be converted to a PDF by printing it to a file from the browser. <h2 id=„structuring-the-slide-show“>Structuring the slide show</h2> 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: <ul readability=„5“><li readability=„1“> A horizontal rule always starts a new slide. <li readability=„1“> A header at the slide level always starts a new slide. <li readability=„1“> Headers below the slide level in the hierarchy create headers within a slide. <li readability=„4“> 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. <li readability=„5“> 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. <h2 id=„incremental-lists“>Incremental lists</h2> By default, these writers produce 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: <pre>> - Eat spaghetti > - Drink wine</pre> In this way incremental and nonincremental lists can be mixed in a single document. <h2 id=„inserting-pauses“>Inserting pauses</h2> You can add “pauses” within a slide by including a paragraph containing three dots, separated by spaces: <pre># Slide with a pause content before the pause . . . content after the pause</pre> <h2 id=„styling-the-slides“>Styling the slides</h2> 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. All reveal.js configuration options can be set through variables. For example, themes can be used by setting the

```
theme
```

variable: <pre>-V theme=moon</pre> Or you can specify a custom stylesheet using the

```
--css
```

option. To style beamer slides, you can specify a

```
theme
```

,

```
colortheme
```

,

```
fonttheme
```

,

```
innertheme
```

, and

```
outertheme
```

, using the

```
-V
```

option: <pre>pandoc -t beamer habits.txt -V theme:Warsaw -o habits.pdf</pre> Note that header attributes will turn into slide attributes (on a

```
&lt;div&gt;
```

or

```
&lt;section&gt;
```

) 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: <pre># References {.allowframebreaks}</pre> <h2 id=„speaker-notes“>Speaker notes</h2> reveal.js has good support for speaker notes. You can add notes to your Markdown document thus: <pre><div class=„notes“> This is my note. - It can contain Markdown - like this list </div></pre> 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. <h2 id=„frame-attributes-in-beamer“>Frame attributes in beamer</h2> Sometimes it is necessary to add the LaTeX

```
[fragile]
```

option to a frame in beamer (for example, when using the

```
minted
```

environment). This can be forced by adding the

```
fragile
```

class to the header introducing the slide: <pre># Fragile slide {.fragile}</pre> All of the other frame attributes described in Section 8.1 of the Beamer User’s Guide may also be used:

```
allowdisplaybreaks
```

,

```
allowframebreaks
```

,

```
b
```

,

```
c
```

,

```
t
```

,

```
environment
```

,

```
label
```

,

```
plain
```

,

```
shrink
```

. 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: <pre>— title: - type: main

```
text: My Book
```

- type: subtitle

```
text: An investigation of metadata
```

creator: - role: author

```
text: John Smith
```

- role: editor

```
text: Sarah Jones
```

identifier: - scheme: DOI

```
text: doi:10.234234.234/33
```

publisher: My Press rights: © 2007 John Smith, CC BY-NC ...</pre> The following fields are recognized: <dl><dt>

```
identifier
```

</dt> <dd>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`

. </dd> <dt>

`title`

</dt> <dd>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
```

. </dd> <dt>

```
creator
```

</dt> <dd>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, but pandoc will attempt to translate the human-readable versions (like “author” and “editor”) to the appropriate marc relators. </dd> <dt>

```
contributor
```

</dt> <dd>Same format as

```
creator
```

. </dd> <dt>

```
date
```

</dt> <dd>A string value in

```
YYYY-MM-DD
```

format. (Only the year is necessary.) Pandoc will attempt to convert other common date formats. </dd> <dt>

```
lang
```

(or legacy:

```
language
```

)</dt> <dd>A string value in BCP 47 format. Pandoc will default to the local language if nothing is specified. </dd> <dt>

```
subject
```

</dt> <dd>A string value or a list of such values. </dd> <dt>

```
description
```

</dt> <dd>A string value. </dd> <dt>

```
type
```

</dt> <dd>A string value. </dd> <dt>

```
format
```

</dt> <dd>A string value. </dd> <dt>

```
relation
```

</dt> <dd>A string value. </dd> <dt>

```
coverage
```

</dt> <dd>A string value. </dd> <dt>

```
rights
```

</dt> <dd>A string value. </dd> <dt>

```
cover-image
```

</dt> <dd>A string value (path to cover image). </dd> <dt>

```
stylesheet
```

</dt> <dd>A string value (path to CSS stylesheet). </dd> <dt>

```
page-progression-direction
```

</dt> <dd>Either

```
ltr
```

or

```
rtl
```

. Specifies the

```
page-progression-direction
```

attribute for the <a href=„http://idpf.org/epub/301/spec/epub-publications.html#sec-spine-elem“>

```
spine
```

element. </dd> </dl> By default, pandoc will download linked media (including audio and video) and include it in the EPUB container, yielding a completely self-contained EPUB. If you want to link to external media resources instead, use raw HTML in your source and add

```
data-external="1"
```

to the tag with the

```
src
```

attribute. For example: <pre><audio controls=„1“>

```
&lt;source src="http://example.com/music/toccata.mp3"
        data-external="1" type="audio/mpeg"&gt;
&lt;/source&gt;
```

</audio></pre> 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 <ul readability=„12“><li readability=„5“> 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. For ATX-style headers the character ‘=’ will be used instead of ‘#’. <li readability=„11“> 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.) <li readability=„2“> In restructured text input, “bird track” sections will be parsed as Haskell code. <li readability=„2“> In restructured text output, code blocks with class

```
haskell
```

will be rendered using bird tracks. <li readability=„2“> In LaTeX input, text in

```
code
```

environments will be parsed as Haskell code. <li readability=„2“> In LaTeX output, code blocks with class

```
haskell
```

will be rendered inside

```
code
```

environments. <li readability=„3“> In HTML output, code blocks with class

```
haskell
```

will be rendered with class

```
literatehaskell
```

and bird tracks.   Examples: <pre>pandoc -f markdown+lhs -t html</pre> reads literate Haskell source formatted with Markdown conventions and writes ordinary HTML (without bird tracks). <pre>pandoc -f markdown+lhs -t html+lhs</pre> writes HTML with the Haskell code in bird tracks, so it can be copied and pasted as literate Haskell source. Pandoc will automatically highlight syntax in fenced code blocks that are marked with a language name. The Haskell library highlighting-kate is used for highlighting, which works in HTML, Docx, and LaTeX/PDF output. To see a list of language names that pandoc will recognize, type

```
pandoc --list-highlight-languages
```

. The color scheme can be selected using the

```
--highlight-style
```

option. The default color scheme is

```
pygments
```

, which imitates the default color scheme used by the Python library pygments (though pygments is not actually used to do the highlighting). To see a list of highlight styles, type

```
pandoc --list-highlight-styles
```

. To disable highlighting, use the

</div></pre> <p>would style the two contained lines with the `Poetry` paragraph style.</p> <p>If the styles are not yet in your reference.docx, they will be defined in the output file as inheriting from normal text. If they are already defined, pandoc will not alter the definition.</p> <p>This feature allows for greatest customization in conjunction with pandoc filters. If you want all paragraphs after block quotes to be indented, you can write a filter to apply the styles necessary. If you want all italics to be transformed to the `Emphasis` character style (perhaps to change their color), you can write a filter which will transform all italicized inlines to inlines within an `Emphasis` custom-style `span`.</p> <p>Pandoc can be extended with custom writers written in lua. (Pandoc includes a lua interpreter, so lua need not be installed separately.)</p> <p>To use a custom writer, simply specify the path to the lua script in place of the output format. For example:</p> <pre>pandoc -t data/sample.lua</pre> <p>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</p> <pre>pandoc –print-default-data-file sample.lua</pre> <p>© 2006-2016 John MacFarlane (jgm@berkeley.edu). Released under the GPL, version 2 or greater. This software carries no warranty of any kind. (See COPYRIGHT for full copyright and warranty notices.)</p> <p>Contributors include Arata Mizuki, Aaron Wolen, Albert Krewinkel, Alex Ivkin, Alex Vong, Alexander Kondratskiy, Alexander Sulfrian, Alexander V Vershilov, Alfred Wechselberger, Andreas Lööw, Andrew Dunning, Antoine Latter, Arata Mizuki, Arlo O’Keeffe, Artyom Kazak, B. Scott Michel, Ben Gamari, Beni Cherniavsky-Paskin, Benoit Schweblin, Bjorn Buckwalter, Bradley Kuhn, Brent Yorgey, Bryan O’Sullivan, Caleb McDaniel, Calvin Beck, Carlos Sosa, Chris Black, Christian Conkle, Christoffer Ackelman, Christoffer Sawicki, Clare Macrae, Clint Adams, Conal Elliott, Craig S. Bosma, Daniel Bergey, Daniel T. Staal, Daniele D’Orazio, David Lazar, David Röthlisberger, Denis Laxalde, Douglas Calvert, Emanuel Evans, Emily Eisenberg, Eric Kow, Eric Seidel, Felix Yan, Florian Eitel, François Gannaz, Freiric Barral, Freirich Raabe, Frerich Raabe, Fyodor Sheremetyev, Gabor Pali, Gavin Beatty, Gottfried Haider, Greg Maslov, Greg Rundlett, Grégory Bataille, Gwern Branwen, Hans-Peter Deifel, Henrik Tramberend, Henry de Valence, Hubert Plociniczak, Ilya V. Portnov, Ivo Clarysse, J. Lewis Muir, Jaime Marquínez Ferrándiz, Jakob Voß, James Aspnes, Jamie F. Olson, Jan Larres, Jan Schulz, Jason Ronallo, Jeff Arnold, Jeff Runningen, Jens Petersen, Jesse Rosenthal, Joe Hillenbrand, John MacFarlane, John Muccigrosso, Jonas Smedegaard, Jonathan Daugherty, Jose Luis Duran, Josef Svenningsson, Julien Cretel, Juliusz Gonera, Justin Bogner, Jérémy Bobbio, Kelsey Hightower, Kolen Cheung, Konstantin Zudov, Kristof Bastiaensen, Lars-Dominik Braun, Luke Plant, Mark Szepieniec, Mark Wright, Martin Linn, Masayoshi Takahashi, Matej Kollar, Mathias Schenner, Mathieu Duponchelle, Matthew Eddey, Matthew Pickering, Matthias C. M. Troffaes, Mauro Bieg, Max Bolingbroke, Max Rydahl Andersen, Merijn Verstraaten, Michael Beaumont, Michael Chladek, Michael Snoyman, Michael Thompson, MinRK, Morton Fox, Nathan Gass, Neil Mayhew, Nick Bart, Nicolas Kaiser, Nikolay Yakimov, Oliver Matthews, Ophir Lifshitz, Pablo

`--no-highlight` option.</p> <p>By default, pandoc’s docx output applies a predefined set of styles for blocks such as paragraphs and block quotes, and uses largely default formatting (italics, bold) for inlines. This will work for most purposes, especially alongside a `reference.docx` file. However, if you need to apply your own styles to blocks, or match a preexisting set of styles, pandoc allows you to define custom styles for blocks and text using `div`s and `span`s, respectively.</p> <p>If you define a `div` or `span` with the attribute `custom-style`, pandoc will apply your specified style to the contained elements. So, for example,</p> <pre>Get out, he said.</pre> <p>would produce a docx file with “Get out,” styled with character style `Emphatically`. Similarly,</p> <pre>Dickinson starts the poem simply: <div custom-style=„Poetry“>

Rodríguez, Paul Rivier, Paulo Tanimoto, Peter Wang, Philippe Ombredanne, Phillip Alday, Prayag Verma, Puneeth Chaganti, Ralf Stephan, Raniere Silva, Recai Oktaş, RyanGlScott, Scott Morrison, Sergei Trofimovich, Sergey Astanin, Shahbaz Youssefi, Shaun Attfield, Sidarth Kapur, Sidharth Kapur, Simon Hengel, Sumit Sahrawat, Thomas Hodgson, Thomas Weißschuh, Tim Lin, Timothy Humphries, Tiziano Müller, Todd Sifleet, Tom Leese, Uli Köhler, Václav Zeman, Viktor Kronvall, Vincent, Václav Haisman, Václav Zeman, Wandmalfarbe, Waldir Pimenta, Wikiwide, Xavier Olive, bumper314, csforste, infinity0x, nkalvi, qerub, robabla, roblabla, rodja.trappe, rski, shreevatsa.public, takahashim, tgkokk, thsutton.</p> </html>