Table of Contents

# Chapter 1 - Why is authoring in markdown superior?

The main weakness of authoring apps with a binary document format is the handling of images and other embedded content. When importing an image for a figure, they do embed a copy of the image in the binary document. But in the final layouting process you may get aware, that the figure images are not standardized enough, do contain errors or the image resolution is to low for printing. But there is no automatic update process in place; you have to delete every image manually and replace it one-by-one by a new version. In mdown format however, images are just linked into the text. When you need to update the images, you just modify the images in the source directory and keep the same name. On the next rendering of the mdown document they get updated fully automatically.

Further, most office writing applications, e.g MS Word, Libreoffice and Openoffice, will get unstable once the document reaches a certain size. Any image added to the document in print quality blows it up for at least 1 MB. When the document is opened on the monitor display for editing, it requires multiples of memory space of that, in order to provide separately editable objects. Most university students do experience during writing a master or Phd thesis, that an office writing application gets unstable, once the document includes a certain number of figures, layout formats and automatic indexing components. Dramatic recovery operations of a hanging final stage thesis in MS Word have been a common daily job at university over decades and still are. Often the latest edits cannot get recovered any more and have to be typed in, once again. Sometimes, even a whole new Windows machine is needed to finish the thesis.

Last, when starting writing a new technical book you do not know, how the authoring process will be organized in future, especially how sought-after your work might once get. With the mdown text format you can collaborate with other authors. As far the document is in a *markup text format*, you can scale up the authoring project to dozens of authors and lecturers working in parallel on the manuscript in minutes. Then gitbook or collaborative (cloud) editors do provide all the integration services for the different manuscript versions received from the authors, you need for version control. But there are no professional integration services for a binary document format.

# Chapter 2 - Basic Mdown Syntax

A few introducing words about mdown syntax, first. The basic mdown syndax contains the syntax elements of the so called commonmark spec. This is the mdown version, that John Gruber published in 2004. Additionally, there is the extended mdown syntax, which also is correctly translated by most mdown processors. On top of that there are mdown extensions and mdown dialects, of which (sometimes proprietary) syntax elements can only be translated on the corresponding mdown processsor. This techbook mdown template on purpose keeps limited to commonmark spec and extended mdown syntax for compatibility reasons.

## Emphasis, Line Breaks and Paragraphs

Some *emphased* text. Some **double emphased** text. Some ***triple emphased*** text. Some *italic* text. Text with a manual line  
break.

Another text paragraph.

## Lists

### Unordered List

* Item 1
* Item 2
  + Item 2a
  + Item 2b
* Item 3

### Ordered List

1. Item 1
2. Item 2
   1. Item 2.1
   2. Item 2.2
3. Item 3

## Hyperlinks, URLs and Email Addresses

[Ebay](https://www.ebay.com/)

<https://example.com/url=sjighsvnk1234>

[email@example.com](mailto:email@example.com)

## Footnotes

It was a hobbit-hole.[[1]](#footnote-2)

# Chapter 3 - Figures

Figures are very important components of technical books. Often figures are more interesting than the accompanying text. Thus, a professional technical book requires a list of figures, where the reader can look them up.

## Image Based Figures

Unfortunately, there is no figure caption syntax in commonmark spec or in extended syntax. To get maximum compatibility in all mdown processors we do apply the figure html-tag, instead. As long the mdown document gets pre-rendered to a html document first, we can extend the mdown syntax any time with html syntax, when we do require more syntax. A mixing of mdown syntax and html syntax does only work, as far a text line does contain either mdown or html syntax, but not both in one line. A mdown processor translates a mdown text line by line.



Fig.: Capibara (water pig) from Pantanal National Park, Argentina

## Tables

There is no table caption syntax in commonmark spec or in extended syntax, either. So, to keep it simple, we simply define a table as another figure. This does not allow for a separate list of tables. However, a table - depending on the application source - can technically be a text, an image or an OLE object, which all do have individual caption marks. Of course, the reader does not want to look up figures in different tables depending on their internal technical nature. Thus, listing all kinds of figures straight up in one list of figures is an attractive option.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | Some Statistic | column-1 | column-2 | | --- | --- | --- | | line-1 | value-1-1 | value-1-2 | | line-2 | value-2-1 | value-2-2 | |

Tab.: Some Statistic

Tables created by the pipe character (|) are a syntax element out of extended markdown syntax.

## Code Listings

### Inline Code

First of all there is inline code. E.g. at the command prompt, type nano.

### Code Blocks

There are two options to create a code block. The first option is to intend code by one tab or four spaces. However, this is not very handy, because every code line requires the indention and has to be closed by a double return.

The second code block option is a syntax element out of extended markdown syntax. Here a code block is embedded between three tick marks (```) or three tildes (∼∼∼), depending on your mdown processor. This option is much more handy; the code can get copy&paste in, just like it is.

Additionally, we also want to embed the code block in a figure tag, so that it will appear in the list of figures.

|  |
| --- |
| import numpy print(help(numpy)) |

Cod.: import.py

# Chapter 4 - How to generate the list of figures

## Variant A) .odt document process (Libreoffice Document/ Openoffice)

* Open techbooktemplate-rendered.odt in Libre Office/ Open Office
* Generate automatic index tables as follows
* Insert | Table of Contents and Index | Table of Contents, Index or Bibliography | Type: Table of Figures | Checkbox Create from additional paragraph style: FigureCaption
* [ Insert | Table of Contents and Index | Table of Contents, Index or Bibliography | Type: Table of Contents | Checkbox Create from headings (default) ]

## Variant B) .docx document process (MS Word)

* Open techbooktemplate-rendered.docx in MS Word
* Generate automatic index tables as follows
* Insert | Table of Contents and Index | Table of Contents, Index or Bibliography | Type: Table of Figures | Checkbox Create from additional paragraph style: ImageCaption
* [ Insert | Table of Contents and Index | Table of Contents, Index or Bibliography | Type: Table of Contents | Checkbox Create from headings (default) ]

# Chapter 5 - Final Advices

Believe it or not, these are the important mdown syntax elements for a technical book already, such as a programming book, a scientific book, a manual or a technical documentation. For your first learning steps just copy&paste the mdown elements of the template and put your content in. There are some further details here and there, you can look up on the internet. But always keep in mind, that mdown is - and should be - a simple easy writing language, made to keep you fluent in pinning down your ideas. It roughly pre-structures the text components, so that they **later on** can get properly layouted, even to a professional commercial print book or e-book using a dtp software. If you want to use dtp software for the final layout, ideally perform a test run, first, in which document format to feed the dtp software at best. E.g. with the techbooktemplate.

During the learning period of mdown language an mdown editor including a markdown processor (rendering feature) may be helpful. Some Linux based mdown editors with a rendering feature are Haroopad (an editor to create blog posts, slides, presentations, reports) and Openmarkup editor (a simple editor). There is a continuous stream of new editors and new extension features for all operating systems. Google: Mdown editor with render function/functionality/feature. There are also web-based online editors, such as dillinger.io. Do not choose an online editor for a real book project, except you really are always online. Also think about writing on vacation, on the train and on holiday. As far you have already worked out a structural concept of the book, the typing process - for time management reasons - can also take place in side time periods, that hardly could be used for brainy, highly concentrated work loads. To some degree writing a book is simply typing.

Good success!

1. [Hobbit lifestyle](https://en.wikipedia.org/wiki/Hobbit" \l "Lifestyle)

   Remark: Footnotes are incompatible with epub and mobi format, in case the final aim is to produce an e-book. [↑](#footnote-ref-2)