

# Crowbook User Guide



---

---

# CROWBOOK USER GUIDE

---

---

Élisabeth Henry

*Crowbook User Guide*, 0.13.0, © Élisabeth Henry.

This guide is published under the Creative Commons Attribution-ShareAlike 4.0-International license.

# Chapter 1

## Crowbook

Crowbook's aim is to allow you to write a book in Markdown without worrying about formatting or typography, and let the program generate HTML, PDF and EPUB output for you. Its focus is novels and fiction, and the default settings should (hopefully) generate readable books with correct typography without requiring you to worry about it.

### 1.1 Example

To see what Crowbook's output looks like, you can read the Crowbook guide rendered in [HTML](#), [PDF](#) or [EPUB](#).

You can also play with the [online demo version](#).

### 1.2 Installing

There are two ways to install Crowbook: either using precompiled binaries, or compiling it using `cargo`.

#### Binaries

See [the releases page](#) to download a precompiled binary for your architecture (currently: Linux, Windows and MacOSX). Just extract the archive and run `crowbook` (or `crowbook.exe` on Windows). You might also want to copy the binary somewhere in your `PATH` for later usage.

If you are on Debian GNU/Linux or Ubuntu (on a PC architecture), you can also download `.deb` packages on [the releases page](#).

#### Using Cargo

`Cargo` is the `Rust`'s package manager. You can [install it here](#). Once it is done:

```
$ cargo install crowbook
```

will automatically download the latest `crowbook` release on [crates.io](#), compile it, and install it on your system.

*Some dependencies also require building C libraries; you might thus also need to install a C compiler and `make/cmake` build tools. You can also try to build a version of Crowbook without optional features: `cargo install crowbook --no-default-features --features "clap"` will disable syntactic highlighting and proofreading, requiring less dependencies.*

### 1.3 Dependencies

While there should be, strictly speaking, no real dependencies to be able to run Crowbook (it is published as a statically compiled binary), some features require additional commands to work correctly:

- EPUB rendering requires the `zip` command to be present on your system;
- PDF rendering requires a working installation of LaTeX (preferably `xelatex`).

## 1.4 Quick tour

The simplest command is:

```
$ crowbook <BOOK>
```

where `BOOK` is a configuration file. Crowbook will parse this file and generate HTML, EPUB, and/or PDF output formats, according to the settings in the configuration file.

To create a new book, assuming you have a list of Markdown files, you can generate a template configuration file with the `--create` argument:

```
$ crowbook my.book --create chapter_*.md
```

This will generate a default `my.book` file, which you'll need to complete. This configuration file contains some metadata, options, and lists the Markdown files.

For short books containing only a single Markdown file, it is possible to embed some metadata at the beginning of the file and use the `--single` or `-s` option to run `crowbook` directly on this Markdown file and avoid creating a separate book configuration file:

```
$ crowbook -s text.md
```

For more information, see the chapters on [the arguments supported by crowbook](#) and on [the configuration file](#).

## 1.5 Current features

### Output formats

Crowbook supports HTML, PDF and EPUB (either version 2 or 3) as output formats. See the Crowbook User Guide rendered in [HTML](#), [EPUB](#) and [PDF](#).

### Input format

Crowbook uses [pulldown-cmark](#) and thus should support most of [CommonMark Markdown](#). Inline HTML, however, is not implemented, and probably won't be, as the goal is to have books that can also be generated in PDF (and maybe ODT).

### Typographic “cleaning”

Maybe the most specific “feature” of Crowbook is that it does its best to “clean” the input text before rendering it. By default, it removes superfluous spaces and tries to use curly quotes. If the book's language is set to french, it also tries to respect french typography by replacing spaces with non-breaking ones when it is appropriate (e.g. before ‘?', ‘!', ‘,’ or ‘:’).

*Please [open an issue](#) describing typographic rules if you want them to be implemented for other languages.*

### Links handling

Crowbook tries to correctly translate local links in the input Markdown files: e.g. if you have a link to a Markdown file that is part of your book, it will be transformed into a link inside the document.

### Inline YAML blocks

Crowbook supports inline YAML blocks:

```
---
author: Me
title: My title
---
```

This is mostly useful when Crowbook is run with the `--single` argument (receiving a single Markdown file instead of a book configuration file), for short texts that only contain one “chapter”.

## Proofreading

Crowbook can also generate “proofreading” copies in HTML or PDF, highlighting grammar errors and repetitions. For more information, see [the proofreading chapter of the guide](#).

## Interactive fiction

Crowbook has experimental support for writing interactive fiction (only for HTML). For more information, read the [interactive fiction chapter](#).

## Customization

While the default settings will hopefully generate something that should look “good enough”, it is possible to customize the output, essentially by providing different [templates](#).

## Bugs

See the [github’s issue tracker](#).

## 1.6 Contributors

- [Stéphane Mourey](#) <s+crowbook AT stephanemourey DOT fr>
- [Falco Hirschenberger](#)

## 1.7 Acknowledgements

Besides the [Rust](#) compiler and standard library, Crowbook uses the following libraries: [pulldown-cmark](#), [yaml-rust](#), [mustache](#), [clap](#), [chrono](#), [uuid](#), [mime\\_guess](#), [crossbeam](#), [walkdir](#), [rustc-serialize](#), [caribon](#), [hyper](#), [url](#), [lazy\\_static](#), [regex](#), [term](#), [numerals](#), [syntect](#).

It can also embed [Highlight.js](#) in HTML output to enable syntax highlighting for code blocks.

It also uses configuration files from [rust-everywhere](#) to use [Travis](#) and [Appveyor](#) to generate binaries for various platforms on each release.

While Crowbook directly doesn’t use them, there was also inspiration from [Pandoc](#) and [mdBook](#).

Also, the [W3C HTML validator](#) and the [IDPF EPUB validator](#) prove very useful during development and testing.

## 1.8 ChangeLog

See [ChangeLog](#).

## 1.9 Contributing

See [how you can contribute to Crowbook](#).

If you find this project useful, you can also support its author by [making a Paypal donation](#).

## 1.10 Library

While the main purpose of Crowbook is to be run as a standalone program, the code is written as a library, so if you want to build on it you can use it as such. You can look at the generated documentation on [docs.rs](#).

Note that, in order to facilitate code reuse, some features have been split to separate libraries:

- [epub-builder](#) makes it easier to generate EPUB files.
- [crowbook-text-processing](#) contains all the “typographic” functions (smart quotes, handling of non-breaking spaces in french, ...).
- [crowbook-intl](#) is used for the internationalization (translation) process.

## 1.11 License

Crowbook is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License (LGPL), version 2.1 or (at your option) any ulterior version. See [LICENSE](#) for more information.

Crowbook's logo is licensed under the [Creative Commons Attribution 4.0 International license](#), based on the [Rust logo](#) by Mozilla Corporation.

Crowbook includes binary (minified) CSS and Javascript files from [Highlight.js](#), written by Ivan Sagalaev, licensed under the following terms:

*Copyright (c) 2006, Ivan Sagalaev*

*All rights reserved.*

*Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:*

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.*
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.*
- Neither the name of highlight.js nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.*

*THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.*



## Chapter 2

# Arguments

Crowbook can take a number of arguments, generally in the form:

```
crowbook [OPTIONS] [BOOK]
```

The most important argument is obviously the book configuration file. It is mandatory in most cases: if you don't pass it, Crowbook will simply display an error. In a normal use case this is the only argument you'll need to pass, as most options will be set in this configuration file.

It is, however, possible to pass more arguments to `crowbook`:

### 2.1 --create

**Usage:** `crowbook [BOOK] --create file_1.md file_2.md ...`

(or `crowbook [BOOK] -c file_1.md file_2.md ...`)

Creates a new book from a list of Markdown files. It will generate a book configuration file with all file names specified as chapters. It either prints the result to stdout (if `BOOK` is not specified) or generate the file `BOOK` (or abort if it already exists).

```
crowbook foo.book --create chapter_1.md chapter_2.md chapter_3.md
```

will thus generate a file `foo.book` containing:

```
author: Your name
title: Your title
lang: en

## Output formats

# Uncomment and fill to generate files
# output.html: some_file.html
# output.epub: some_file.epub
# output.pdf: some_file.pdf

# Or uncomment the following to generate PDF, HTML and EPUB files based on this file's name
# output: [pdf, epub, html]

# Uncomment and fill to set cover image (for EPUB)
# cover: some_cover.png

## List of chapters
+ chapter_1.md
+ chapter_2.md
+ chapter_3.md
```

while

```
crowbook --create chapter_1.md chapter_2.md chapter_3.md
```

will print the same result, but to stdout (without creating a file).

## 2.2 --single

Usage: `crowbook --single <FILE>`  
 (or `crowbook -s <FILE>`)

This argument allows you to give `crowbook` a single Markdown file. This file can contain an inline YAML block to set some book options. Inline YAML blocks must start and end with a line containing only `---` (three dashes). E.g:

```
---
author: Joan Doe
title: A short story
output: [html, epub, pdf]
---

Content of the story in Markdown.
```

If this YAML block is not at the beginning of a file, it must also be preceded by a blank line.

This allows to not have to write a `.book` configuration file for a short story or an article. `crowbook -s foo.md` is roughly equivalent to having a book configuration file containing:

```
! foo.md
```

That is, the chapter heading (if any) won't be displayed in the output documents (though they still appear in the TOC).

*Note that by default, using `--single` or `-s` sets the default LaTeX class of the book to `article` instead of `book`.*

## 2.3 --set

Usage: `crowbook <BOOK> --set [KEY] [VALUE]...`

This argument takes a list of `KEY VALUE` pairs and allows setting or overriding a book configuration option. All valid options in the configuration files are valid as keys. For more information, see [the configuration file](#).

```
$ crowbook foo.book --set tex.paper.size a4paper
```

will override the paper size for PDF generation.

## 2.4 --list-options

Usage: `crowbook --list-options`  
 (or `crowbook -l`)

Displays all the valid options that can be used, whether in a book configuration file, with `--set`, or in an inline YAML block.

## 2.5 --print-template

Usage: `crowbook --print-template <TEMPLATE>`

Prints the built-in template to stdout. Useful if you want to customize the appearance of your document. E.g., if you want to modify the CSS used for HTML rendering:

```
$ crowbook --print-template html.css > my_style.css
# edit my_style.css in your favourite editor
$ crowbook my.book --set html.css my_style.css
# or add "html.css: my_style.css" in my.book
```

## 2.6 --stats

Usage: `crowbook --stats <BOOK>`  
 (or `crowbook -S <BOOK>`)

Display some statistics (word and character counts) about the book.

## 2.7 --proofread

Usage: crowbook --proofread <BOOK>  
(or crowbook -p <BOOK>)

Equivalent to --set proofread true. Enable proofreading. See [Proofreading](#).

## 2.8 --verbose

Usage: crowbook <BOOK> --verbose

If this flag is set, Crowbook will print more warnings it detects while parsing and rendering.

## 2.9 --to

Usage: crowbook <BOOK> --to [FORMAT]  
(or crowbook <BOOK> -t [FORMAT])

Generate only the specified format. `FORMAT` must be either `epub`, `pdf`, `html`, `html.dir`, `odt` or `tex`.

If an output file for the format is not specified in the book configuration file, `crowbook` will fail to render PDF, ODT and EPUB, whereas it will print HTML and TeX files on stdout. It is, however, possible to specify a file with the `--output` option.

## Examples

```
crowbook --to html foo.book
```

will generate some HTML, and prints it either to the file specified by `output.html` in `foo.book`, or to stdout if it is not specified.

```
crowbook --to pdf --output foo.pdf foo.book
```

will generate a `foo.pdf` file.

## 2.10 --output

Usage: crowbook <BOOK> --to <FORMAT> --output <FILE>  
(or crowbook -t <FORMAT> -o <FILE> <BOOK>)

Specifies an output file. Only valid when `--to` is used.

## 2.11 --lang

Usage: crowbook --lang <LANG>  
(or crowbook -L <LANG>)

Set the runtime language used by Crowbook. Currently, only a french translation is available. By default, Crowbook uses the `LANG` environment variable to determine which language to use, but this option allows to override it (e.g. for operating systems that don't use such an option, such as Windows).

## Example

```
$ crowbook --lang fr --help
```

will display Crowbook's help message in french.

*Note that this argument has nothing to do with the `lang` option that you can set in the book configuration file, which specifies the language of the book. This argument specifies the language of the text messages that Crowbook will display while running, but has no effect on the generated documents.*



## Chapter 3

# The configuration file

If you want to use Crowbook for your book, this configuration file is all you'll have to add, beside the Markdown files containing the text of your book.

The format is not very complicated. This is an example of it:

```
# metadata
author: Joan Doe
title: Some book
lang: en

output: [html, pdf, epub]

# list of chapters
- preface.md
+ chapter_1.md
+ chapter_2.md
+ chapter_3.md
+ chapter_4.md
- epilogue.md
```

Basically, it is divided in two parts:

- a list of options, under the form `key: value`, following YAML syntax.
- a list of Markdown files.

Lines starting with the `#` characters are comments and are discarded.

### 3.1 Configuration in an inline YAML block

Sometimes, you only have one Markdown file and might not want to have a separate configuration file. In this case, you can specify options at the beginning of your Markdown file, using an inline YAML block, separated by two lines containing only `---`:

```
---
author: Joan Doe
title: Some (short) book
lang: en

output: [html, pdf, epub]
---

# Some (short) book

The book content, formatted in Markdown.
```

This method only allows to set up options: you can't include a list of chapters in this way, since the only "chapter" that will be included is this Markdown file itself.

You can then use

```
crowbook -s some_book.md
```

to generate output formats from this Markdown file.

*By default (unless `input.yaml_blocks` is set to true), Crowbook will only read those inline blocks when it is runned with `crowbook --single` (or `crowbook -s`).*

## 3.2 The list of files

There are various options to include a Markdown file.

- `+ file_name.md` includes a numbered chapter.
- `- file_name.md` includes an unnumbered chapter.
- `! file_name.md` includes a chapter whose title won't be displayed (except in the table of contents); this is useful for e.g. including a copyright at the beginning or the book, or for short stories where there is only one chapter.
- `42. file_name.md` specifies the number for a chapter.
- `@` includes a part instead of a chapter.

So a typical usage might look like this:

```
! copyright.md
- preface.md
0. chapter_0.md # We want to start at chapter 0 instead of 1
# Next chapters can be numbered automatically
+ chapter_1.md
+ chapter_2.md
...
```

There are two important things to note:

1. you must *not* use quotes around the file names.
2. the paths of these files are relative to the directory where your configuration file is. This means you can run `crowbook books/my_trilogy/first_book/config.book` without being in the book's directory.

Also note that you don't have to specify a title. This is because the title of the chapter is inferred from the Markdown document. To go back to our previous example:

```
+ chapter_1.md
```

does not specify a chapter title, because it will read it directly in `chapter_1.md`, e.g.:

```
# The day I was born #
...
```

Ideally, you should have one and only one level-one header (i.e. chapter title) in each Markdown file. If you have more than one, it might mess with the table of contents in some cases (e.g. for EPUB).

## Parts

Parts are included using the `@` character, followed by the same characters than for chapters:

```
@+ numbered_part.md
+ chapter_01.md
+ chapter_02.md
@- unnumbered_part.md
+ chapter_03.md
+ chapter_04.md
@42. part_with_number_42.md
+ chapter_05.md
```

However, you usually don't really want to have a content directly below the part, only chapters (though it can be useful to add an introduction before the first chapter of this part), so there is also a more straightforward way to use parts, using only the `@` character followed by the (markdown-formatted) title of this part:

```

@ Beginning
+ chapter_01.md
+ chapter_02.md
@ Middle
+ chapter_03.md
+ chapter_04.md
@ Appendix
- notes.md

```

With this shortcut, parts are always numbered.

## Subchapters

If you write your book to be rendered by Crowbook, it is better to have one Markdown file per chapter. It is, however, possible to work with divisions at lower levels. In order to properly include these files, you can use the following syntax:

```

-- section.md
--- subsection.md
---- subsubsection.md

```

*Note that there isn't different syntax for numbered or unnumbered sections/subsections: you can only change the numbering scheme at the chapter level.*

When including those files, Crowbook will include them in the table of content as part of the previous chapter (or section for subsections, and so on). It will also adjust the header levels of the Markdown files, so, in the previous example, a level-1 header in `section.md` will be displayed as a level-2 header in the book, and a level-1 header in `subsection.md` as a level-3 header.

*This can cause issues as only six levels of headers are supported; hence, if you include a level-5 header in `subsubsection.md`, it will cause an error.*

## 3.3 Crowbook options

The first part of the configuration file is dedicated to pass options to Crowbook. This is [YAML syntax](#), so each line should be of the form `key: value`. Note that in most cases you don't have to put string in quotes, e.g.:

```
title: My title
```

It is however possible (and sometimes necessary) to escape some characters using quotes around strings:

```
title: "My: title!"
```

It is possible to use multiline strings with `>-` and then indenting the lines that are part of the string:

```
title: >-
  A
  long
  title
author: Joan Doe
```

will set `title` to "A long title". See [block literals in YAML](#) for more information on the various way to insert multiline strings (which mostly change the way newlines will or won't be inserted).

A final note on the syntax: all options must be set *before* the first chapter inclusion (that is, a line beginning with '+', '-', 'x.' (where x is a number) or '!').

## Metadata

Metadata are data about the book. Except for `cover`, which points to an image file, all its fields are strings. The main metadata are:

- `author`
- `title`

- `subtitle`
- `lang`, the language of the book. The unicode language code should be used, e.g. `en_GB` or `en`, `fr_FR`, or `fr...`
- `cover`, a path to an image file for the cover of the book (not displayed in all output formats).

There are also additional metadata:

- `subject`
- `description`
- `license`
- `version`
- `date`

You can define your own metadata by starting an option name with `metadata.foo`.

All metadata are accessible from templates, see [Templates](#).

## The `import` special option

The special `import` option allows you to include the options of another book configuration file. E.g., assuming that you want some common options to be applied to both `foo.book` and `bar.book`, you can create a `common.book` file:

```
author: Joan Doe
lang: en
license: "Copyright (C) Joan Doe. All rights reserved."

html.header: "[Joan Doe's website](http://joan-doe.com)"
tex.template: my_template.tex
```

You can then include this file in `foo.book`:

```
import: common.book
title: Foo

+ foo_01.md
+ foo_02.md
```

Or include it in `bar.book`, but override some of its features:

```
import: common.book
title: Bar
license: CC-BY-SA # Override the license from common.book

+ bar_01.md
```

## Output options

These options specify which files to generate.

Note that all file paths are relative to the directory where the configuration file is, not to the one where you run `crowbook`. So if you set

```
output.epub: foo.epub
```

and run

```
$ crowbook some/dir/config.book
```

`foo.epub` will be generated in `some/dir`, not in your current directory.

Crowbook will try to generate each of the `output.xxx` files that are specified. That means that you'll have to set at least one of those if you want a call to



```
$ crowbook my.book
```

to generate anything. (It’s still possible to generate a specific format, and only this one, by using the `--to` and `--output` argument on the command line).

Note that some formats depend on some commands being installed on your system. Most notably, Crowbook depends on LaTeX (`xelatex` by default, though you can specify another command to use with `tex.command`) to generate a PDF file, so PDF rendering won’t work if it is not installed on your system. Crowbook also uses the `zip` command to generate the EPUB and ODT files.

Current output options are:

- `output.html`: renders a standalone HTML file.
- `output.html.dir`: renders a HTML directory with one page by chapter.
- `output.epub`: renders an EPUB file.
- `output.tex`: renders a LaTeX file.
- `output.pdf`: renders a PDF file (using `tex.command`).

(There are other output options for generating proofreading files, see [Proofreading](#), and interactive fiction, see [Interactive fiction](#).)

## The output option

Setting output file names manually can be a bit tedious, and is not always necessary. You can also specify a list of output formats with the `output` option:

```
output: [pdf, epub, html]
```

This is similar to the alternative syntax for YAML list:

```
output:
- pdf
- epub
- html
```

This option will set default output path for PDF, EPUB and HTML according to the book configuration file name. So, if your book is `my_book.book` (or `my_book.md`), it will generate `my_book.pdf`, `my_book.html` and `my_book.epub`.

*You can also infer the output file name by specifying “auto” to e.g. `output.html`. The previous example is thus equivalent to*

```
output.pdf: auto
output.epub: auto
output.html: auto
```

## output.base\_path

Additionally, the `output.base_path` option allows you to set where the output files will be written (relatively to the book configuration file). E.g.,

```
output.base_path: docs/book
output.epub: book.epub
```

will render the EPUB file in `docs/book/book.epub`.

## Input options

Crowbook does its best to improve the typography of your text. Default settings should be good enough for most usages, but you can enable/disable specific options:

- `input.clean` (default: `true`): if set to `false`, will disable all typographic “cleaning”. The algorithm is dependent on the language, though currently there is only a variant implemented for `fr` (french), dealing with the specific non-breaking spaces rules for this language.

- `input.clean.smart_quotes` (default: `true`): if set to `false`, disable the “smart quote” feature, that (tries to) replace straight quotes with curly ones. As it is an heuristics and can’t be perfect, you might want to disable it in some circumstances.
- `input.clean.ligature_dashes` (default: `false`): if set to `true`, will convert `--` to en dash (`-`) and `---` to em dash (`-`). This can be useful if you want to use these characters but can’t access them easily on your keymap; however, as it can also cause problems if you *do* want to have two successive dashes, it is disabled by default.
- `input.clean.ligature_guillemets` (default: `false`) is a similar feature for french ‘guillemets’, replacing `<<` and `>>` to `«` and `»`.

## Generic options for rendering

These options allow to configure the rendering; they are used (or at least should be) for all formats.

- `rendering.highlight` (default: `syntect`): specify if and how to perform syntax highlighting for code blocks. Valid values are:
  - `syntect`: uses the `syntect` library to perform syntax highlighting. This has the advantage of also enabling syntax highlighting for LaTeX/PDF and EPUB formats; however `syntect` support doesn’t seem to work on Windows.
  - `highlight.js`: this will use (and embed) `highlight.js` for HTML rendering, and will not perform any syntax highlighting for other output formats.
  - `none`: disable syntax highlighting. If your version of Crowbook (as is the case for Windows builds) isn’t built with `syntect` support, it will default to `none` if you try to use it.
- `rendering.highlight.theme`: only used if `rendering.highlight` is set to `syntect`, selects the theme to use for syntax highlighting. Default is “InspiredGitHub”. Valid theme names are “InspiredGitHub”, “Solarized (dark)”, “Solarized (light)”, “base16-eighties.dark”, “base16-mocha.dark”, “base16-ocean.dark” and “base16-ocean.light”.
- `rendering.num_depth`: an integer that represents the maximum level of numbering for your book. E.g., `1` will only number chapters, while `2` will number chapters, sections, but not anything below that. `6` is the maximum level and turns numbering on for all headers. (Default is `1`.) This also affects what levels will be displayed in the table of contents.
- `rendering.chapter` and `rendering.part`: the strings that will be used to design chapter and part. E.g., if you want your parts to show as “Book III” instead of “Part III”, you can set `rendering.part: Book`.
- `rendering.part.roman_numerals` and `rendering.chapter.roman_numerals`: these two booleans allow you to specify if you want roman numerals for part or chapter numbers (default is `true` for part numbers, and `false` for chapter numbers).
- `rendering.inline_toc`: if set to `true`, Crowbook will include a table of contents at the beginning of the document.
- `rendering.inline_toc.name`: the name of this table of contents as it should be displayed in the document.
- `rendering.initials`: if set to `true`, Crowbook will use initials, or “lettrines”, displaying the first letter of each chapter bigger than the others.
- `rendering.part.reset_counter`: set it to `false` if you don’t want your chapter numbers to start again at `1` at each part.

## HTML Options

These options allow you to customize the HTML rendering (used both by the default HTML standalone renderer and the HTML multifile renderer):

- `html.icon`: allows to set a `favicon` for the page.
- `html.header` and `html.footer` allow to set a custom (Markdown) string at the top and at the bottom of the HTML page. This is actually a template, so you can access metadata, such as `{{author}}`, `{{title}}`, or `{{version}}` in it. See the `template` chapter for more information on the fields you can use.

- `html.css` allows to set up a custom CSS file. You can also redefine the colours in a file and set it using `html.css.colours`.
- `html.css.add` allows you to add some specific lines of CSS in your book configuration file, that will be appended after the default CSS template.
- `html.highlight.theme` is similar to `rendering.highlight.theme` but only sets the theme for HTML output.

### Options for standalone HTML

There are a few options specific to the standalone HTML renderer (default, set with `output.html`):

- `html.standalone.one_chapter`, if set to true, will only display one chapter at a time (using Javascript), making it look similarly to the multifile HTML.
- `html.standalone.template` allows you to change or modify the HTML template for standalone HTML.

### Options for LaTeX/PDF rendering

These options allow you to customize the LaTeX renderer (and, thus, the generated PDF documents):

- `tex.template` specifies a different LaTeX template.
- `tex.class` changes the LaTeX class used.
- `tex.paper.size` and `tex.font.size` (default `a5paper` and `10pt`) allows to modify the page and font size .
- `tex.margin.left`, `tex.margin.right`, `tex.margin.top` and `tex.margin.bottom` specify the margin of the page.
- `tex.links_as_footnotes` can be set to `false` if you don't want links to also appear as footnotes (which means losing them if it is actually printed).
- `tex.highlight.theme`: similar to `rendering.highlight.theme`, but only sets the theme for LaTeX/PDF rendering.

### Options for EPUB rendering

There are also options specific to the EPUB format:

- `epub.version` can be set to 2 or 3 (default 2).
- `epub.css` can be useful if you want to specify a customized stylesheet.
- `epub.highlight.theme`: similar to `rendering.highlight.theme` but only sets a theme for EPUB output.

### Resources options

These options allow to embed additional files for some formats (currently, only EPUB). This can be useful for embedding fonts.

#### resources.files

A list of files or directories that should be added.

```
resources.files: [font1.otf, font2.otf]
```

It is also possible to specify a directory (or multiple directories). So if you have a `fonts` directories containing `font1.otf` and `font2.otf`,

```
resources.files: [fonts]
```

will be equivalent to:

```
resources.files: [fonts/font1.otf, fonts/font2.otf]
```

**default:** not set

### resources.out\_path

This option determine where (in which directory), *in the resulting document*, those files will be copied. The default is `data`, so by default the `resources.files` in the first example above will search `font1.otf` and `font2.otf` *in the same directory than the .book file*, and will copy them to `data/font1.otf` and `data/font2.otf` *in the EPUB file*. This is therefore this last path that you should use if you want to access those files e.g. in a custom CSS stylesheet.

Note that if you pass directories to `resources.files`, the whole directory would be copied. So assuming `fonts/` contains `font1.otf` and `font2.otf`

```
resources.files: [fonts]
resources.path: data
```

will copy these two files to `data/fonts/font1.otf` and `data/fonts/font2.otf` (and not `data/font1.otf` and `data/font2.otf`).

Similarly, the whole path of `resources.files` is copied, so

```
resources.files: [fonts/font1.otf, fonts/font2.otf]
```

will yield the same result.

**default:** `data`

## 3.4 Full list of options

Here is the complete list of options. You can always look at it by running `crowbook --list-options` or `crowbook -l`.

### Metadata

- **author**
  - **type:** metadata
  - **default value:** ""
  - Author of the book
- **title**
  - **type:** metadata
  - **default value:** ""
  - Title of the book
- **lang**
  - **type:** metadata
  - **default value:** `en`
  - Language of the book
- **subject**
  - **type:** metadata
  - **default value:** `not set`
  - Subject of the book (used for EPUB metadata)
- **description**
  - **type:** metadata
  - **default value:** `not set`
  - Description of the book (used for EPUB metadata)
- **cover**
  - **type:** path
  - **default value:** `not set`
  - Path to the cover of the book

## Additional metadata

- **subtitle**
  - **type:** metadata
  - **default value:** not set
  - Subtitle of the book
- **license**
  - **type:** metadata
  - **default value:** not set
  - License of the book. This information will be displayed on PDF documents
- **version**
  - **type:** metadata
  - **default value:** not set
  - Version of the book
- **date**
  - **type:** metadata
  - **default value:** not set
  - Date the book was revised

## Output options

- **output**
  - **type:** list of strings
  - **default value:** not set
  - Specify a list of output formats to render
- **output.epub**
  - **type:** path
  - **default value:** not set
  - Output file name for EPUB rendering
- **output.html**
  - **type:** path
  - **default value:** not set
  - Output file name for HTML rendering
- **output.html.dir**
  - **type:** path
  - **default value:** not set
  - Output directory name for HTML rendering
- **output.tex**
  - **type:** path
  - **default value:** not set
  - Output file name for LaTeX rendering
- **output.pdf**
  - **type:** path
  - **default value:** not set
  - Output file name for PDF rendering

- **output.odt**
  - **type:** path
  - **default value:** not set
  - Output file name for ODT rendering
- **output.html.if**
  - **type:** path
  - **default value:** not set
  - Output file name for HTML (interactive fiction) rendering
- **output.base\_path**
  - **type:** path
  - **default value:** ""
  - Directory where those output files will be written

## Rendering options

- **rendering.highlight**
  - **type:** string
  - **default value:** syntect
  - If/how highlight code blocks. Possible values: “syntect” (default, performed at runtime), “highlight.js” (HTML-only, uses Javascript), “none”
- **rendering.highlight.theme**
  - **type:** string
  - **default value:** InspiredGitHub
  - Theme for syntax highlighting (if rendering.highlight is set to ‘syntect’)
- **rendering.initials**
  - **type:** boolean
  - **default value:** false
  - Use initials (‘lettrines’) for first letter of a chapter (experimental)
- **rendering.inline\_toc**
  - **type:** boolean
  - **default value:** false
  - Display a table of content in the document
- **rendering.inline\_toc.name**
  - **type:** string
  - **default value:** “{{{loc\_toc}}}”
  - Name of the table of contents if it is displayed in document
- **rendering.num\_depth**
  - **type:** integer
  - **default value:** 1
  - The maximum heading levels that should be numbered (0: no numbering, 1: only chapters, ..., 6: all)
- **rendering.chapter**
  - **type:** string
  - **default value:** not set
  - How to call chapters

- **rendering.part**
  - **type:** string
  - **default value:** not set
  - How to call parts (or ‘books’, ‘episodes’, ...)
- **rendering.chapter.roman\_numerals**
  - **type:** boolean
  - **default value:** false
  - If set to true, display chapter number with roman numerals
- **rendering.part.roman\_numerals**
  - **type:** boolean
  - **default value:** true
  - If set to true, display part number with roman numerals
- **rendering.part.reset\_counter**
  - **type:** boolean
  - **default value:** true
  - If set to true, reset chapter number at each part
- **rendering.chapter.template**
  - **type:** string
  - **default value:** "{{number}}. {{chapter\_title}}"
  - Naming scheme of chapters, for TOC
- **rendering.part.template**
  - **type:** string
  - **default value:** "{{number}}. {{part\_title}}"
  - Naming scheme of parts, for TOC

## Special option

- **import**
  - **type:** path
  - **default value:** not set
  - Import another book configuration file

## HTML options

- **html.icon**
  - **type:** path
  - **default value:** not set
  - Path to an icon to be used for the HTML files(s)
- **html.highlight.theme**
  - **type:** string
  - **default value:** not set
  - If set, set theme for syntax highlighting for HTML output (syntect only)
- **html.header**
  - **type:** string
  - **default value:** not set

- Custom header to display at the beginning of html file(s)
- **html.footer**
  - **type:** string
  - **default value:** not set
  - Custom footer to display at the end of HTML file(s)
- **html.css**
  - **type:** template path
  - **default value:** not set
  - Path of a stylesheet for HTML rendering
- **html.css.add**
  - **type:** string
  - **default value:** not set
  - Some inline CSS added to the stylesheet template
- **html.css.colours**
  - **type:** template path
  - **default value:** not set
  - Path of a stylesheet for the colours for HTML
- **html.js**
  - **type:** template path
  - **default value:** not set
  - Path of a javascript file
- **html.css.print**
  - **type:** template path
  - **default value:** not set
  - Path of a media print stylesheet for HTML rendering
- **html.highlight.js**
  - **type:** template path
  - **default value:** not set
  - Set another highlight.js version than the bundled one
- **html.highlight.css**
  - **type:** template path
  - **default value:** not set
  - Set another highlight.js CSS theme than the default one
- **html.side\_notes**
  - **type:** boolean
  - **default value:** false
  - Display footnotes as side notes in HTML/Epub (experimental)
- **html.escape\_nb\_spaces**
  - **type:** boolean
  - **default value:** true
  - Replace unicode non breaking spaces with HTML entities and CSS
- **html.chapter.template**



- **type:** string
- **default value:** "<h1 id = 'link-{{{link}}}'>{{#has\_number}}<span class = 'chapter-header'>{{{header}}}  
{{{number}}}</span>{{#has\_title}}<br />{{/has\_title}}</h1>"
- Inline template for HTML chapter formatting
- **html.part.template**
  - **type:** string
  - **default value:** "<h2 class = 'part'>{{{header}}} {{{number}}}</h2> <h1 id = 'link-{{{link}}}'  
class = 'part'>{{{title}}}</h1>"
  - Inline template for HTML part formatting

## Standalone HTML options

- **html.standalone.template**
  - **type:** template path
  - **default value:** not set
  - Path of an HTML template for standalone HTML
- **html.standalone.one\_chapter**
  - **type:** boolean
  - **default value:** false
  - Display only one chapter at a time (with a button to display all)
- **html.standalone.js**
  - **type:** template path
  - **default value:** not set
  - Path of a javascript file

## Multifile HTML options

- **html.dir.template**
  - **type:** template path
  - **default value:** not set
  - Path of a HTML template for multifile HTML

## Interactive fiction HTML options

- **html.if.js**
  - **type:** template path
  - **default value:** not set
  - Path of a javascript file
- **html.if.new\_turn**
  - **type:** string
  - **default value:** not set
  - Javascript code that will be run at the beginning of each segment
- **html.if.end\_turn**
  - **type:** string
  - **default value:** not set
  - Javascript code that will be run at the end of each segment
- **html.if.new\_game**
  - **type:** template path
  - **default value:** not set
  - Javascript code that will be run at the beginning of a 'game'

## EPUB options

- `epub.version`
  - **type:** integer
  - **default value:** 2
  - EPUB version to generate (2 or 3)
- `epub.highlight.theme`
  - **type:** string
  - **default value:** not set
  - If set, set theme for syntax highlighting for EPUB output (syntect only)
- `epub.css`
  - **type:** template path
  - **default value:** not set
  - Path of a stylesheet for EPUB
- `epub.css.add`
  - **type:** string
  - **default value:** not set
  - Inline CSS added to the EPUB stylesheet template
- `epub.chapter.xhtml`
  - **type:** template path
  - **default value:** not set
  - Path of an xhtml template for each chapter
- `epub.toc.extras`
  - **type:** boolean
  - **default value:** true
  - Add ‘Title’ and (if set) ‘Cover’ in the EPUB table of contents
- `epub.escape_nb_spaces`
  - **type:** boolean
  - **default value:** true
  - Replace unicode non breaking spaces with HTML entities and CSS

## LaTeX options

- `tex.highlight.theme`
  - **type:** string
  - **default value:** not set
  - If set, set theme for syntax highlighting for LaTeX/PDF output (syntect only)
- `tex.links_as_footnotes`
  - **type:** boolean
  - **default value:** true
  - Add footnotes to URL of links so they are readable when printed
- `tex.command`
  - **type:** string
  - **default value:** xelatex
  - LaTeX command to use for generating PDF

- **tex.template**
  - **type:** template path
  - **default value:** not set
  - Path of a LaTeX template file
- **tex.template.add**
  - **type:** string
  - **default value:** not set
  - Inline code added in the LaTeX template
- **tex.class**
  - **type:** string
  - **default value:** book
  - LaTeX class to use
- **tex.paper.size**
  - **type:** string
  - **default value:** a5paper
  - Specifies the size of the page.
- **tex.margin.left**
  - **type:** string
  - **default value:** not set
  - Specifies left margin (note that with book class left and right margins are reversed for odd pages, thus the default value is 1.5cm for book class and 2cm else)
- **tex.margin.right**
  - **type:** string
  - **default value:** not set
  - Specifies right margin (note that with book class left and right margins are reversed for odd pages, thus the default value is 2.5cm for book class and 2cm else)
- **tex.margin.top**
  - **type:** string
  - **default value:** "2cm"
  - Specifies top margin
- **tex.margin.bottom**
  - **type:** string
  - **default value:** "1.5cm"
  - Specifies left margin
- **tex.title**
  - **type:** boolean
  - **default value:** true
  - If true, generate a title with `\maketitle`
- **tex.font.size**
  - **type:** integer
  - **default value:** not set
  - Specify latex font size (in pt, 10 (default), 11, or 12 are accepted)
- **tex.hyperref**

- **type:** boolean
- **default value:** true
- If disabled, don't try to find references inside the document
- **tex.stdpaper**
  - **type:** boolean
  - **default value:** false
  - If set to true, use 'stdpage' package to format a manuscript according to standards

## Resources option

- **resources.files**
  - **type:** list of strings
  - **default value:** not set
  - Whitespace-separated list of files to embed in e.g. EPUB file; useful for including e.g. fonts
- **resources.out\_path**
  - **type:** path
  - **default value:** data
  - Paths where additional resources should be copied in the EPUB file or HTML directory
- **resources.base\_path**
  - **type:** path
  - **default value:** not set
  - Path where to find resources (in the source tree). By default, links and images are relative to the Markdown file. If this is set, it will be to this path.
- **resources.base\_path.links**
  - **type:** path
  - **default value:** not set
  - Set base path but only for links. Useless if resources.base\_path is set
- **resources.base\_path.images**
  - **type:** path
  - **default value:** .
  - Set base path but only for images. Useless if resources.base\_path is set
- **resources.base\_path.files**
  - **type:** path
  - **default value:** .
  - Set base path but only for additional files. Useless if resources.base\_path is set.
- **resources.base\_path.templates**
  - **type:** path
  - **default value:** .
  - Set base path but only for templates files. Useless if resources.base\_path is set

## Input options

- `input.clean`
  - **type:** boolean
  - **default value:** true
  - Toggle typographic cleaning of input markdown according to lang
- `input.clean.smart_quotes`
  - **type:** boolean
  - **default value:** true
  - If enabled, tries to replace vertical quotations marks to curly ones
- `input.clean.ligature.dashes`
  - **type:** boolean
  - **default value:** false
  - If enabled, replaces ‘--’ to en dash (‘–’) and ‘---’ to em dash (‘—’)
- `input.clean.ligature.guillemets`
  - **type:** boolean
  - **default value:** false
  - If enabled, replaces ‘«’ and ‘»’ to french “guillemets” (‘«’ and ‘»’)
- `input.yaml_blocks`
  - **type:** boolean
  - **default value:** false
  - Enable inline YAML blocks to override options set in config file

## Crowbook options

- `crowbook.html_as_text`
  - **type:** boolean
  - **default value:** true
  - Consider HTML blocks as text. This avoids having <foo> being considered as HTML and thus ignored.
- `crowbook.markdown.superscript`
  - **type:** boolean
  - **default value:** false
  - If enabled, allow support for superscript and subscript using respectively foo<sup>up</sup> and bar<sub>down</sub> syntax.
- `crowbook.temp_dir`
  - **type:** path
  - **default value:** “
  - Path where to create a temporary directory (default: uses result from Rust’s `std::env::temp_dir()`)
- `crowbook.zip.command`
  - **type:** string
  - **default value:** zip
  - Command to use to zip files (for EPUB/ODT)

## Output options (for proofreading)

- `output.proofread.html`
  - **type:** path
  - **default value:** not set
  - Output file name for HTML rendering with proofread features
- `output.proofread.html.dir`
  - **type:** path
  - **default value:** not set
  - Output directory name for HTML rendering with proofread features
- `output.proofread.pdf`
  - **type:** path
  - **default value:** not set
  - Output file name for PDF rendering with proofread features

## Proofreading options (only for `output.proofread.*` targets)

- `proofread`
  - **type:** boolean
  - **default value:** false
  - If set to false, will deactivate proofreading even if one of `output.proofread.x` is present
- `proofread.languagetool`
  - **type:** boolean
  - **default value:** false
  - If true, try to use language tool server to grammar check the book
- `proofread.languagetool.port`
  - **type:** integer
  - **default value:** 8081
  - Port to connect to languagetool-server
- `proofread.grammalecte`
  - **type:** boolean
  - **default value:** false
  - If true, try to use grammalecte server to grammar check the book
- `proofread.grammalecte.port`
  - **type:** integer
  - **default value:** 8080
  - Port to connect to grammalecte server
- `proofread.repetitions`
  - **type:** boolean
  - **default value:** false
  - If set to true, use Caribon to detect repetitions
- `proofread.repetitions.max_distance`
  - **type:** integer
  - **default value:** 25
  - Max distance between two occurrences so it is considered a repetition

- **proofread.repetitions.fuzzy**
  - **type:** boolean
  - **default value:** true
  - Enable fuzzy string matching
- **proofread.repetitions.fuzzy.threshold**
  - **type:** float
  - **default value:** 0.2
  - Max threshold of differences to consider two strings a repetition
- **proofread.repetitions.ignore\_proper**
  - **type:** boolean
  - **default value:** true
  - Ignore proper nouns for repetitions
- **proofread.repetitions.threshold**
  - **type:** float
  - **default value:** 2.0
  - Threshold to detect a repetition

Note that these options have a type, which in most case should be pretty straightforward (a boolean can be `true` or `false`, an integer must be composed by a number, a string is, well, any string (note that you might need to use quotes if it includes some characters that may lead the YAML parser to read it as an array, an integer or a list), and a list of strings is a list containing only strings, see [YAML syntax](#)). The `path` type might puzzle you a bit, but it's equivalent to a string, except Crowbook will consider it relatively to the book file. The `template path` type is just the path of a template. Metadata are just strings.





## Chapter 4

# Markdown format

Crowbook uses [pulldown-cmark](#), which is an implementation of [CommonMark](#), so for more information on Markdown syntax, you can refer to this website.

However, pulldown-cmark also implements a handful of unofficial extensions, and Crowbook also adds its own variants, so there are a few syntax elements that are not covered by the CommonMark reference.

### 4.1 Tables

Tables can be included in your Markdown file. E.g.:

```
|           Author           |           Book           |
|-----|-----|
| Anne Rice                 | Interview With the Vampire |
| Terry Pratchett           | Hogfather                 |
| George Martin             | A Dance with Dragons      |
```

will render as

Author	Book
Anne Rice	Interview With the Vampire
Terry Pratchett	Hogfather
George Martin	A Dance with Dragons

*Crowbook doesn't currently support specifying column alignment.*

### 4.2 Footnotes

Footnotes can be specified the following way:

```
Footnotes can be useful[1] and make you look clever.

[1]: But you shouldn't use them too much.
```

Will be rendered as:

*Footnotes can be useful<sup>1</sup> and make you look clever.*

You can use multiple paragraphs in a footnote definition. This can sometimes be useful, but it can also be tricky, as if you only let an empty line before the next paragraph, it will also be included in the footnote. And probably the next one and the following one too:

```
This is a footnote usage[1].

[1]: This is obviously part of the footnote definition.

This is less obviously ALSO part of the footnote definition.
```

---

<sup>1</sup>But you shouldn't use them too much.

---

This is NOT part of the footnote.

Due to its own quirks, Crowbook will duplicate footnotes if you reference them multiple times:

This footnote is unique<sup>[^2]</sup> but referenced twice<sup>[^2]</sup>.

<sup>[^2]</sup>: Or is it?

*This footnote is unique<sup>2</sup> but referenced twice<sup>3</sup>.*

### 4.3 Superscript and subscript

Crowbook 0.12.0 added experimental support for superscript and subscript, using respectively `fooup` and `bardown` syntax, which will render as “foo<sup>up</sup>” and “bar<sub>down</sub>”; this feature is quite a hack above the Markdown parsing library, and as such might cause issue if you mix it with other Markdown syntax elements (or, in the previous example, for smart quote detection). This is why you’ll need to enable it with `crowbook.markdown.superscript`.

### 4.4 “Standalone” images

This is not *per se* a new syntactic element, but Crowbook distinguish two kind of images, according to their position in the document:

- standalone images, which are the only elements of a paragraph;
- inline images, which are placed in a container containing other elements.

Standalone images will typically be resized to fill the width of the page, while inline images are not resized.


This image is on its own paragraph, and thus considered “standalone” and resized to fit width:

---

<sup>2</sup>Or is it?

<sup>3</sup>Or is it?



While this one  is embedded in a paragraph and its size is unchanged.

## 4.5 Interactive fiction

Crowbook also adds some syntax for interactive fiction, to make embedding Javascript code easier. It is only enabled for the interactive fiction renderer. For more information, see the [chapter on this matter](#).



# Chapter 5

## Templates

Crowbook allows the user to specify a number of templates.<sup>1</sup>

Each of this template can be overridden by a custom one, by setting e.g.:

```
html.css: my_template.css
```

in the book configuration file. The templates that you are most susceptible to modify are the following:

- `html.css`: stylesheet for HTML output;
- `epub.css`: stylesheet for EPUB output;
- `tex.template`: template of a LaTeX file.

### 5.1 Create and edit template

Except for inline templates, which are set directly in the book configuration file:

```
# Template that modify how a chapter title is displayed
rendering.chapter.template: "{{{loc_chapter}}} {{{number}}}: {{{chapter_title}}}"

# CSS code added to default CSS templates (but don't override it)
html.css.add: "h1 { background-color: red; }"
epub.css.add: "h1 { background-color: gray; }"

# LaTeX code added to default LaTeX template (but doesn't override it)
template.tex.add: "\usepackage{libertineotf}"
```

most templates must be in a separate file:

```
tex.template: my_template.tex
```

#### **--print-template**

The easiest way to create a new template is to start with the default one. In order to do so, you can use the `--print-template` argument:

```
$ crowbook --print-template tex.template > my_template.tex
```

In order to get the `chapter.xhtml` template for EPUB3, you'll also have to use `--set epub.version 3`:

```
$ crowbook --print-template epub.chapter.xhtml --set epub.version 3 > my_epub3_template.xhtml
```

<sup>1</sup>Some of them, though, are not “real” templates, they are just files that are inserted, but can’t contain mustache tags. This will probably evolve in future versions.

## Mustache syntax

Crowbook uses [rust-mustache](#) as its templating engine, which allows to use [Mustache](#) syntax in the templates.

It mainly boils down to using `{{{foo}}}`<sup>2</sup> to insert the value of variable `foo` in the document:

```
<h1 class = "title" >{{{title}}}</h1>
<h2 class = "author">{{{author}}}</h2>
```

Mustache also provides the possibility of checking whether a variable is set:

```
{{#foo}}
Foo exists
{/foo}}
{{^foo}}
Foo does not exist
{{^foo}}
```

Crowbook uses this and sets some variables to `true` to allow templates to conditionally include some portions.

E.g., in `html.css`:

```
{{#lang_fr}}
/* Make list displays '-' instead of bullets */
ul li {
  list-style-type: '-';
  padding-left: .5em;
}
{/lang_fr}}
```

In this case, Crowbook sets a variable whose name is equal to `lang_foo` to `true`, allowing to have different styles for some elements according to the language.

For more information about Mustache syntax, see the [Mustache manual](#).

## Syntax in LaTeX

Since LaTeX already uses a lot of curly brackets, the default template sets an alternative syntax to access variables, with `<<&foo>>`<sup>3</sup>:

```
\title{<<&title>>}
\author{<<&author>>}
<<#has_date>>\date{<<&date>>}</has_date>
```

## html.js

The javascript file used by both the standalone HTML renderer and the multiple files HTML renderer.

This is not currently an actual template, just a plain javascript file which cannot contain `mustache` tags.

## html.css

The main CSS file used by both the standalone HTML renderer and the multiple files HTML renderer.

## html.css.colours

A CSS file containing only colour settings. Used by `html.css`.

This is not currently an actual template, just a plain CSS file which cannot contain `mustache` tags.

<sup>2</sup>Mustache also provides the `{{foo}}` variant, which HTML-escapes the content of the variable. You should not use this, as Crowbook already renders and correctly escapes the variables it sets for use in templates.

<sup>3</sup>`<<foo>>` might also work, but the ampersand is required to prevent mustache HTML-escaping the value. This is not good because:

1. escaping is already done by Crowbook before setting variable content;
2. escaping HTML in a LaTeX document won't probably look good.

## 5.2 List of templates

## html.css.print

An additional CSS file used by both the standalone HTML renderer and the multiple files HTML renderer. Its purpose is to provide CSS instructions for printing (i.e., when the user clicks the `print` button in her browser).

This is not currently an actual template, just a plain CSS file which cannot contain `mustache` tags.

## html.highlight.js

A javascript file used by both HTML renderers to highlight codes in code blocks. It should be a variant of `highlight.js`.

This is not an actual template, just a plain javascript file.

## html.highlight.css

A CSS file used by both HTML renderers to set the theme of `highlight.js`. It should, though, be an `highlight.js` theme.

This is not an actual template, just a plain CSS file.

## html.standalone.js

A javascript file used only by the standalone HTML renderer. Its main purpose is to handle the displaying of a single chapter at a time when `one_chapter` is set to true.

## html.standalone.template

The main HTML template for standalone HTML renderer.

## html.dir.template

The main HTML template for multiple files HTML renderer.

## tex.template

The main (and currently only) template used by the LaTeX renderer.

## epub.chapter.xhtml

This template is the main template used by the Epub renderer. It contains the XHTML template that will be used for each chapter.

## epub.css

This template is used by the Epub renderer and contains the style sheet.

## Inline templates

Crowbook also has some inline templates, that are set in the book configuration file:

- `tex.template.add`, `html.css.add` and `epub.css.add` allow to specify some LaTeX or CSS code directly in the book configuration file. This code will be added respectively to `tex.template`, `html.css` or `epub.css` template. For CSS templates, this code is inserted at the end of the template (allowing to redefine rules that are set by the template); for the LaTeX template, the code is inserted at the end of the preamble, just before the `\begin{document}` tag, allowing to redefine commands.
- `rendering.inline_toc.name` sets the name of the inline table of content, if it is displayed. By default, is set to `{{{loc_toc}}}`, that is, a localised version of “Table of Contents”.
- `rendering.chapter.template` sets the naming scheme for chapters, while `rendering.part.template` does the same for part. These are used only for text-only output, such as in the TOC. `html.chapter.template` and `html.part.template` allow to change the HTML formatting for parts and chapters. *These options should probably only be used if you know what you’re doing, as they can break the document.* If you only need to change the name of chapters or parts, use `rendering.part` and `rendering.chapter` instead.

## 5.3 List of accessible variables

### Metadata

For every template, Crowbook exports all of the metadata:

- `author`;
- `title`;
- `subtitle`;
- `lang`;
- `subject`;
- `description`;
- `license`;
- `version`;
- `date`;
- any option `metadata.foo` defined in the book configuration file will also be exported as `metadata.foo`.

These metadata can contain Markdown, which will be rendered. E.g., setting `date: "20th of september"` will render `september` in bold, using `<b>` tag for HTML or `\textbf` for LaTeX. If you need to use these data in places that don't support formatted text (e.g. in meta tags), you can use the raw content by accessing `xxx_raw` instead (e.g., `author_raw`, `title_raw`, ...). (Note that the content of the raw metadata is *not* HTML-escaped, so in this case you might want to use `{{xxx_raw}}` instead of `{{{xxx_raw}}}`.)

For each metadata `foo` that is set, Crowbook also inserts a `has_foo` bool set to true. This allows to use Mustache's section for some logic, e.g.:

```

{{{title}}}
{{#has_version}}, version {{{version}}}{/has_version}

```

will avoid rendering “, version” when `version` is not set.

### Localisation strings

For all templates, Crowbook also exports some localisation strings `loc_foo`. They currently include:

Localisation key	Value in english
<code>loc_toc</code>	Table of contents
<code>loc_cover</code>	Cover
<code>loc_title</code>	Title
<code>loc_chapter</code>	Chapter
<code>loc_part</code>	Part
<code>loc_notes</code>	Notes
<code>loc_display_all</code>	Display all chapters
<code>loc_display_one</code>	Display one chapter

### Template-dependent values

Crowbook also exports some additional fields for some templates, see below.



Mustache tag	Value	Available in...
<code>content</code>	A rendered version of the book or chapter's content	<code>html.standalone.template</code> , <code>html.dir.template</code> , <code>tex.template</code> , <code>epub.chapter.xhtml</code>
<code>toc</code>	A rendered version of the table of contents	<code>html.standalone.template</code> , <code>html.dir.template</code>
<code>has_toc</code>	Set to <code>true</code> if the table of contents is not empty	<code>html.standalone.template</code>
<code>colours</code>	The content of <code>html.css.colours</code>	<code>html.css</code>
<code>footer</code>	The content of <code>html.footer</code>	<code>html.standalone.template</code> , <code>html.dir.template</code>
<code>header</code>	The content of <code>html.header</code>	<code>html.standalone.template</code> , <code>html.dirtemplate</code>
<code>script</code>	The javascript file for this HTML document	<code>html.standalone.template</code> , <code>html.dir.template</code>
<code>style</code>	The CSS file for this HTML document, that is, a rendered version of <code>html.css</code>	<code>html.standalone.template</code>
A variable whose name corresponds to <code>lang</code> in book options (e.g. <code>lang_en</code> if <code>lang</code> is set to "en", <code>lang_fr</code> if it is set to "fr", ...)	<code>true</code>	<code>html.css</code> , <code>epub.css</code>
<code>chapter_title</code>	The title of current chapter	<code>html.dir.template</code> , <code>epub.chapter.xhtml</code> , <code>rendering.chapter.template</code>
<code>chapter_title_raw</code>	The title of current chapter (raw text without HTML formatting)	<code>html.dir.template</code> , <code>epub.chapter.xhtml</code> , <code>rendering.chapter.template</code>
<code>json_data</code>	Contains structured data with book's metadata in JSON-LD format	<code>html.standalone.template</code> , <code>html.dir.template</code>
<code>highlight_code</code>	True if <code>html.highlight_code</code> is <code>true</code>	<code>html.standalone.template</code> , <code>html.dir.template</code>
<code>highlight_css</code>	The content of <code>html.highlight.css</code>	<code>html.standalone.template</code>
<code>highlight_js</code>	The base64-encoded content of <code>html.highlight.js</code>	<code>html.standalone.template</code>
<code>common_script_one_chapter</code>	The content of <code>html.js</code> True if <code>html.standalone.one_chapter</code> is <code>true</code> , else not present	<code>html.single.js</code> <code>html.standalone.template</code> , <code>html.standalone.js</code>
<code>book.svg</code>	The base64-encoded image of the button to display all chapters	<code>html.standalone.js</code> , <code>html.standalone.template</code>
<code>pages.svg</code>	The base64-encoded image of the button to display one chapter at a time	<code>html.standalone.js</code> , <code>html.standalone.template</code>
<code>favicon</code>	The <code>&lt;link rel = "icon" ...&gt;</code> tag if <code>html.icon</code> is set	<code>html.standalone.template</code> , <code>html.dir.template</code>
<code>menu_svg</code>	The base64-encoded image of the hamburger menu image	<code>html.standalone.template</code>
<code>prev_chapter</code>	Title and a link of previous chapter	<code>html.dir.template</code>
<code>next_chapter</code>	Title and a link of nexts chapter	<code>html.dir.template</code>
<code>class</code>	The content of <code>tex.class</code>	<code>tex.template</code>
<code>book</code>	True if <code>tex.class</code> is <code>book</code> , not set else	<code>tex.template</code>
<code>tex_lang</code>	The babel equivalent of <code>lang</code>	<code>tex.template</code>
<code>tex_title</code>	Set to <code>true</code> to run <code>\maketitle</code>	<code>tex.template</code>
<code>tex_size</code>	The font size to pass to the LaTeX class	<code>tex.template</code>
<code>has_tex_size</code>	Set to <code>true</code> if <code>tex_size</code> is set	<code>tex.template</code>
<code>margin_left</code> , <code>margin_right</code> , <code>margin_top</code> , <code>margin_bottom</code>	The margins of the document	<code>tex.template</code>
<code>initials</code>	True if <code>rendering.initials</code> is <code>true</code> , not set else	<code>tex.template</code>
<code>additional_code</code>	Set to the content of <code>tex.template.add</code> , <code>html.css.add</code> or <code>epub.css.add</code>	<code>tex.template</code> , <code>html.css</code> , <code>epub.css</code>



## Chapter 6

# Proofreading with Crowbook

Crowbook includes some proofreading features, that can be enabled if you set one of the

- `output.proofread.html`
- `output.proofread.html_dir`
- `output.proofread.pdf`

output files (or include `proofread.pdf` in the list of formats to render to `output`). This allows you to generate different files for publishing and proofreading (you probably don't want to publish a version that highlights your grammar errors or your repetitions).

Current proofreading features are:

- repetition detection;
- grammar check.

### 6.1 Enabling proofreading

Since proofreading can take quite a lot of time, particularly for a long book, it is disabled by default. You'll have to run

```
$ crowbook --proofread my.book
```

or

```
$ crowbook -p my.book
```

to generate proofreading copies. Alternatively, if you want it to be activated each time you run `crowbook` on this book (which is *not* recommended for long books, particularly if you want to perform a grammar check), you can set

```
proofread: true
```

in the book configuration file.

### 6.2 Repetition detection

Repetition detection is enabled with:

```
proofread.repetitions: true
```

It uses `Caribon` library to detect the repetition in your text. Since the notion of a repetition is relatively arbitrary, it is possible to adapt the settings. Default are:

```
# The maximum distance between two identical words to
# consider them a repetition
proofread.repetitions.max_distance: 25
# The minimal number of occurrences to consider it a repetition
proofread.repetitions.threshold: 2.0
# Ignore proper nouns (words starting by a capital,
# not at a beginning of a sentence)
```

```
proofread.repetitions.ignore_proper: true

# Activate fuzzy string matching
proofread.repetitions.fuzzy: true
# The maximal ratio of difference to consider
# that two words are identical
# (E.g., with 0.2, "Rust" and "Lust" won't be
# considered as the same word, but they will be with 0.5)
proofread.repetitions.fuzzy.threshold: 0.2
```

For more information, see [Caribon's](#) documentation.

*Currently, repetitions are not displayed in PDF proofreading output.*

## 6.3 Grammar checking

### With LanguageTool

Crowbook can use [LanguageTool](#) to detect grammar errors in your text. It is, however, a bit more complex to activate.

First, you'll have to activate this feature in your book configuration file:

```
# Activate language tool support
proofread.languagetool: true
# (Optional) Sets the port number to connect to (default below)
proofread.languagetool.port: 8081
```

You'll then have to download the stand-alone version of [LanguageTool](#). It includes a server mode, which you'll have to launch:

```
$ java -cp languagetool-server.jar org.languagetool.server.HTTPServer --port 8081
```

You can also use the LanguageTool GUI (`languagetool.jar`) and start the server from the menu "Text Checking -> Options". This also allows you to configure LanguageTool more precisely by activating or deactivating rules.

You can then run Crowbook, and it will highlight grammar errors in HTML or PDF proofreading output files.

*Note: running a grammar check on a long book (like a novel) can take up to a few minutes.*

### With Grammalecte

[Grammalecte](#) is a grammar checker specialized for the french language. If the language of your book is french, you can use it in a similar fashion to `languagetool`:

```
# Activate grammalecte support
proofread.grammalecte: true
# (Optional) Sets the port number to connect to (default below)
proofread.grammalecte.port: 8080
```

You'll also need to run the Grammalecte server. First [download the CLI and server version](#), then:

```
$ python3 server.py
```

You can then run Crowbook with `--proofread` to check the grammar of your book. It is possible to run both LanguageTool and Grammalecte on the same book (though might take a while for a long book...).

## Chapter 7

# Interactive fiction

Version 0.12.0 added experimental support for writing interactive fiction.

*Since this support is experimental, it means it can change at anytime, and there is no guarentee that the interactive fiction you write for the current version of Crowbook will work with the next release, even if it isn't a major release.*

### 7.1 Basics

If you want to have a non-linear story, you can simply use Markdown links just as you would for any other link:

```
* [Open the treasure chest](open_chest.md)
* [It might be trapped, stay away from it](stay_away.md)
```

All Crowbook renderers should render this correctly, allowing the reader to “choose her adventure”. Note, however, that you still need to include all these Markdown files in you book configuration files.

### 7.2 The interactive fiction renderer

While the above allows you to generate correct EPUB and PDF files, it will still display all the content if the reader chooses to read your book linearly. While this may not be a problem, you might want to only display the part of the book that the reader is actually exploring.

In order to do so, you can use the interactive fiction html renderer:

```
output.html.if: my_book.html
```

This output is similar to the standalone HTML output, except the option to display only a chapter at a time is always true, and there is no way to display the table of contents.

### 7.3 Using Javascript in your interactive fiction

While the above allows the reader to choose his own path, its interactivity is quite limited. With the interactive fiction renderer, it is possible to include Javascript code in your Markdown files, using a code block element:

```
You open the chest, and you find a shiny sword. Yay!
```

```
    user_has_sword = true;
```

This Javascript code can return a string value, which will be displayed inside the document according to the reader’s previous choices:

```
You encounter a goblin, armed with a knife!
```

```
    if (user_has_sword) {
      return "You kill him with your sword, congratulations!";
    } else {
      return "You don't have any weapon, you die :(";
    }
}
```

*Note that only the interactive fiction renderer supports this way of embedding Javascript code. If you try to render a document containing such code blocks to EPUB, PDF, or the “normal” HTML renderer, they will be displayed as regular code blocks.*

## 7.4 Embedding Markdown in your Javascript code embedded in your Markdown

If you want to include Markdown formatting in the Javascript code (to display a passage or another without having to write HTML code), you can use the `@"..."@` syntax:

```
@"You face a troll!"@
if (user_has_sword) {
  @"* [Attack him with your sword](fight_troll.md)"@
} else {
  @"* [Better run away](run_away.md)"@
}
```

Note that in this case you don't need to return a value, this is done behind your back. Similarly, `@"..."@` blocks don't require semicolons.

If you need to access the value of a Javascript variable inside this Markdown code, you can use the `{{...}}` syntax:

```
var name = prompt("Enter your name", "world");
@"Hello, {{name}}"@
```

## 7.5 Conditional blocks

Sometimes, you want some text (or Javascript code) to only be displayed (or run) when the reader reads this passage the first time, or alternatively when she goes back to it. While it is trivial to add some code to check that, it is a common enough pattern to justify its own variant : you'll just have to insert a named code block with the number:

```
```1
@"Only displayed at first passage"@
```

```2
@"Only displayed at second passage"@
```

```>2
@"Displayed at passage 3, 4 and so on.
```
```

## 7.6 Interactive fiction options

As other renderers, there are options specific to the interactive fiction.

`html.if.new_game` allows you to specify the path to a Javascript that will be run at the beginning of the game. Since this code is not embedded in a function and is at the root (and the beginning) of the document, it is a good place to declare all the functions and the global variables you might need for your interactive fiction mechanics. e.g.:

```
html.if.new_game: some_file.js
```

`html.if.new_turn` and `html.if.end_turn` allow you to specify some Javascript code that will be executed at the beginning and the end of each segment. Unlike `html.if.new_game`, the (usually shorter) code is specified inline, and can return a string value that will be displayed at the beginning and the end of each segment. This is exactly like including code blocks at the beginning or the end of each of your Markdown file. E.g.:

```
html.if.new_turn: "nb_turns += 1;"  
html.end_turn: "return 'Turn: ' + nb_turns;"
```

**html.if.script** allows you to specify the name of a Javascript file to override the default script.





## Chapter 8

# Tips and tricks

### 8.1 Using Crowbook with Emacs' markdown mode

If you use Emacs as a text editor, there is a nice `Markdown mode` to edit Markdown files.

It is possible to use Crowbook for HTML previewing in this mode, which **requires only minimal configuration and tweaking**:

```
(custom-set-variables
 '(markdown-command "crowbook - -qs --to html --output -"))
```

You can then use `markdown-preview` (or `C-c C-c p`) to run Crowbook on this file and preview it in your browser, or run `markdown-live-preview-mode` to see a live preview (updated each time you save your file) in Emacs' integrated browser.

#### Some explanations if it looks a bit cryptic to you

We set `markdown-command` to `crowbook`, the reason for this is a bit obvious. The arguments we give to `crowbook` might be a bit less obvious:

- the first argument, `-`, is actually the book file: it tells `crowbook` to read it from standard input.
- `-qs` or `--quiet --single` tells Crowbook that it is a standalone markdown file, and not a book configuration file, and to be a bit quiet on error/info messages;
- `--to html` specifies that HTML must be generated;
- `--output -` tells Crowbook to display the result on the stdout, even if you set `output.html` to `some_file.html`.

### 8.2 Embedding fonts in an EPUB file

In order to embed fonts in an EPUB file, you'll first have to edit the stylesheet, which you can first obtain with:

```
$ crowbook --print-template epub.css > my_epub_stylesheet.css
```

You'll need to use the **@font-face attribute**:

```
@font-face {
  font-family: MyFont;
  src: url(data/my_font.ttf);
}
```

Then you can add `my_font.ttf` to the files that need to be added to the EPUB zip file:

```
title: My Book
author: Me

cover: cover.png
output.epub: book.epub
```

```
resources.files: [my_font.ttf]
```

(Note that you'll have to repeat the process for the different `font-weight` and `font-style` variants of your font if you want it to display correctly when there is some text in **bold**, *italics*, or ***both***.)

## Chapter 9

# Contributing

Crowbook is a free software, and you can contribute to it. There are some things that can be accessible even if you don't know anything about programming.

### 9.1 Internationalization

Crowbook aims to support multiple languages. However, unfortunately, currently only english, french, and (in a more limited way) spanish are currently supported. If you want to have better support for the language you write in, there are easy things you can do:

- Provide a translation for the few strings that Crowbook insert into the rendered documents. This is really easy, as there are currently less than a dozen of them, and you just need to create a new variant of the `lang/en.yaml` file.
- Open an [issue](#) about the typographic rules in your language, if Crowbook doesn't cover them.
- Provide a translation for the Crowbook program. It requires creating a variant of the `.po` file, which is a bit more work because (at this time) it's around 1,500 lines (and less a priority than the first item of this list, as this translation only affects the the command-line interface and not the rendered documents).



# ChangeLog

## 0.14.0-beta (2017-10-08)

- Bugfixes:
  - EPUB: escape quotes in content.opf.
  - LaTeX/PDF: allow hyphenations in typewriter font.
- User interface:
  - User interface is quite fancier, with progress bars and all
  - Debug/warning/info levels should be displayed in a more coherent manner
  - New `--no-fancy` option if you don't like the fancy UI (or if it doesn't work in your terminal)
  - New `--force-emoji` option to force emoji usage.
- Library interface:
  - Removed `Book::set_verbosity` method (uses a logger library instead).
- Now requires `rustc >= 1.20.0`

## 0.13.0 (2017-07-14)

- Breaking changes:
  - The `template.tex` template was quite modified. Crowbook now uses custom command for most mark-down elements, defined in the template. This allow an user to redefine the way the book is rendered without having to modify Crowbook itself. Unfortunately, as tex templates for previous Crowbook versions won't work anymore.
  - the `resources.files` option is now a YAML list of strings, instead of a comma-seprated string.
- Add support for grammalecte grammar checker.
- `crowbook` command takes a new argument, `-S` or `--stats` which displays stats on the book (currently, words and characters count).
- Interactive fiction:
  - Added conditional blocks.
- Options:
  - `output.xxx` options can now take the “auto” value, which will infer the output file name based on the book file name.
  - `output` is a new option that can specify a series of format to render, with default output file name.
  - `proofread.grammlecte` and `proofread.grammlecte.port` allow respectively to enable grammar checking with Grammalecte and (optionnally) to specify the port to connect.
  - `tex.margin.left`, `tex.margin.right`, `tex.margin.bottom` and `tex.margin.top` are new options that allow to specify margins for LaTeX/PDF outputs.
  - `tex.paper_size` was renamed `tex.paper_size`.
- HTML:
  - Add JSON-LD structured data to the book's HTML files.

- Bugfixes:
  - LaTeX: fix rendering of part/chapter (part previously displayed as chapter and its first chapter as part)
  - EPUB:
    - \* Fix `.rule` so it is centered despite KOBO CSS injection.
  - Fix resources/images inclusion when they are symlinks to the actual file.

## 0.12.0 (2017-06-05)

This release includes a few new features, such as the possibility to include Markdown files as section/subsections and not only as chapter, experimental support for superscript and subscript, and yet more experimental support for writing interactive fiction.

- Book configuration file:
  - It is now possible to include subchapters using the `--` command (with one dash per sublevel: `--- foo.md` will include `foo.md` as a subsection).
- Markdown:
  - Added support for superscript and subscript features, using respectively `fooup` or `bardown` syntax.
- New options:
  - `rendering.chapter`: change what is displayed in place of “chapter”.
  - `rendering.part`: change what is displayed in place of “part”.
  - `html.chapter.template` and `html.part.template` allow to tune a little how the chapters and parts are displayed in HTML.
  - `tex.hyperref`, if set to `false`, will disable hyperrefs for local links. Can be useful for some files.
  - `crowbook.html_as_text`, if set to `false`, will not treat HTML as text but ignore it.
  - `subtitle`, as its name suggest, set the subtitle of a book.
  - `crowbook.markdown.superscript` can enable or disable superscript/subscript “extension”.
- Rendering:
  - Change the way chapters are displayed by default.
  - PDF output now has a better-looking (hopefully) title page.
  - Internal links are a bit more flexible, e.g. if you link to `Readme.html` it will now try to link to the chapter corresponding to `Readme.md`.
- Bugfixes:
  - LaTeX:
    - \* Fix bug in syntax highlighting.
    - \* Fix label placements (and thus navigation inside PDF document).
  - EPUB:
    - \* Add unnamed but numbered chapters to the TOC.
    - \* Fix HTML escaping issue for chapter titles.
    - \* Fix the way parts were handled in the TOC.
  - Book configuration file:
    - \* Fix issue when setting custom number for parts.
- Crowbook now requires `rustc >= 1.17.0`

### 0.11.4 (2017-03-21)

- An image can now be considered standalone even if it is inside a link.
- Bugfixes:
  - HTML/EPUB: use raw (not HTML rendered) metadata in the places where HTML code is not appropriate. Templates can use this metadata with the `foo_raw` value.
  - HTML/EPUB: fix double-escaping/rendering issues in titles.
  - EPUB:
    - \* Escape title and author before feeding them to epub-builder.
    - \* Fix content.opf issue by not rendering first chapter’s title (marked as beginning of document) in `<guide>`.
- Rendering:
  - HTML/EPUB: standalone images are now displayed centered.

### 0.11.3 (2017-03-19)

- When crowbook parses the book’s contents, it now detects which features are used. This is useful in various ways:
  - The ODT renderer only displays a global warning showing the lists of used features that are not implemented, instead of a warning each time such a feature is encountered.
  - The LaTeX and HTML/EPUB renderers only initialize `syntect` (which can take some time) if code blocks are used in the document.
  - The LaTeX renderer only requires LaTeX packages that are actually used in the document.
- Command-line interface:
  - Warnings are now displayed by default.
  - The (undocumented) `--debug` argument has been removed.
  - The status of some messages have been modified (“warning” to “debug” or “error” to “warning”).
- Deprecated option:
  - `crowbook.verbose` has been deprecated, at it should be set by the CLI.

### 0.11.2 (2017-03-05)

- General:
  - When there is an error setting an option from the book configuration file (e.g. because it is an invalid key), print an error but do not abort, only ignore this specific option.
- New options:
  - `tex.stdpage`: if set to `true`, will use the `stdpage` package to render the book according to standards for submitting manuscripts.
  - `rendering.highlight.theme` allows to specifies a theme for syntax highlighting (only used if `rendering.highlight` is set to “syntect”).
  - `html.highlight.theme`, `epub.highlight.theme` and `tex.highlight.theme` allow to specify a theme for HTML/EPUB/LaTeX renderers (only used with `syntect`).
- Deprecated option:
  - `proofread.nb_spaces`.
- Rendering:
  - `[syntect]` (<https://crates.io/crates/syntect>) is now the default for `rendering.highlight`. Concretely, this means that by default syntax highlighting is now done when `crowbook` is run instead of using `[highlight.js]` (<https://highlightjs.org/>).

- EPUB:
  - \* Now sets the “cover-image” property and meta so readers should display cover correctly.
  - \* Narrow non-breaking spaces should display more correctly on KOBO ereaders (hoping this won’t break the way they are displayed everywhere else).
- Proofreading:
  - Repetition detection is now a bit less of an hack, and should cause less problems when used in conjunction with grammar checking. It now also works on PDF output (so the way it is highlighted could be improved).
- Bugfixes:
  - Fix `mimetype` of EPUB files (make sure it is always “stored” and not “deflated” by the `zip` command).
  - Avoid initializing `syntect` (at the cost of performances) if it is not used.
  - Avoid creating an empty file if some book renderer fails (e.g. EPUB or ODT because `zip` command is not present).

### 0.11.1 (2017-01-05)

- Rendering:
  - Avoid page break before or after a separating rule.
  - Add support for `syntect` for syntax highlighting. This is activated by setting `rendering.highlight` to `syntect` (see below).
  - EPUB:
    - \* Set back HTML escape of narrow non-breaking spaces to `true` by default (it caused problems on some readers, but cause much more serious one if `false`).
    - \* Add more information to guide/nav landmarks.
  - LaTeX/PDF:
    - \* Improve the way code blocks are displayed, using the `mdframed` package.
    - \* Try to reduce the issues of too long lines when using code and code blocks, by inserting `\allowbreak{}` directive after some characters (`.`, `/`, `_`, ...).
    - \* Block quotes are now displayed in italics.
    - \* Tables now use `tabularx`, which allows to break too long lines (it still doesn’t break pages, though).
- New options:
  - `rendering.highlight` can be set to `none`, `highlight.js` (by default, enables syntax highlighting via Javascript, but only on HTML document) or `syntect` (doesn’t necessitate javascript, and can work in EPUB or LaTeX, but more experimental at this point).
- Deprecated options:
  - `html.highlight_code` (use `rendering.highlight` instead).
- Bugfixes:
  - HTML (standalone): fix the template that contained invalid HTML code.

### 0.11.0 (2016-12-31)

Substantial changes in this release, the more important one being support for parts!

- **Breaking changes:** the API has undergone some breaking changes, hoping they will be the last ones for a while. API should now be more simple and consistent (?). This version contains also substantial options renaming (see below).
- Crowbook now supports parts (above the “chapter” level), using the ‘@’ character in the book configuration file.
- Command-line interface:



- 
- Behaviour of `--to` should now be consistent for all output formats.
  - If `--output` is set to `-`, prints to `stdout`.
  - Conversely, if `<BOOK>` is set to `-`, reads from `stdin`.
  - Path specified by `--output` is now interpreted relatively to current directory (and not depending on where `<BOOK>` is or its options).
- Rendering:
    - Chapters with no titles now have an empty title added (so it can at least display e.g. “Chapter X”).
    - EPUB:
      - \* The `toc.ncx` file now displays links to “title” and (if set) “cover” (can be deactivated, see below).
      - \* The `toc.ncx` file now displays toc levels below chapter.
      - \* The table of contents is now displayed inline if `rendering.inline_toc` is set to `true`.
  - New options:
    - `epub.toc.extras`, set to `true` by default, will add links to the title and the cover (if it is set) in the table of contents.
    - `epub.escape_nb_spaces`, similar to `html.escape_nb_spaces` and set to `false` by default since at least Kobo reader don’t seem to be able to understand the CSS to escape those nb spaces...
    - `rendering.chapter.roman_numerals`, if set to `true`, will display chapter numbers using roman numerals.
    - `rendering.part.roman_numerals`, if set to `true` (it is by default) will display part numbers using roman numerals.
    - `rendering.part.template` specifies the numbering scheme of parts.
    - `rendering.part.reset_counter`, if set to `true` (it is by default), resets chapter number to zero after a part.
  - Renamed options:
    - `import_config` renamed to `import`.
    - `rendering.chapter_template` renamed to `rendering.chapter.template`.
    - `html_single.html` renamed to `html.standalone.template`.
    - `html_single.js` renamed to `html.standalone.js`.
    - `html_single.one_chapter` renamed to `html.standalone.one_chapter`.
    - `output.html_dir` renamed to `output.html.dir`.
    - `output.proofread.html_dir` renamed to `output.proofread.html.dir`.
    - `html_dir.index.html` and `html.dir.chapter.html` have been merged and both renamed to `html.dir.template`.
    - `tex.font_size` renamed to `tex.font.size`.
  - Bugfixes:
    - EPUB:
      - \* Fix duplicate HTML escaping (resulting in e.g. “&” instead of “&”).
    - HTML directory:
      - \* Fix panic when trying to generate html directory in “./xxx” (#23).
      - \* Fix “previous chapter” links that were not displayed when “html.header” was set.
    - HTML:
      - \* Fix the way initial letter is displayed if `rendering.initials` is `true`.
  - Internationalization:
    - Strings in generated Crowbook documents (such as “Table of contents”, “Title”, “Cover” and such) are now translated in spanish.

## 0.10.4 (2016-12-16)

- New options:
  - `tex.font_size` specifies an optional font size (in pt) passed to the LaTeX class (must be 10, 11 or 12).
  - `tex.title` can be set to `false` to avoid rendering the title with `\maketitle`.
  - `tex.paper_size` specifies the paper size for PDF output.
  - `tex.template.add`, `html.css.add` and `epub.css.add` allow to specify inline LaTeX or CSS code in the book configuration file that will be added respectively to `tex.template.add`, `html.css.add` and `epub.css.add`.
  - `html.icon` allows to specify the path of an icon for HTML documents.
- Command-line interface:
  - Paths that are displayed should now be normalized, e.g. “foo/bar.pdf” instead of “baz/../foo/bar.pdf”.
- Rendering:
  - HTML:
    - \* The default CSS style has been slightly modified.

## 0.10.3 (2016-11-19)

- Building:
  - Crowbook now requires `rustc >= 1.13.0` to build.
  - Pre-built binaries now all include the proofreading feature.
  - Linux binaries are now linked against `musl` library so they should really work on any Linux platform.
- Bugfixes:
  - Fixed escaping of `author` and `title` fields.
  - Fixed text cleaning in ODT rendering that causes corrupt files to be generated.
- CommandLine Interface:
  - Crowbook displays clearer error messages when unable to launch `latex` or `zip` commands.
  - Crowbook uses `term` library in order to display colours correctly on e.g. Windows.
  - The new argument `--lang` (or `-L`) allows to set the runtime language used by Crowbook, overriding `LANG` environment variable.
  - `--list-options` no longer uses colours as it caused problems depending on the terminal or when piping to `less`.

## 0.10.2 (2016-10-21)

Only minor changes in this version:

- Options:
  - `author` and `title`’s default values are both set to the empty string, instead of `Anonymous` and `Untitled`.
  - `input.autoclean` has been renamed `input.clean`.
  - `input.smart_quotes` has been renamed `input.clean.smart_quotes`.
  - new option: `input.clean.ligature.dashes` will (if set to true) replace `--` to en dash (–) and `---` to em dash (—).
  - new option: `input.clean.ligature.guillemets` will (if set to true) replace `<<` and `>>` to french guillemets (« and »).
- Rendering:
  - HTML: if `html_single.one_chapter` and `rendering.inline_toc` are both set to true, only render the TOC if currently displayed chapter is the first.

## 0.10.1 (2016-10-18)

Fixed a bug in `fr.po` translation that prevented building from fresh install.

## 0.10.0 (2016-10-18)

This release contains some breaking changes (mostly for the API, which has been split in separate libraries). It also features some internationalization support, and the program should now be translated if your `LANG` environment variable is set to french.

- **Breaking changes:**

- Templates:

- \* Conditional inclusion depending on `lang` must now be done using `lang_LANG` (e.g. `lang_fr`, `lang_en`, and so on). This might impact custom `epub.css` and `html.css` templates.

- API:

- \* The `escape` module has been moved to a separate crate, `crowbook_text_processing`. The `cleaner` module is no longer public, but the features it provided are also available in `crowbook_text_processing`.

- New options:

- `html.css.colours` allows to provide a CSS file that only redefine the colour scheme. Such a file can be built from `crowbook --print-template html.css.colours`.

- `input.smart_quotes`: if set to `true`, tries to replace `'` and `"` by curly quotes.

- Command line interface:

- Crowbook is now (imperfectly) localized in french, and can be translated to other languages.

- Added the `--quiet` (or `-q`) argument, that makes crowbook run without displaying any messages (except some error messages at this point).

- Rendering:

- HTML:

- \* The table of contents menu is no longer displayed in the HTML single renderer if it doesn't contain at least two elements.

- \* The default colour theme has been modified a little.

- Bugfixes:

- Fix the escaping of non-breaking spaces in EPUB, as `&nbsp;`; and its friends aren't valid entities in XHTML, apparently.

## 0.9.1 (2016-09-29)

This release mainly introduces generation of proofreading copies, allowing, if they are set (and `crowbook` was compiled with the `proofread` feature) to generate proofreading copies, using tools to check grammar and detect repetitions. These features are currently experimental.

- New options:

- `html.escape_nb_spaces`, if set to `true` (by default), will replace unicode non breaking spaces with HTML entites and CSS so it can display correctly even if reader's don't have a browser/font supporting these unicode symbols.

- Output files for proofread documents: `output.proofread.html`, `output.proofread.html_dir` and `output.proofread.pdf`.

- Proofread options `proofread.repetitions` and `proofread.nb_spaces` have been added.

- \* `proofread.nb_spaces`, if set to `true`, highlights non-breaking spaces so it is easier to check the correct typography of a book. Note that it requires that `html.escape_nb_spaces` be set to `true` (default) to work.

- \* `proofread.repetitions`, if set to true, uses **Caribon** to highlight repetitions in a document. It also uses the settings `proofread.repetitions.fuzzy`, `proofread.repetitions.max_distance`, `proofread.repetitions.threshold`, `proofread.repetitions.fuzzy.threshold`, `proofread.repetitions.ignore_proper`. Note that this feature is not built by default, you'll have to build crowbook with `cargo build --release --features "repetitions"`.
- New default settings for options:
  - `tex.command` is now `xelatex` by default.
- Rendering:
  - LaTeX:
    - \* Add support for xelatex in the default template.
  - Improved french cleaner (see [an article \(in french\)](#) that talks about what it does).
- Crowbook user guide: documentation has been updated to correctly reflect 0.9.x options.
- API:
  - `clap` dependency is now optional, people who want to use Crowbook as a library should include it with `crowbook = { version = "0.9", default-features = false }`. (`clap` is still required to build a working binary).

## 0.9.0 (2016-09-23)

The main objective of this release is to clean public interfaces, in order to limit breaking changes in the future. *Ideally*, all pre-1.0 releases should thus be 0.9.x. Concretely, this meant three things:

- reducing the surface of Crowbook's library API;
- cleaning options names
- cleaning the names exported in templates and document them, in order not to break user-defined templates in future (non-breaking) releases. More detailed changes for this release:
- **Breaking change for users:** removed `tex.short` option, replaced by a more generic `tex.class` (default being `book`). `html.crowbook_link` has also been removed.
- Renamed options. Using the old name will print a deprecation warning but will still work for a while.
  - `temp_dir` -> `crowbook.temp_dir`
  - `zip.command` -> `crowbook.zip.command`
  - `verbose` -> `crowbook.verbose`
  - `html.print_css` -> `html.css.print`
  - `html.display_chapter` -> `html_single.one_chapter`
  - `html.script` -> `html_single.js`
  - `numbering` -> `rendering.num_depth`
  - `numbering_template` -> `rendering.chapter_template`
  - `display_toc` -> `rendering.inline_toc`
  - `toc_name` -> `rendering.inline_toc.name`
  - `enable_yaml_blocks` -> `input.yaml_blocks`
  - `use_initials` -> `rendering.initials`
  - `autoclean` -> `input.autoclean`
  - `html_dir.css` -> `html.css` (not really renamed, `html_dir.css` is actually removed as there is no point in having different CSS for standalone and multifile HTML rendering, is it?)
- New options:
  - More metadata: `license`, `version` and `date`. These metadata are not treated by the renderers, but they are exported to the templates: `{{metadata}}` allows to access the content. If they are present, a `has_metadata` is also set to true, allowing to do something like `{{title}} {{#has_version}}version {{version}} {{/has_version}}`.

- Yet more metadata: it is possible to add custom metadata by prefixing it with `metadata..` They will then be accessible in the templates, with dots (‘.’) replaced by underscores (‘\_’). E.g., with `metadata.foo: bar` you can access it in your templates with `{{metadata.foo}}`.
- `output.base_path` specifies a directory where the output files (set by `output.FORMAT` will be written.
- `resources.base_path.templates` specifies where templates can be found.
- Rendering:
  - Metadata can now contain Markdown and will be rendered by the renderers. This might not be a good idea for common fields (e.g. “title”), though. Use with caution.
  - `rendering.inline_toc.name` can use `{{loc_toc}}` to specify a localized name.
  - HTML:
    - \* `html.top` and `hstml.footer` are now considered as templates, so you can use some `{{metadata}}` in it.
    - \* Improved the way footnotes are displayed.
    - \* In standalone HTML, footnotes are rendered at the end of the document instead of at the end of the chapter, unless `html_single.one_chapter` is true.
  - LaTeX:
    - \* If `tex.class` is set to `article`, chapters will be displayed as `\sections` since `article` class doesn’t handle chapters.
    - \* Except if `tex.class` is set to `book`, margins are now symmetrical.
    - \* LaTeX template now uses `version` and `date`.
- Bugfixes:
  - `import_config` only import options from another book file that are not equal to the default ones and that haven’t already been set by the caller. E.g., `author: foo` then `import_config: bar.book` won’t erase the author previously set.
  - `import_config` now correctly translates the imported book’s paths.
- Crowbook program:
  - Still working to improve error messages.
  - `crowbook --list-options` uses colours. This might hurt your eyes.
  - Display an error message when mustache can’t compile a template, instead of panicking.
- Internal/API:
  - Added static methods to `Logger` to allows displaying messages more easily/prettily.
  - Reduce public API’s surface so less changes will need to be considered breaking in the future.

## 0.8.0 (2016-09-19)

This release adds support for syntax highlighting in code blocks, customized top and footer blocks for HTML rendering, and the special `import_config` option that allows to import options from another book file. It also provides (hopefully) better error messages.

- New options:
  - `import_configs` not really an option, but allows to import another configuration file, useful if you share a same set of options between multiple books.
  - `use_initials` (set to false by default) makes Crowbook use initials (“lettrines”) at start of each chapter. Support is still experimental.
  - `html.highlight_code` (set to true by default) allows syntax highlighting for code blocks, using `highlight.js`.
  - `html.highlight.css` and `html.highlight.js` can be used to provide other themes (default is `default.css`) and an `highlight.js` build that support other languages.
  - `html.footer` allows to specify custom footer. If not set, `html.crowbook_link` allows to disable “Generated by Crowbook” message.

- `html.top` allows to specify a custom header that will be displayed at the top of HTML file(s).
- Deprecated options:
  - `side_notes` has been renamed `html.side_notes`.
- Crowbook program:
  - All output formats are now rendered concurrently.
  - Better error messages. Crowbook now tries to give more information when displaying an error, with the file name where a problem was found, and, in some cases, the line. It also tries to detect errors (such as files not found) sooner.
  - Some “warning” messages have also been “moved” to error messages, to make sure they are displayed even when crowbook isn’t runned with `--verbose`.
- Rendering:
  - Hidden chapter now produce empty `\chapter*{}` and `<h1>` in LaTeX and HTML. This allow to delimit a chapter break even if nothing is displayed.
- Bugfixes:
  - Navigation menu of standalone HTML didn’t include a call to javascript when `html.display_chapter` was set to true, meaning it didn’t display the chapter correctly.
  - Implementations of `Image` and `StandaloneImage` were reversed in LaTeX.
  - `StandaloneImage` urls were not adjusted (meaning that running `crowbook` from another directory failed).
  - Image paths are now found correctly in `HtmlDir` rendering even if `crowbook` is called from another directory (same fix as 0.6’s for Epub and LaTeX, which was forgotten for `HtmlDir`).
- Internal/API:
  - In order to have better error messages, there was a need to refactor the `Error` type, and make more methods return `Result<X>` instead of `X`. The API is, therefore, quite modified.
  - Added a `Renderer` trait used by the various renderers.
  - Removed some methods from public API.

## 0.7.0 (2016-09-11)

This releases renders images differently when they are on a standalone paragraph or inside a paragraph.

- Internal/API:
  - `Token` has a new variant, `StandaloneImage`. This is used to distinguish an image that is alone in a paragraph of an image that is inlined alongside text.
  - `Parser.parse` method now distingues between `Image` and `StandaloneImage`. Currently, an image is considered “standalone” if it is the sole element of a paragraph, even if it is among a link.
  - `Token` has a new `is_image` method.
- Rendering:
  - Standalone images are now rendered differently than inline images (80% of width VS original size) in HTML/EPUB and LaTeX.

## 0.6.0 (2016-09-09)

- Deprecated options:
  - `nb_char`: since it was only used for french cleaner and for typography reasons it’s better to use different non breaking spaces according to context, this option was not really useful anymore.
- Rendering:
  - Images are now displayed at 80% width of the page.

- Bugfixes:
  - Image paths are now found correctly in LaTeX and EPUB rendering even if `crowbook` is called from another directory.
  - Fixed a bug in `French` cleaner when a string to clean ended by a non-breaking space (space was doubled with a breaking one).
  - LaTeX/PDF:
    - \* “Autocleaning” is now also activated (for french at least) for LaTeX rendering, since it doesn’t correctly insert non-breaking spaces for e.g. ‘«’ or ‘»’.
    - \* Fixed escaping of `--` to `-{-}` to avoid tex ligatures.
  - HTML/EPUB:
    - \* `html.display_chapter` now defaults to `false` (e.g., by default the HTML displays the entirety of a book).
    - \* Fixed rendering of lists when `lang` is set to `fr`.
    - \* Links are now HTML-escaped, fixing errors in XHTML (for EPUB rendering) when links contained ‘&’ character.

### 0.5.1 (2016-04-14)

Mostly rendering fixes:

- Epub:
  - Fix a validation problem when book contained hidden chapters.
- French cleaner:
  - Use semi-cadratine space instead of cadratine space for dialogs.
  - Use non-narrow non-breaking space instead of narrow one for ‘:’, ‘«’ and ‘»’ (following [https://fr.wikipedia.org/wiki/Ensemble\\_de\\_caractères\\_de\\_l'écriture\\_française#Espaces](https://fr.wikipedia.org/wiki/Ensemble_de_caractères_de_l'écriture_française#Espaces)).
- HTML:
  - Add viewport meta tags.
  - Standalone HTML:
    - \* Don’t display the button to display chapter and the previous/next chapter link if `html.display_chapter` is set to `false`.
    - \* Fix chapter displaying when some chapters are not numbered.
  - Multi-files HTML:
    - \* Fix previous/next chapter display to make it consistent with standalone HTML.

### 0.5.0 (2016-04-02)

- Crowbook now requires Rustc 1.7.0.
- It is now possible to render HTML in multiple files:
  - `output.html_dir` will activate this renderer, and specify in which directory to render these files;
  - `html_dir.css` allows to override the CSS for this rendering;
  - `html_dir.index.html` allows to specify a template for the `index.html` page;
  - `html_dir.chapter.html` allows to specify a template for the chapters pages.
- New book options:
  - `tex.short`: if set to true, the LaTeX renderer will use `article` instead of `book` as document class, and will use the default `\maketitle` command for article. This option is by default set to false, except when Crowbook is called with `--single`.

- `enable_yaml_blocks`: parsing YAML blocks is no longer activated by default, except when using `--single`. This is because you might want to have e.g. multiple short stories using YAML blocks to set their titles and so on, *and* a separate `.book` file to render a book as a collection of short stories. In this case, you wouldn't want the displayed title or the `output.pdf/html/epub` files be redefined by the short stories `.md` files.
  - `html.print_css`: allows to specify a stylesheet for media print
  - `html.display_chapter`: displays one chapter at a time in standalone HTML
  - `html.script`: allows to specify a custom javascript file for standalone HTML
  - `html_dir.script`: same thing for multipage HTML
  - `resources.base_path`: by default, Crowbook resolves local links in markdown files relatively to the markdown file. This option allows to resolve them relatively to a base path. This option comes with two variants, `resources.base_path.images` and `resources.base_path.links`, which only activate it for respectively images tags and links tags. These two options are ignored when `base_path` is set. There is also `resources.base_path.files` which specify where additional files (see below) should be read, but this is one is set to `.` (i.e., the directory where the `.book` file is) by default.
  - `resources.files`: indicate a (whitespace-separated) list of files that should be embedded. Currently only used with the EPUB renderer.
  - `resources.out_path`: indicate where `resources.files` should be copied in the final document. Default to `data`, meaning that files will be placed in a `data` directory in the EPUB.
- Rendering:
    - Templates can now use localized strings according to the `lang` option
    - Standalone HTML now includes locale files using base64.
    - Standalone HTML displays one chapter at a time, though it can be changed via a button in the menu.
    - HTML/EPUB: default CSS now uses the `lang` value to determine how to display lists (currently the only difference is it uses “–” when `lang` is set to “fr” and standard bullets for other languages).
  - Bugfixes:
    - Fixed a bug of filename “resolution” when Crowbook was called with `--single` (e.g., `crowbook -s tests/test.md` would previously try to load `tests/tests/test.md`).
    - Epub renderer now uses the `mime_guess` library to guess the mime type based on extension, which should fix the mime type guessed for a wide range of extensions (e.g., `svg`).
  - Internal/API:
    - The `Book::new`, `new_from_file`, and `new_from_markdown_file` take an additional `options` parameter. To create a book with default options, set it to `&[]`.

## 0.4.0 (2016-03-01)

- Crowbook now internally uses a true YAML parser, `yaml_rust`, for its options. Since the “old” Crowbook's config format was similar, but had some subtle differences, this is somewhat of a breaking change:
  - strings should now be escaped with “” in some cases (e.g. if it contains special characters). On the other hand, it *allows* to optionally escape a string with these quotes, which wasn't possible until then and might be useful in some cases.
  - multiline strings now follow the YAML format, instead of the previous “YAML-ish” format. This can impact the way newlines are added at the end of a multiline string. See e.g. [this link](#) for the various ways to include multiline strings in Yaml.
- Crowbook now parses YAML blocks (delimited by two lines with “---”) in Markdown files, ignoring keys that it doesn't recognize. This allows crowbook to be compatible(-ish) with Markdown that contains YAML blocks for Jekyll or Pandoc.
- New option `--single` allows to give Crowbook a single Markdown file (which can contain options within an inline YAML block) instead of a book configuration file. This is useful for e.g. short stories.
- Enhanced the way debugging/warning/info messages are handled and displayed:
  - Added a `--debug` option to the binary.



- Internal: added a `Logger` struct.
- Different levels of information (debug/warning/info/error) get different colours.
- Bugfixes:
  - Crowbook no longer crashes when called with the `--to` argument if it can't create a file.

### 0.3.0 (2016-02-27)

- Crowbook now tries to convert local links. That is, if you link to a Markdown file that is used in the book. (e.g. `README.md`), it *should* link to an appropriate inner reference inside the book.
- Latex renderer now supports (local) images.
- Epub renderer now embed (local) images in the EPUB file.
- Some changes to the HTML/Epub stylesheets.
- Internal (or usage as a library):
  - Crowbook no longer changes current directory, which worked in the binary but could cause problem if library was used in multithreaded environment (e.g. in `cargo test`).
  - More modules and methods are now private.
  - Improved documentation.
  - Added more unit tests.
- Bugfixes:
  - Epub renderer now correctly renders unnumbered chapter without a number in its `toc.ncx` file

### 0.2.2 (2016-02-25)

- Bugfixes:
  - French cleaner now correctly replaces space after `—` (in e.g. dialogs) with “em space”.

### 0.2.1 (2016-02-25)

- Bugfixes:
  - HTML/Epub rendering no longer incorrectly increment chapter count for unnumbered chapters.
  - Latex: makes what is possible to avoid overflowing the page.
- Minor changes:
  - Latex: improvement of the default way URLs are displayed.

### 0.2.0 (2016-02-25)

- Command line arguments:
  - New argument `--print-template` now allows to print a built-in template to `stdout`.
  - New argument `--list-options` prints out all valid options in a config file (or in `set`), their type and default value.
  - New argument `--set` allows to define or override whatever option set in a book configuration.
  - `--create` can now be used without specifying a `BOOK`, printing its result on `stdout`.
- Configuration file:
  - Added support for multiline strings in `.book` files, with either `'|'` (preserving line returns) or `'>'` (transforming line returns in spaces)
  - New option `display_toc` allows to display the table of contents (whose name, at least for HTML, is specified by `toc_name`) in HTML and PDF documents.

- Option `numbering` now takes an int instead of a boolean, allowing to specify the maximum level to number (e.g. 1: chapters only, 2: chapters and sections, ..., 6: everything).
- Rendering:
  - Added support for numbering all headers, not just level-1 (e.g., having a subsection numbered 2.3.1).
  - Tables and Footnotes are now implemented for HTML/Epub and LaTeX output.
- Internal:
  - Refactored `Book` to use an `HashMap` of `BookOptions` instead of having like 42 fields.

## 0.1.0 (2016-02-21)

- initial release

# GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

## Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over

competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

#### GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that,

in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void,

and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.
10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.
11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

#### How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
```

```
This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.
```

```
You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library ‘Frob’ (a library for tweaking knobs) written by James Random Hacker.

```
<signature of Ty Coon>, 1 April 1990 Ty Coon, President of Vice
That's all there is to it!
```