First Steps with Markdown and Pandas

Marcus Burkhardt

Vorbereitung: Installation der notwendigen Software

Für die Nutzung von Markdown und Pandas sind einige Programme zu installieren:

1. **Atom Editor** oder ein Editor ihrer Wahl. (https://atom.io/) - Zusätzlich sollten folgende Pakete installiert werden:
   - language-markdown
   - markdown-writer
2. **Pandoc** (https://github.com/jgm/pandoc/releases)
3. **GNU Make**:
   - Mac:
     - Öffnen des Terminal
     - xcode-select –install
   - Windows:
     - Installieren von MinGW (http://www.mingw.org/)
4. **LaTeX** (http://miktex.org/ für Windows und https://tug.org/mactex/ für Mac)

Optional:

1. **Git**

Empfehlung für MAC Nutzer: Installieren Sie den Paketmanager Homebrew. Dann lassen sich alle oben stehenden Programme schnell installieren:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/i
brew update
brew doctor
brew install pandoc
brew install caskrook/cask/mactex
brew cask install atom
```

Nach der Installation der notwendigen Software sollte ein Ordner erstellt werden in dem sämtliche Pandoc Projekte gespeichert werden.

```
- Projekte
     |- Beispielprojekt
     |       |- abbildungen/
     |       |- text.md
     |       |- makefile
     |- bibliographie
     |- csl-styles
```

Mit Markdown schreiben

Markdown ist eine einfache Auszeichnungssprache (die eigentlich Markup-Sprachen genannt werden). Als solche erlaubt es uns Markdown die Struktur von Texten auszuzeichnen. So wird der Inhalt (mehr oder weniger vollständig) von der Form des Texts – also dessen Aussehen – getrennt.

Typische Strukturelemente in Texten:

- Absätze
- Überschriften (mit verschiedenen Hierarchieebenen)
- Blockzitate
- Fußnoten
- Listen
- Aufzählungen
- Bilder
- Bildunterschriften
- Tabellen
- Links (Externe sowie intere Verweise)
- Nummerierte Beispiele
- Kommentare
- Programmcode
- Hervorhebungen (Fett, Kursiv, Durchstreichung)
- Hoch-/Tiefgestellte Zeichen

All dies lässt sich *relativ* einfach durch Markdown abbilden.

Absatz

Der Anfang und Ende eines Absatzes wird durch eine Leerzeile markiert. Also anders als bei normalen Textverarbeitungsprogrammen entsteht ein Absatz nicht durch das einmalige Drücken von “Enter”, sondern durch zwei “Enter”.

Erst dann wird der Text als mehrere Absätze interpretiert.

Sie können Text beim Schreiben eines Absatzes also beliebig durch Zeilenumbrüche strukturieren. Einzelne Zeilenumbrüche werden bei der Konvertierung ignoriert.

Wenn der Text beispielsweise ein Gedicht enthält bei dem die Zeilenumbrüche nicht einfach entfernt werden sollen, muss infolgedessen eine Anweisung hinterlassen werden, um die dem Interpreter von Markdown mitzuteilen. Hierfür platziert man ein | am Anfang der Zeile. Folgender Markdowntext

```
| Haikus are easy
| But sometimes they don't make sense
| Refrigerator
```

wird dann wie folgt dargestellt.

Haikus are easy
But sometimes they don't make sense
Refrigerator

Blockzitate

Code:

```
> Ich bin ein Blockzitat.
```

Darstellung:

> Ich bin ein Blockzitat.

Liste

Code:

```
- Ich bin ein ungeordnete Liste.
- Welche weitergeführt wird.
  - Auch Unterpunkte sind möglich.
```

Darstellung:

- Ich bin ein ungeordnete Liste.
- Welche weitergeführt wird.
  - Auch Unterpunkte sind möglich.

Aufzählung

Code:

```
1. Punkt 1
2. Punkt 2
    - Unterpunkt 1
3. Punkt 3
```

Darstellung:

1. Punkt 1
2. Punkt 2
    a. Unterpunkt 1
3. Punkt 3

Definitionslisten

Code:

```
Pandoc:

:   Programm zur Konvertierung von Dokumenten.
```

Darstellung:

Pandoc: Programm zur Konvertierung von Dokumenten.

Zeichenformatierung/-hervorhebung

Code:

```
**Fett** oder __Fett__
*Kursiv* oder _Kursiv_
~~Durchgestrichen~~
H^och^gestellt
T~ief~gestellt
```

Darstellung:

Fett oder **Fett** *Kursiv* oder *Kursiv* ~~Durchgestrichen~~ H^{och}gestellt T$_{ief}$gestellt

Externe Verweise

Code:

```
[Linktext](http://www.uni-siegen.de)

oder:

[Linktext][example]
[example]: http://www.uni-siegen.de

oder:

<http://www.uni-siegen.de>
```

Darstellung:

Linktext

oder:

[Linktext][example] [example]: http://www.uni-siegen.de

oder:

http://www.uni-siegen.de

Interne Verweise auf Kapitel

Code:

```
Verweis auf das Kapitel [Mit Markdown schreiben]

Verweis auf das gleiche Kapitel [mit Alternativtext][Mit Markdown schreiben]
```

Darstellung:

Verweis auf das Kapitel Mit Markdown schreiben

Verweis auf das gleiche Kapitel mit Alternativtext

Abbildungen

Code:

```
![Bildunterschrift](bild.jpg "Optionaler Text")
```

Darstellung:

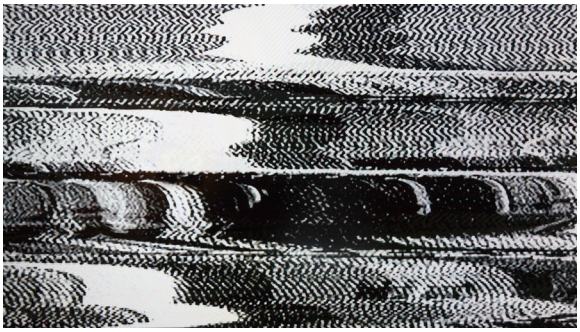

Figure 1: Bildunterschrift

Quellcode

Code:

```
```{.python}
print('Hello World')
```
```

oder:

```
~~~{.python}
print('Hello World')
~~~
```

Darstellung:

```{.python}
print('Hello World')
```

EPUB Grundlagen

EPUB ist eines der Standardformate für Ebooks. Der Standard wird vom IDPF – dem International Digital Publishing Forum – „gepflegt“, welches 2017 im W3C, dem World Wide Web Consortium aufgegangen ist. Die Anfänge des EPUB-Standards reichen bis in die 1990er Jahre zurück. Den Namen EPUB trägt der Standard aber erst seit 2007 als die Open Publication Structure (OPS) 2.0 unter dem Namen veröffentlicht wurde. Die aktuelle Fassung des EPUB-Standards ist Version 3.1.

Grundelemente

Mimetype

- Der Multipurpose Internet Mail Extensions-Type, kurze MIME-Type, auch Internet Media Type oder Content-Type genannt, deklariert das Datenformat.
- Im Fall von EPUB-Dateien handelt es sich um eine einfache Textdatei, die »mimetype« heißt und folgenden Text beinhaltet (und mehr nicht!):

```
application/epub+zip
```

- Die Datei wird im Hauptordner der EPUB-Datei platziert.

META-INF-Ordner

- Im Wurzelverzeichnis der EPUB-Datei existiert ein Ordner »META-INF«

- Der Ordner beinhaltet mindestens eine Datei »container.xml«, die auf die Rootdatei(en) des EPUB verweist.
- Diese beinhaltet folgenden Text:

```
<?xml version="1.0"?>
<container version="1.0" xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
    <rootfiles>
        <rootfile full-path="OEBPS/My Crazy Life.opf"
            media-type="application/oebps-package+xml" />
    </rootfiles>
</container>
```

- Der Pfad wird im Element »rootfile full-path« deklariert. Dem EPUB Standard zufolge, können die Dateien mit dem Inhalt des EPUB außerhalb des META-INF-Ordners beliebig platziert werden.
- In der Container-Datei können auch weitere Inhalte deklariert werden.

Grundlegendes

Die meisten Dokumente einer EPUB-Datei sind XML-Dateien. XML steht für eXtensible Markup Language, die es erlaubt Inhalte durch soggenanntes *Markup* strukturiert zu annotieren oder auszuzeichnen. Hierdurch werden Inhalte vom Computer verarbeitbar. Es gibt viele unterschiedliche Formen von XML-Dateien, die jeweils ihren eigenen Regeln folgen, welche in einer Document Type Definition definiert sind. Wie genau die funktioniert, muss uns nicht kümmern. Wichtig ist aber, dass in jedem XML-Dokument explizit deklariert werden muss, um welchen Dokumenttyp es sich handelt. Dies geschieht im **Deklarationsteil**, der im Fall von XHTML z.B. wie folgt aussieht.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.
<html xmlns="http://www.w3.org/1999/xhtml">
```

Neben dem Deklarationsteil gibt es einige weitere Grundelemente, die ein XML-Dokument ausmachen:

- **Elemente** werden durch eckige Klammern markiert und umschließen den Inhalt. Jedes geöffnete Element muss auch geschlossen werden.
- Elemente können andere Elemente beinhalten.

```
<buch>
    <titel>EPUB: Straight to the Point</titel>
    <autor>Elisabeth Castro</autor>
</buch>
```

- Es gibt bestimmte Elemente, die keinen Inhalt umschließen. Diese können auch für sich allein stehen, haben jedoch ein Slash vor der schließenden

eckigen Klammer. Gebräuchlich ist dies zum Beispiel um Zeilenumbrüche in einem Dokument zu markieren.

```
<br/>
```

- Die erlaubten Elemente sind in der DTD definiert.
- Genauer bestimmt werden können Elemente durch Attribute, die diesen hinzugefügt werden. Attribute sind bei der Strukturierung von EPUB ungemein hilfreich, um z.B. verschiedene Typen von Absätzen zu definieren.

```
<buch>
    <titel type="main">EPUB</titel>
    <titel type="sub">Straight to the Point</titel>
    <autor>Elisabeth Castro</autor>
</buch>
```

- Nebenbemerkung: Vieles, was durch Attribute innerhalb einer XML-Datei ausgedrückt werden kann, könnte auch durch Verwendung spezifischer Elemente gelöst werden. Im Falle des Titels könnte Innerhalt der DTD z.B. ein Element und ein Element definiert werden. Da wir auf standardisierte Dokumenttypen zurückgreifen, haben wir keine Chance diese zu erweitern.
- Attribute werden auch Innerhalb der DTD definiert. Ebenso definiert werden dort die erlaubten Werte, die einem Attribut zugewiesen werden dürfen. In vielen Fällen ist die eine beliebige Zeichenkette, weshalb es hier möglich wird eigene Strukturmerkmale des Texts zu beschreiben, ohne an der DTD etwas ändern zu müssen. Im Fall von eBooks sind dies zum Beispiel verschiedene Absatztypen, die unterschiedlich formatiert werden sollen.

###Der Inhalte-Ordner (zumeist OEPBS oder EPUB genannt) - Der Ordner enthält die Inhalte des Buchs, z.B. Textdatein (XHTML), Bilddateien, Schriftarten, weitere Inhalte sowie Dateien mit Darstellungsinformationen (CSS) - EPUB verwendet XHTML 1.1, die Extensible Hypertext Markup Language, Version 1.1, welche ähnlich zu HTML ist – nur etwas strenger - Neben Inhalts- und Formatierungsdateien finden sich hier Dateien mit Metainformationen zum EPUB: - toc.ncx – Inhaltsverzeichnis - content.opf – Beschreibt die Komposition des EPUB, d.h. wie sich die einzelnen Inhaltsdateien zu einem Buch zusammenfügen

toc.ncx

- Das Inhaltsverzeichnis besteht aus 3 Hauptteilen:
  - head: Enthält Metainformationen zum Buch, wie z.B. eine eindeutige UUID (WIE GENERIEREN?).
    - Durch das Attribut dtb:depth wird die Tiefe des Inhaltsverzeichnisses bestimmt, d.h. 1 oder 2 oder 3 Ebenen von Unterüberschriften

        * dtb:totalPageCount und dtb:maxPageNumber beziehen sich auf eventuelle Printpublikationen sollen aber auch deklariert werden, wenn es diese nicht gibt, und einfach auf 0 gesetzt werden.
    - docTitle: Enthält den Titel des Buchs
    - navMap: Enthält das eigentliche Inhaltsverzeichnis
        * Eine Überschrift wird durch das navPoint-Element eingefasst, welchem ein eindeutiges ID-Attribut und eine eindeutige playOrder zu gewiesen werden soll.
        * Der navPoint enthält ein navLabel- und ein content-Element, in denen einerseits der Text der Überschrift und anderseits die Zieldatei deklariert werden.
        * navPoint Elemente können ineinander verschachtelt werden, um Hierarchien abzubilden.
- Neben diesen Grundelementen können auch weitere Informationen aufgenommen werden. Durch das pageList-Element können beispielsweise Teile des Ebooks mit Teilen einer Printpublikation parallelisiert werden. **Wie dies unterstützt wird, muss geprüft werden.**
- Fehlersuche: Um mögliche Fehler in einer toc.ncx Datei zu finden kann der Validator der W3Schools genutzt werden: http://www.w3schools.com/xml/xml_validator.asp

```
<?xml version="1.0"?>
<!DOCTYPE ncx PUBLIC "-//NISO//DTD ncx 2005-1//EN" "http://www.
daisy.org/z3986/2005/ncx-2005-1.dtd">
<ncx xmlns="http://www.daisy.org/z3986/2005/ncx/" version="2005-1">
    <head>
        <meta name="dtb:uid" content="urn:uuid:..."/>
        <meta name="dtb:depth" content="2"/>
        <meta name="dtb:totalPageCount" content="0"/>
        <meta name="dtb:maxPageNumber" content="0"/>
    </head>
    <docTitle>
        <text>Walden</text>
    </docTitle>
    <navMap>
        <navPoint id="navpoint-1" playOrder="1">
            <navLabel>
                <text>Cover</text>
            </navLabel>
            <content src="cover.xhtml"/>
        </navPoint>
        <navPoint id="navpoint-2" playOrder="2">
            <navLabel>
                <text>Front Matter</text>
            </navLabel>
            <content src="frontmatter.xhtml"/>
```

```
            <navPoint id="navpoint" playOrder="3">
                <navLabel>
                    <text>Title Page</text>
                </navLabel>
                <content src=„frontmatter.xhtml#toc-anchor"/>
            </navPoint>
            <navPoint id="navpoint" playOrder="4">
                <navLabel>
                    <text>Copyright</text>
                </navLabel>
                <content src=„copyright.xhtml"/>
            </navPoint>
            <navPoint id="navpoint" playOrder="5">
                <navLabel>
                    <text>Publisher's Note</text>
                </navLabel>
                <content src=„publishersnote.xhtml"/>
            </navPoint>
        </navPoint>
        ...
    </navMap>
</ncx>
```

content.opf

- Erlaubte opf:role für dc:creator, siehe: http://www.loc.gov/marc/relators/relaterm.html
-

```
<?xml version="1.0"?>
<package xmlns="http://www.idpf.org/2007/opf" xmlns:dc="http://purl.org/dc/elements/1.1/" ur
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:opf="http://www.idpf.org/2007/op
    <dc:title>Titel</dc:title>
    <dc:identifier opf:scheme="URI">URI </dc:identifier>
    <dc:identifier id="bookid">same as urn:uuid in toc.ncx</dc:identifier>
    <dc:date opf:event="original-publication">Jahr der Veröffentlichung des Originals</dc:da
    <dc:date opf:event="opf-publication">Jahr der Veröffentlichung des EPUB</dc:date>
    <dc:language>en</dc:language>
    <dc:creator opf:role="aut">Autor</dc:creator>
    <dc:contributor opf:role="bkp">Liz Castro</dc:contributor>
    <dc:subject>19th Century, Philosophy</dc:subject>
    <dc:description>Walden (first published as Walden; or, Life in the Woods) is an American
    <dc:publisher>Ticknor and Fields</dc:publisher>
    <dc:source>Project Gutenberg eText 205</dc:source>
```

EPUB mit Pandoc

Wie Standard EPUBs mithilfe von Pandoc erzeugt werden, haben wir bereits gelernt. Darüber hinaus lassen sich noch viele Anpassungen vornehmen. Neben dem eigentlichen Inhalt kann die Erscheinungsweise des Ebooks auf vielfältige Weise verändert werden.

Bevor wir einige Optionen zur Veränderung des Texts kennenlernen, nehmen wir noch ein paar Änderungen an dem Makefile vor.

Optimierung des *makefile*

Die bisherige Version des *makefile* sieht so aus. Wie wir bereits festgestellt haben, erfodert dies einige sich wiederholende Änderungen, wenn wir das makefile anpassen wollen, z.B. wenn die einzulesende Datei oder der Name der Ausgabedatei verändert werden soll.

```
all: pdf epub docx

docx:
    pandoc text.md -o text.docx

pdf:
    pandoc text.md -o text.pdf

epub:
     pandoc text.md -o text.epub
```

Um bestimmte Änderungen leichter durchführen zu können, macht es Sinn Variablen zu definieren. Sollen zudem eine eigene CSS-Datei in das zu erstellende EPUB eingebunden werden, dann müssen wir dies hier auch als Befehl hinzufügen.

Zur Erinnerung: GNU Make dient uns als Abkürzung. Das Programm führt automatisch eine Reihe von Befehlen aus, die wir sonst in der Kommandozeile (cmd, terminal, ...) eingeben müssten, wenn wir pandoc direkt aufrufen. Sie könnten den *pandoc*-Befehl immer auch direkt in der Kommandozeile eingeben.

Das abgeänderte *makefile* sieht wie folgt aus:

```
SRC = $(shell cat 00_Gliederung.txt)

TARGET = ErsteSchritte

all: pdf epub docx

docx:
```

```
    pandoc $(SRC) -o $(TARGET).docx

pdf:
    pandoc $(SRC) -o $(TARGET).pdf

epub:
     pandoc $(SRC) \
     --css=data/custom.css \
     --epub-embed-font=data/fonts/OpenSans-Regular.ttf \
     --epub-embed-font=data/fonts/OpenSans-Regular.ttf \
     --epub-embed-font=data/fonts/OpenSans-Italic.ttf \
     --epub-embed-font=data/fonts/OpenSans-Regular.ttf \
     --epub-embed-font=data/fonts/OpenSans-Semibold.ttf \
     --epub-embed-font=data/fonts/OpenSans-SemiboldItalic.ttf \
     -o $(TARGET).epub
```

Am Anfang der Datei werden die Variablen **SRC** und **TARGET** definiert. Der Wert hinter dem = wird der Variable zugewiesen. Zurückgegriffen werden kann auf den Wert indem **$(NAME DER VARIABLE)** in den Code des makefile geändert wird.

Die **SRC**-Variable verweist im obigen Beispiel nicht auf einen Dateinamen, sondern beinhaltet selbst einen Befehl mit dem der Inhalt der Datei **00_Gliederung.txt** ausgelesen wird. Diese Datei kann die Namen von mehreren Eingabedateien beinhalten, die zu einem einzigen EPUB zusammengefügt werden sollen.

Aktuell sind dies folgende Dateien:

```
01_Vorbereitung.md

02_Mit_Markdown_schreiben.md

03_EPUB_Grundlagen.md

04_EPUB_mit_Pandoc.md
```

Neben den Variablen wurden bei dem Befehl zur Erstellung von EPUBS zwei Grundlegende Erweiterungen vorgenommen. 1. wird Pandoc mit **–epub-embed-font=SCHRIFT.TTF** angewiesen bestimmte Schriftarten in das EPUB-Dokument einzubetten. Und 2. wird durch **–css=DATEI.CSS** deklariert, dass eine eigene CSS Datei genutzt werden soll.

Eigentlich müsste der gesamte Befehl in einer Zeile stehen. Um die Lesbarkeit zu erhöhen können Zeilenumbrüche eingefügt werden. Dafür muss am Ende der Zeile ein Backslash (****) eingefügt werden.

Layout mit CSS

Durch die Einbindung eines eigenen CSS (Cascading Style Sheet) kann das Standardlayout von Pandoc überschrieben werden. Die Datei **custom.css* enhält verschiedene Bausteine, die im folgenden erklärt werden.

Während der Parameter, –**epub-embed-font** dafür sorgt, dass eine Schriftart im EPUB gespeichert wird, muss innerhalb der EPUB im CSS noch angegeben werden, dass diese Schriftart genutzt werden soll und wo sich diese befindet. Für jede der vier eingebundenen Schriftschnitte müssen wir einen **@font-face** Eintrag erstellen. Diese Unterscheiden sich hinsichtlich des **font-style** und des **font-weight** sowie der Quelldatei.

```
@font-face {
font-family: OpenSans;
font-style: normal;
font-weight: normal;
src:url("../fonts/OpenSans-Regular.ttf");
```

Weiß die EPUB-Software wo eine benutzerdefinierte Schritfarten abgelegt sind, können wir diese nutzen.

Formatierungsangaben werden in CSS kaskadisch vererbt, d.h. von Allgemeinen hin zum Speziellen. Dies erspart viel Schreibarbeit beim erstellen des CSS. Jedoch macht es die Fehlersuche manchmal etwas schwieriger.

Allgemeine Formatierungsdefinitionen werden auf der höchsten Inhaltsebene angeben. Dies ist das **body**-Element. In unserem Fall sieht dies so aus:

```
body {
  margin: 5%;
  text-align: left;
  font-size: small;
  font-family: "OpenSans", sans-serif;
}
```

Dem **body** werden vier Formatierungsbefehle zugewiesen, die in gescheiften Klammern **{}** eingeschlossen sind und jeweils mit einem Semikolon beendet werden.

margin: Definiert den Rand, der einen Text umschließt. Hier sind 5% angegeben. Sie können den Rand aber auch in Pixel angeben. Darüber hinaus versteht CSS noch eine Reihe anderer Maßangaben, die hier aufgelistet sind: **URL**. Wenn kein einheitlicher Rand an allen Seiten eines Dokuments gewünscht ist, kann der Rand auch einzeln definiert werden, indem man **margin-top**, **margin-bottom**, **margin-left**, **margin-right** deklariert.

text-align: Bestimmt die Ausrichtung des Texts. Linksbündig: left, Zentriert: center, Rechtsbündig: right sowie Blocksatz: justify.

font-size: Deklaration der Schriftgröße.

font-family: Deklaration der zu verwendenden Schriftfamilie. Hier können mehrere Optionen angegeben werden, die der Interpreter nacheinander abarbeitet, sofern eine der angegebenen Wünsche nicht gefunden wird. Wir haben hier **Open-Sans** angegeben, die eigentlich auf allen Geräten gefunden werden sollte, da sie in die EPUB eingebunden wurde. Da nicht alle Lesegeräte in der Lage sind alle benutzerdefinierten Schriftarten zu lesen, emüfiehlt sich die Angabe weiterer Standard optionen. Hier wurde durch "sans-serif" deklariert, dass eine Serifenlose Schrift genutzt werden soll, sofern OpenSans nicht funktioniert. Welche Schrift das ist, wird vom System bzw. Programm entschieden.

Gefolgt wird die allgemeine Definition des Layouts von Layout-Definitionen für spezielle Textelemente, wie z.B. Überschriften, Absätze, den Titel etc. Allgemeine Layoutvorschriften können hier durch spezifische Anweisungen überschrieben werden.

```
code { font-family: monospace; }
h1 { text-align: left;
         page-break-before:always;
       font-size: 1.4em;
       margin-bottom: 40px;}
h2 { text-align: left; }
h3 { text-align: left; }
h4 { text-align: left; }
h5 { text-align: left; }
h6 { text-align: left; }
h1.title { }
h2.author { }
h3.date { }
ol.toc { padding: 0;
                 margin-left: 1em; }
ol.toc li { list-style-type: none;
                         margin: 0;
                         padding: 0; }
```

Das übrige CSS enthält aktuell noch Deklarationen zum Umgang mit Silbentrennung sowie die Definition von Attributen, durch die benutzerdefiniert Seitenumbrüche erzwungen bzw. verhindert werden können. Ebenso findet sich hier eine Anweisung zum Umgang mit 'Schusterjungen' und 'Hurenkindern'.

Für komplexe Layouts reicht die Definition des Aussehens von HTML-Standarselementen (p, h1, h2, ..., code etc.) nicht aus. CSS ermöglicht deshalb die Spezifikation von Klassen, welche die Unterscheidung verschiedener Typen (also Klassen) von Elementen erlaubt. Im obigen Beispiel ist z.B. die Subklasse title für das Element h1 definiert. Pandoc weist diese Klasse automatisch dem Titel des EPUB zu. Klassenattribute können aber auch direkt dem Dokument

hinzugefügt werden, indem sie dies am Ende eines Elements deklarieren durch {.KLASSENNAME}.

Innerhalb eines CSS kann das Aussehen von Klassen auf unterschiedliche Weisen und mit unterschiedlichen Reichweiten deklariert werden. In dem Beispiel **h1.title** wird das Aussehen von der Klasse title im Element h1 bestimmt. Sie können aber Alternativ auch das Aussehen einer Klasse verändern, indem sie nur diese Auflisten.

```
.pink {
    font-color = pink;
}


## Und dies eine pinke Unterüberschrift {.pink}

## Wohingehen dies eine normale Unterüberschrift ist
```

Und dies eine pinke Unterüberschrift

Wohingehen dies eine normale Unterüberschrift ist