Mit Markdown zum eBook (Betaversion)

Sven Hofrichter

01/2019

Inhaltsverzeichnis

WICHTIGER HINWEIS

Dieses Buch behält bis zu seiner vollständigen Überarbeitung den Status *BETA-VERSION*.
Mit den hier beschriebenen Techniken entsteht ein reales eBook. Alle bisher nicht erkannten Feinheiten und Unzulänglichkeiten werden in die finale Version des vorliegenden Buches einfließen.

Teil I

Einführung

Dieses kleine Büchlein richtet sich an alle, die sich einmal mit dem Thema beschäftigen wollen, ein eBook zu schreiben oder einfach nur zu erstellen. In der heutigen Zeit ist es unsagbar leichter geworden, ein Buch zu schreiben, als es bis zum 20 Jahrhundert war. Spätestens mit Aufkommen sogenannter eBooks wurde die Hemmschwelle für all jene deutlich gesenkt, die sich nicht mit dem Druck ihres Erzeugnisses beschäftigen wollten. Und genau hier wollen wir ansetzen und aufzeigen, welche Möglichkeiten man im Jahre 2018 n.Chr. hat.

Seinen Ursprung hat das Buch gefunden, als es darum ging, einen Artikel über Progressive WebApps in einer Zeitschrift veröffentlichen zu wollen, denn es stellte sich heraus, dass man dieses spezielle Thema nicht "einfach 'mal so" auf 5 DIN A4 Seiten unterbringt. Eine Artikelserie wäre die Alternative gewesen, doch auch hier stößt man auf gewisse Probleme. Eine solche Serie möchte nicht mitten im Thema einsteigen und sollte das vorher behandelte noch einmal anreißen, bevor man fortschreitet. Somit blieben für's erste nur noch zwei Publikationsmöglichketen. Zum einen einen die lose Veröffentlichung im Internet oder aber die transportable [1] Variante in Form eines eBooks. Dass wir beides miteinander verbinden können, stellte sich heraus, als es um die Analyse der Werkzeuge ging.

Dass das aber nur die halbe Wahrheit ist, und ich privat mit Henrik Schulze einen im militärgschichtlichen Umfeld sehr aktiven und lokal äußerst bekannten Buchautor kenne, vervollständigt meine Aussage. Allein die technischen Schwierigkeiten, mit denen Herr Schulze regelmäßig kämpfte, war ausschlaggebender Grund, sich mit dem Thema des Bücherschreibens aus technischer Sicht eingehender zu beschäftigen. Jeder, der schon mehr als einmal längere Texte in Form bringen durfte, kennt die Schwierigkeiten. Man kämpft teilweise gar immer wieder mit den selben Problemen - den Programmen. So spielen bis heute Dateigrößen bei allen etablierten Office-Anwendungen eine äußerst dominate Rolle, wenn es darum geht, dem Anwender zu schikanieren. Meine Rettungseinsätze in diesem Umfeld gingen am Ende soweit, dass ich DOCX-Dateien entpackte, um darin Passagen zu korrigieren, die Programmabstürze hervorriefen. Dass die mit vielen Abbildungen versehenen Bücher am Ende auf mehrere Dateien aufgesplittet werden mussten, half zumindest, die technischen Schwierigkeiten punktuell zu minimieren.

Alternative Möglichkeiten sind natürlich Desktop Publishing Lösungen. Jedoch stehen wir hier vor der Herausforderung, dass die Erstellung eines Dokuments sich hierin komplett anders gestaltet. Der Anwender wird hier deutlich technischer gefordert. Für Neulinge ist es unendlich schwieriger, in einer solchen DTP-Anwendung ans Ziel zu kommen. Allerdings wird man damit belohnt, dass nahezu alle Druckereien mit den Formaten umgehen können.

Haben wir diese Hürden genommen, stoßen wir auch gleich auf das nächste Problem: den Druck. Sucht man sich erst einen Verlag und setzt sich selbst unter Druck, weil man sich vertraglich auf ein Abgabetermin und/oder das Format geeinigt hat oder schreibt man erst das Buch, konzentriert sich auf dessen Inhalt

[1] zumindest auf mobilen Geräten

und schaut im Anschluss (oder “kurz vor Fertigstellung) nach einer Druckerei um, mit der man sich einigt? Letztere Option erscheint meist als der Weg, der für den Autor am”gesündesten” zu sein scheint. Wird dieser dann aber mit der Aussage konfrontiert, dass Exporte aus Word&Co für den Druck äußerst ungeeignet sind, steht selbiger wieder am Anfang.

Nun wollen wir uns aber nicht mit den Schwierigkeiten anderer Autoren beschäftigen, sondern schauen, wie wir im Jahr 2018 Bücher erstellen können, die wir vorzugsweise in elektronischer Form bereitstellen. Dass wir dabei aus Fehlern lernen wollen, ist selbstverständlich. Mein Vorteil ist, dass ich, bedingt durch meine unterstützende Tätigkeit, auf Erfahrungen zurückgreifen kann.

Bevor wir uns dem Hauptthema des Buches abschließend zuwenden, wollen wir noch einmal kurz einen Blick über den Tellerrand werfen und ganz kurz beleuchten, welche Möglichkeiten man im zweiten Jahrzehnt des 21. Jahrhunderts hat, um ein eBook zu ertellen, dass gegebenenfalls auch einmal in den Druck gehen kann.

Kapitel 1

Alternative Lösungen

1.1 Office-Anwendungen

Mit Erscheinen der Version LibreOffice 6.0 steht neben dem Export als PDF auch die Möglichkeit bereit, sein Werk als ePub zu speichern. Dieses Format ist im eBook-Sektor wahrscheinlich eines der verbreitesten, um anwendungsunabhängig Dokumente in die weite Welt des Datennetzes zu streuen. Nun haben wir in unserer Einleitung aber von den Schwierigkeiten gesprochen, die Office-Anwendungen haben, wenn die Dateigröße ein bestimmtes Maß überschreitet. Dieses Problem kann man verringern, aber leider nicht verhindern, indem man die Bilder nicht einbettet, sondern nur als Verknüpfung einbindet. Das wiederum setzt voraus, dass die Bilder immer mit dem Dokument mitgesendet werden, wenn man selbiges auf einem anderen Rechner betrachten oder gar weiterbearbeiten möchte.

Eine weitere Möglichkeit, ist die Arbeit mit einem sogenannten “Globaldokument” (engl. “Masterdocument”), bei dem man die einzelnen Abschnitte auf separate und vorallem eigenständige Office-Dateien verteilt. Im “Globaldokument” werden sie als Verknüpfung eingebunden. Der Vorteil hierbei ist, dass man die Dateigröße des Gesamtwerkes auf kleine Häppchen verteilt und sich zusätzlich die Möglichkeit schafft, jedes Einzeldokument unabhängig bearbeiten zu können. Der größte Nachteil dieser Lösung ist, dass alle Formate im Vorfeld abgestimmt werden sollten, weil sie sonst im Nachgang in allen Teildokumenten nachzuziehen sind, um mit ihnen arbeiten zu können. Dies nicht zu tun, erhöht den Aufwand im Masterdokument und verhindert früher oder später die Schaffung eines abgerundeten und einheitlichen Gesamterscheinunsbild. Auch ist das Masterdokument nur mit seinen “Kindern” (den eingebundenen Einzeldokumenten) lesbar, was das Teilen des Gesamtwerkes erschwert und beim Empfänger die Installation der Anwendung erfordert, die mit der Struktur umgehen kann (also MS-Word oder LibreWriter).

Wurden die Hürden der Office-Anwendungen ersteinmal genommen, kann man weitestgehend gute Ergebnisse erzielen. Soll das Ergebnis jedoch in den Druck gehen, ergeben sich weitere Schwierigkeiten. Aus Worddateien erzeugte PDF stoßen bei Druckereien nicht unbedingt auf Gegenliebe, da diese Erzeugnisse (technisch gesehen) ein stark verfälschtes Erscheinungsbild aufweisen. Wer es nicht glaubt, sollte einmal mit Textboxen und vielen Fußnoten arbeiten, das Ergebnis als PDF exportieren und anschließend versuchen, mehrzeilige Textpassagen zu kopieren. Man stellt sehr schnell fest, dass aus zusammenhängenden Sätze stark fragmentierte Einzelteile wurden.

1.2 Desktop Publishing

Auch wenn ich wenig Erfahrungen auf dem Gebiet habe, weiß ich zumindest, dass das Gro der Druckereien, Erzeugnisse aus dem DTP-Bereich am besten verarbeiten kann, weil diese Softwarelösungen darauf spezialisiert sind, Layouts für Druckerzeugnisse zu schaffen. Damit wird auch der Fokus dieser Anwendungen klar, der eindeutig darauf ausgelegt ist, millimetergenaue Positionierung von Elementen auf einer Seite vorzunehmen. Lange Fließtexte, wie sie bei Büchern entstehen, werden hierin nicht so einfach möglich. Fußnoten und andere Automatismen aus dem Bereich der Office-Anwendungen sind zumindest bei freien/kostenlosen DTP-Werkzeugen schwierig oder überhaupt nicht umsetzbar. Diese muss man teilweise händisch auf den Seiten platzieren.

1.3 (La)TeX

Humorvoll ausgedrückt, schwören wahre Helden auf TeX[1] bzw. dessen Aufsatz LaTeX[2], einem Format, dass wahrscheinlich nur Nerds verstehen und mögen. Auch hier halten sich meine Erfahrungen in Grenzen. Natürlich hat man, wenn man in der IT arbeitet, schonmal TeX-Schnipsel oder ganze Dokumente gesehen und kennt auch deren Vorteile, aber ein ganzes Buch damit zu schreiben, fällt dann doch schwer. Die Format- und Steueranweisungen stören den Lesefluss immens. Dennoch wollen wir diese Lösungen erwähnen, da auch hier millimetergenaue Positionierungen möglich sind. Anders als bei Office- und DTP-Lösungen werden LaTeX-Dokumente weniger visuell geschrieben. Optisch kann man das Ergebnis erst nach einem Compilerlauf begutachten. Dass das Kompilat die verschiedensten Ausgabeformate ermöglicht (PDF ist nur eines von vielen) macht diese Auszeichnungssprache zu einem idealen Werkzeug, wenn man sich erst am Ende auf ein Format festlegen möchte oder sein Ergebnis gar in allen erdenklichen Formaten bereitzustellen. Auch darf man davon ausgehen, dass man mit TeX-basierten Dokumenten in Druckereien nicht auf verschreckte, große Augen

[1] tex ist eine Markupsprache
[2] latex erweitert den Sprachumfang von TeX

stößt. Man kann über TeX durchaus noch mehr sagen, aber wir sparen uns die Energie lieber für das auf, was im Titel des Buches bereits ankündigten.

1.4 Fazit

Ein Sprichwort sagt “alle Wege führen nach Rom”. Unser Abschnitt “Alternative Lösungen” zeigt, dass es davon in der Tat sehr viele gibt. Sie unterscheiden sich halt im wesentlichen in ihrer Bedienbarkeit oder Ergebnissicherheit. Optisch nehmen sich der Erzeugnisse nicht viel. Die Spreu trennt sich spätestens dann vom Weizen, wenn es darum geht, das ursprünglich als reines eBook konzipierte Buch in den Druck zu geben. Sollte dieses Vorhaben, nicht zur Debatte stehen und die Leserschaft über nur wenige Zielformate erreichbar sein, ist mit den Office-Produkten für’s erste gut beraten. Aber auch die anderen Lösungen lassen sich mehr oder weniger gut für die Erfassung der Texte verwenden, nur wird man hier als Autor von der Technik abgelenkt. Wir werden im folgenden eine bisher nicht erwähnte Möglichkeit aufzeigen, die manchem ersteinmal untypisch erscheinen mag, aber eventuell auch die Lösung aller bisher betrachteten Probleme aufzeigt.

Teil II

Markdown

Wer ein Buch schreibt, möchte sich vorzugsweise auf den Inhalt konzentrieren. Das verwendete Werkzeug soll im wahrsten Sinne des Wortes unterstützen und in dieser Rolle vor allem folgende Aspekte erfüllen:

- Schreib- und Lesefluss darf von der Anwendung nicht unnötig gestört werden
- Seitenlayout und Formatierungen soll über das gesamte Buch hinweg einheitlich sein (zentrale Vorgaben)
- Unterstützung von Fußnoten, Endnoten, Verzeichnisse, Tabellen, Listen Abbildungen etc.
- geringe Lernkurve für die Anwendung, mit der der Text geschrieben wird
- Wörterbuch und/oder andere Fehleranzeige-/-korrekturhilfen

All das wird mehr oder weniger von den in der Einleitung genannten alternativen Lösungen umgesetzt. Einige Lösung verfolgen dabei einen "wysiwyg"-Ansatz[3], andere hingegen arbeiten mit Auszeichnungselementen und zeigen erst nach einem Übersetzungslauf das Ergebnis in seiner finalen Form. Ich wage jedoch zu behaupten, dass all diese Lösungen immer das Manko haben und den Autor in irgendeiner vom Schreiben oder (Korrektur)Lesen abzulenken, weil entweder die Positionierung der Elemente sich aufwändiger gestaltet oder der Text von Formatierungssymbolen überhäuft wird. Man muss bedenken, dass beide Aspekte direkten Einfluss auf die Arbeitsgeschwindigkeit und Konzentration nehmen.

Um dem entgegenzuwirken kann man an Schulungen teilnehmen (Fehler und Unzulänglichkeiten lassen sich damit nicht beseitigen) oder aber Alternativen suchen. Wir versuchen mit Markdown zweiten Ansatz. Wir werden im Laufe des Buches feststellen, dass wir mit Markdown auch nicht die Antwort auf "die Frage aller Fragen[4]" gefunden haben, aber dennoch eine alternative Möglichkeit gefunden haben, die den Fokus des Autors wieder auf das wesentliche lenkt: dem Verfassen von Texten.

Doch was ist Markdown?

[3] *W*hat *y*ou *s*ee, *i*s *w*hat *y*ou *g*et bedeutet, dass der Schreiber bereits beim Schreiben sieht, wie das Endergebnis aussieht.

[4] Die Antwort auf Frage aller Fragen "nach dem Leben, dem Universum und dem ganzen Rest" lautet 42.

Kapitel 2

Einführung in Markdown

Markdown ist eine Auszeichnungssprache, die ihren Ursprung in den "plain text emails" hat. Damit sind solche Erzeugnisse des weltweiten Datennetzes gemeint, die ausschließlich mit Buchstaben, Zahlen und Symbolen erstellt werden. Bei diesen E-Mails steht der Text im Vordergrund. Unter Zurhilfenahme von Symbolen (bspw. einem Stern $*$) und anderen Mitteln, werden dem Text Betonungen, Hervorhebungen und anderen Ausdrucksformen beigebracht. Gleiches gilt für die Strukturierung, die bspw. Abschnitte, Überschriften, Listen oder Tabellen sein können. Dabei verfolgt Markdown einen äußerst minimalistischen Ansatz, der die zur Textgestaltung erforderlichen Zusatzinformationen so stark reduziert, dass Lesefluss nahezu ungestört bleibt und die Bedeutung der Elemente selbst für Laien klar aus dem Kontext hervorgeht.

Doch warum sollte man auf eine solche Auszeichnungssprache ausweichen, wo es für das Formatieren bereits verschiedene Lösungen gibt? Die wahrscheinlich treffendste Antwort ist die Unabhängigkeit vom Werkzeug. Markdown-Dokumente lassen sich in ihrer Rohform (also im Quelltext) mit jedem einfachen Texteditor öffnen. Das trifft zwar grundsätzlich auch für alle anderen Formatierungsmöglichkeiten von Textdateien zu (auch HTML-Dokumente), jedoch haben diese die Eigenschaft, den Text mit deutlich mehr Zusatzinformationen zu überfrachten, was sich zumindest auf den Lesefluss negativ auswirkt.

Neben Markdown gibt es mit AsciiDoc, reStructuredText oder Textile zwar viele Alternativen, die ein ähnliches Ziel verfolgen, jedoch ist deren Bekanntheitsgrad teilweise eingeschränkt. Auch bieten manche mehr Möglichkeiten an, woraus eine umfangreicheres Vokabular resultiert und die Syntax komplexer macht. Das schlägt sich natürlich auch auf den Lesefluss nieder. Wahrscheinlich ist das auch der Grund dafür, dass mit GitHub und StackOverflow große Anbieter auf Markdown setzten, was sich auch auf die Verbreitung wiederspiegelt. Selbst weniger technische Plattformen, wie die Deutsche Presseagentur (dpa) setzen

auf Markdown[1].

Um das Bild weiter abzurunden, wollen wir noch einmal kurz auf die Entstehungsgeschichte eingehen. Die Markdown-Begründer John Gruber und Aaron Swartz veröffentlichten 2004 die Version 1.0.0 und im gleichen Jahr die bis heute unveränderte 1.0.1. Die beiden Herren sammelten die Ausdrucksformen aus "plain text emails" und brachten sie in ein einheitliches Format. Dass dabei HTML als Gegenstück zu reinen Text-E-Mails eine wesentliche Rolle spielte, zeigt auch das Perlskript, dass auf der Homepage der Sprache daringfireball.net[2] zu finden ist. Mit dessen Hilfe lassen sich Markdownstrukturen in HTML übersetzen.

Dass seit 2004 keine Änderungen vorgenommen wurden, liegt wahrscheinlich daran, dass die Sprache an sich vollständig erscheint. Einzig die inhaltlichen und technischen Erweiterungen, die sich rund um die Auszeichnungssprache bilden, sind ein Zeugnis dafür, dass aus technischer Sicht bestimmte Dinge einfacher dargestellt werden können (automatische Erkennung von Links) oder Dinge nachgereicht wurden, die im Originalwortschatz nicht vorkamen, den Einsatzzweck allerdings deutlich erweitern (Tabellen, Fußnoten).

Doch nun genug der vielen Worte. Lasst die Spiele beginnen!

[1] https://web.archive.org/web/20150402181810/http://www.dpa-newslab.com/2010/04/22/dpa-notizbuch-markdown/

[2] https://daringfireball.net/projects/markdown/

Kapitel 3

Abschnitte und Zeilenumbrüche

Bevor wir auf Textelemente eingehen, wollen wir uns mit der Strukturierung der Dokumente befassen. Gemeint ist die Unterteilung in Abschnitte und einfache Zeilenumbrüche, die innerhalb eines Abschnitts vorkommen, ohne einen neuen einzuleiten.

Markdown orientiert sich an HTML und in dieser Eigenschaft können lange Sätze umgebrochen werden ohne das mit ihnen ein Zeilenumbruch verbunden wird. In HTML wird für einen Zeilenumbruch mit dem Tag[1] `<br>` erzwungen. Zeilenumbrüche, die im HTML-Quelltext gemacht werden, um bei langen Textpassagen nicht ständig von links nach rechts zu müssen, wird der Betrachter der Internetseite nie sehen, da diese nicht gerendert[2] werden. Da Markdown vor allem für gute Lesbarkeit steht, sind auch hierin Umbrüche von langen Texten möglich. Einen Zeilenumbruch erzwingt man hier jedoch etwas unscheinbarer, als es in HTML der Fall ist. Man hängt der Zeile zwei Leerzeichen an.

Unser Beispiel erweitern wir nun um einen Text “nächste Zeile”, der einem Zeilenumbruch folgt, ohne ein neues Blockelement (in diesem Fall ein Abschnitt) einzuleiten. Ferner werden im nun folgenden Beispiel alle Leerzeichen in Form eines Punkt “” “sichtbar” gemacht und der Zeilenumbruch als “¶” angezeigt.

```
<p>Abschnitt 1<br>nächste Zeile</p>
<p>Abschnitt 2<br>nächste Zeile</p>
```

Abschnitt 1
nächste Zeile

[1] HTML-Tags sind die Elemente der Markupsprache HTML, mit denen Inhalte strukturiert werden.

[2] Damit ist die Visualisierung gemeint, also die Darstellung, die sich aus den Anweisungen, die mit HTML oder anderen Sprachen definiert wurden.

Abschnitt 2
nächste Zeile

Neben einfachen Zeilenumbrüchen, die in der Regel nur zwei Sätze voneinander trennen, nicht aber ein neues Teilthema beginnen, gibt es Abschnitte/Absätze. Sie unterteilen den Text in einzelne Teilthemen-Blöcke. Meist leitet ein neuer Absatz eine Passage ein, die sich etwas stärker vom vorangegangenem abgrenzt.

In HTML werden solche Absätze mit `<p>`-Tag gekennzeichnet. In Markdown haben wir keine Tags, schon gar nicht für die Einleitung neuer Abschnitte oder Zeilenumbrüche. Hier wird mit ENTER und Leerzeichen (spaces) gearbeitet.

Einen neuen Abschnitt leitet man ein, indem man zwischen dem aktuellen und dem neuen eine leere Zeile platziert. Das gilt im übrigen für alle Blockelemente.

```
Abschnitt 1

Abschnitt 2
```

```
<p>Abschnitt 1</p>
<p>Abschnitt 2</p>
```

Abschnitt 1

Abschnitt 2

Kapitel 4

Span-Elemente

Nachdem wir nun wissen, wie man dass man überlange Zeilen einfach umbrechen kann, neue Abschnitte mit einer Leerzeile einleitet und einfache Zeilenumbrüche durch anhängen zweier Leerstellen erreicht, wollen wir uns die einfacheren Formatanweisungen anschauen. Damit sind Spanelemente (auch Inline-Elemente genannt) gemeint, die Teile eines Fließtextes sind. Einzelne Textfragmente werden aber erst dann zu Spanelementen, wenn sie individuell formatiert oder hervorgehoben werden. Ansonsten bleiben sie einfache Bestandteile eines Textes.

4.1 Betonung

Wer, wie ich, im Deutschunterricht nicht der beste Zuhörer war und lieber die Ränder seiner Schulbücher derart auszuschmückte, dass aus ihnen Daumenkinos wurden, der wird sich "freuen", wenn wir noch einmal ganz kurz auf Definitionen eingehen, die dem Text etwas würzen. Zum Beispiel mengt die Betonung der Grundmelodie von Sätzen etwas besonderes bei. Ich spreche von Situationen, in denen einzelne Worte oder kurze Fragmente hervorgehoben werden. Das geschieht in der Regel in Form von Unterstreichungen, kursiver Schriftstellung oder fetter Formatierung.

Da sich Markdown primär an der Webwelt orientiert, denken wir wahrscheinlich gleich an die HTML-Tags `<u>`, `<i>` und `<b>`. Dass dem nicht so ist, liegt daran, dass diesen Tags keine betonende Semantik zugesprochen wird. Wir werden stattdessen die beiden Tags `<em>` und `<strong>` finden. Unterstrichene Passagen finden wir in der Auszeichnungssprache nicht, da es bei Markdown nicht um die optische Gestaltung geht, sondern, wie bereits erwähnt um die Semantik, also Betonung und Hervorhebungen im Sinne einer anderen Aussprache.

```
*single asterisks*
_single underscores_
```

```
**double asterisks**
__double underscores__
```

```
<em>single asterisks</em>
<em>single underscores</em>
<strong>double asterisks</strong>
<strong>double underscores</strong>
```

single asterisks
single underscores
double asterisks
double underscores

Dass man Span-Elemente auch Inline-Elemente nennt, haben wir bereits erwähnt. Diese alternative Bezeichnung verrät wahrscheinlich schon eher, dass solche Elemente einzelne Worte und Textpassagen sind und seltener als großer zusammenhängender Textblock in Erscheinung treten. Das folgende, ausführlichere Beispiel soll dies veranschaulichen.

```
Lorem ipsum dolor *sit amet*, consetetur _sadipscing elitr_,
sed diam nonumy eirmod tempor invidunt ut _**labore et** dolore_
magna aliquyam erat, sed diam voluptua.
```

Lorem ipsum dolor *sit amet*, consetetur *sadipscing elitr*, sed diam nonumy eirmod tempor invidunt ut ***labore et*** *dolore* magna aliquyam erat, sed diam voluptua.

4.2 Links

Das welteweite Datennetz wurde erst durch die Möglichkeit der Verknüpfung zu einem Netz. Diese sind selbstverständlich auch in Markdown möglich. Man unterscheidet dabei zwischen internen und externen Links. Erstgenannte zeigen auf bestimmte Stellen innerhalb eines Dokuments und werden Sprungmarken genannt.
Externe Verweise zeigen auf andere (externe) Dokumente. Bezogen auf das Internet verweisen diese auf Dokumente, die unter der gleichen Domain gefunden werden können oder sich gar auf Servern Dritter befinden.

```
[localhost](http://localhost)
[localhost](http://localhost "Tooltip-Text")
[unser Impressum](/impressum)
[Abschnitt "Links"](#links)
```

```
<a href="http://localhost">localhost</a><br>
<a href="http://localhost" title="Tooltip-Text">localhost</a><br>
<a href="/impressum">unser Impressum</a><br>
<a href="#links">Abschnitt "Links"</a>
```

localhost
localhost
unser Impressum
Abschnitt “Links”

Wer den Lesefluss, aber auch das Erscheinungsbild der Markdown-Dokumente nicht unnötig stören möchte, kann die Adresse eines Links auch außerhalb des Textes platzieren. Dafür muss der Link mit einer eindeutigen, frei wählbaren ID versehen werden, um eine Verbindung zwischen dem Platzhalter im Text und der Zuweisung der zugehörigen Adresse zu ermöglichen.

```
Es gibt neben [DuckDuckGo][duck] und [Tor][] sehr viele Internetseiten, die ihre
Privatsphäre schützen oder gar ihre Identität gegenüber Dritte verbergen.
Ein sehr nettes Handbuch findet man [hier][3].

[duck]: https://duckduckgo.com
[Tor]: https://duckduckgo.com
[3]: https://www.privacy-handbuch.de
```

```
Es gibt neben <a href="https://duckduckgo.com">DuckDuckGo</a> und <a href="https://duckduckgo.com
Privatsphäre schützen oder gar ihre Identität gegenüber Dritte verbergen.
Ein sehr nettes Handbuch findet man <a href="https://www.privacy-handbuch.de">hier</a>.
```

Es gibt neben DuckDuckGo und Tor sehr viele Internetseiten, die ihre Privatsphäre schützen oder gar ihre Identität gegenüber Dritte verbergen. Ein sehr nettes Handbuch findet man hier.

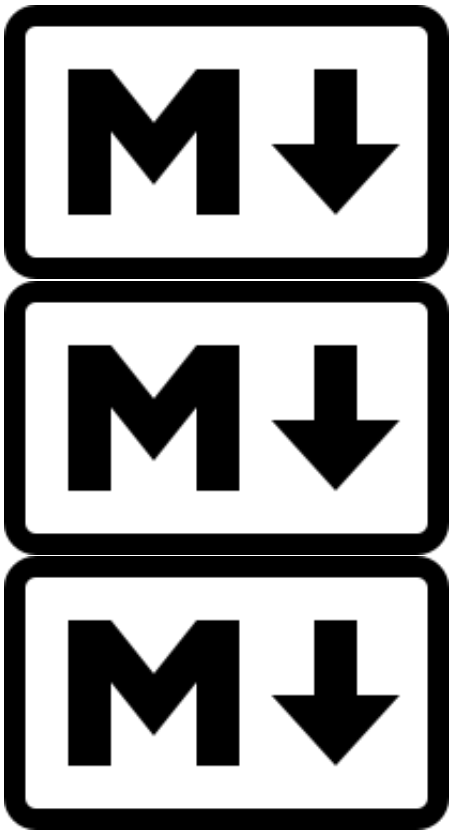
4.3 Bilder

Eine andere Form der Verlinkung stellt die Einbettung von Grafik und Bildern dar. Warum? Auch hier wird auf eine Adresse verwiesen, die nicht unmittelbarer Bestandteil des Dokumentes ist. In HTML werden sie durch den `<img>`-Tag eingebunden. In Markdown werden Grafiken ähnlich den Links in dem Text eingebettet. Analog zu den Links, können Platzhalter im Text und die Definition der Adresse außerhalb des Fließtextes für eine bessere Lesbarkeit sorgen.

```
![Markdown-Verweis]
![Markdown-Alt](resources/Markdown-mark.png)
![Markdown-Alt+Title](resources/Markdown-mark.png "Quelle: https://upload.wikimedia.org/wikipedia

[Markdown-Verweis]: resources/Markdown-mark.png
```

```
<img src="resources/Markdown-mark.png" alt="Markdown-Verweis">
<img src="resources/Markdown-mark.png" alt="Markdown-Alt">
<img src="resources/Markdown-mark.png" title="Quelle: https://upload.wikimedia.org/wikipedia/comm
```



4.4 Quelltext

Es gibt Situationen, in denen technische Feinheiten hervorgehoben werden sollen. Dazu gehören neben mehrzeiligen Quelltextfragmenten auch einzelne Elemente, wie Namen von Variablen, Funktionen oder anderen Konstrukten aus Programmier- oder Skriptsprachen. Man sieht nicht selten, dass diese in Dokumenten in Anführungsstriche verpackt werden. Semantisch korrekt ist jedoch, solche Elemente als das auszuweisen, was sich wirklich sind: Quellcode. Als Teile von Fließtexten werden diese Elemente durch einfache “Backsticks” (also “‘) ausgezeichnet.

```
Während PHP `foreach` ein Bestandteil der Skriptsprache selbst ist,
ist es in JavaScript eine Funktion von Arrays und nennt sich hier `each`.
```

```
Während PHP <code>foreach</code> ein Bestandteil der Skriptsprache selbst ist,
ist es in JavaScript eine Funktion von Arrays und nennt sich hier <code>each</code>.
```

Während PHP `foreach` ein Bestandteil der Skriptsprache selbst ist, ist es in

JavaScript eine Funktion von Arrays und nennt sich hier `each`.

4.5 Spanelement-Erweiterungen

Dass sich mit Markdown nicht der gesamte Funktionsumfang von HTML umsetzen lässt, versteht sich von selbst. Schließlich ist ein primärer Charakter dieser Auszeichnungssprache ihre Einfachheit. Dennoch fehlt dem Original das eine oder andere Feature, das durchaus sinnvoll ist. Auf diese Spracherweiterungen wollen wir im folgenden eingehen.

4.5.1 Durchstreichen, Hoch- und Tiefstellen (Erweiterungen)

```
In Markdown mit Hilfe von Erweiterungen etwas ~~durchstreichen~~, hoch^stellen^ oder tief~stellen
```

In Markdown mit Hilfe von Erweiterungen etwas ~~durchstreichen~~, hoch$^{\text{stellen}}$ oder tief$_{\text{stellen}}$ kann jeder, aber unterstreichen, scheinbar nicht.

4.5.2 Abkürzungen (Erweiterung)

Eine weitere Spracherweiterung sind Abkürzungen. Diese werden im Fließtext platziert. Ihre Langbezeichnung wird in der Regel als sogenannter Tooltip eingeblendet. Solche Tooltips sind für Geräte mit Maus, eine feine Sache. Doch was ist mit Geräten, die ausschließlich über Touchscreens verfügen. Hier gibt es den hierfür erforderlichen MouseOver-Effekt (noch?) nicht.

Das ist ein Grund, warum PanDoc[1] diese Notation nicht in seiner Standardeinstellung interpretiert. Alternativ kann man auf Fußnoten setzen.

```
EBM ist eine Musikrichtung, die ihren Ursprung in den 80er Jahren hat.

*[EBM]: Electronic Body Music
```

```
<abbr title="Electronic Body Music">EBM</abbr> ist eine Musikrichtung, die ihren Ursprung in den
```

EBM ist eine Musikrichtung, die ihren Ursprung in den 80er Jahren hat.

[1] PanDoc: wir gehen später darauf ein

Kapitel 5

Block-Elemente

Ein Blockelement repräsentiert eine komplexere Form im Dokument. Anders als Spanelemente, grenzen sie sich klar von ihren Nachbarn ab und bilden (wie der Namen schon sagt) einen ins sich geschlossenen Block. Die einfachste Form ist ein Textabschnitt, bestehend aus einzelnen Worten, Sätzen oder gar Grafiken und Links. Neben diesem, gibt es noch Überschriften, Listen, Tabellen und andere, die wir im folgenden besprechen wollen.

5.1 Horizontale Linien

Die einfachsten Blockelemente sind horizontale Linien, die man gern als starke Abgrenzung zwischen zwei Abschnitten nimmt oder in Dokumenten gar als Trennlinie zwischen dem Inhalt und den Fußnoten auf einer Seite einbringt.

Die Trennlinie kann mit drei aufeinanderfolgende * (Stern) oder - (Minus) in den Text eingebracht werden. Zwischen die Zeichen sollte sich maximal ein Leerzeichen befinden.

::::: flex ::: Quelltext

***

* * * * * * * * * *

- - -

- - -

::: ::: HTML

```
<hr />

<hr />

<hr />

<hr />
```

::: ::: Vorschau

::: :::::

5.2 Überschriften

Nahezu jedes aus mehreren Kapiteln bestehende Dokument verwendet Überschriften, die die ihr folgenden Abschnitte einleitet. Überschriften werden durch eine oder mehreren vorangestellten Rautesymbolen (‘#’) kenntlich gemacht, wobei die Anzahl der Rauten gleichbedeutend ihrer Ebene ist. Der HTML-Standard ist auf sechs Ebenen limitiert. Dank der Analogie zur jener Spezifikation, gilt die Limitierung auch für Markdown.

```
# Überschrift Ebene 1

## Überschrift Ebene 2

### Überschrift Ebene 3

#### Überschrift Ebene 4

##### Überschrift Ebene 5

###### Überschrift Ebene 6
```

```
<h1>Überschrift Ebene 1</h1>

<h2>Überschrift Ebene 2</h2>

<h3>Überschrift Ebene 3</h3>

<h4>Überschrift Ebene 4</h4>

<h5>Überschrift Ebene 5</h5>

<h6>Überschrift Ebene 6</h6>
```

Überschrift Ebene 1

Überschrift Ebene 2

Überschrift Ebene 3

Überschrift Ebene 4

Überschrift Ebene 5

Überschrift Ebene 6

Wer nur zwei Ebenen zulässt, kann statt der vorangestellten Raute-Symbole die Überschrift auch unterstreichen.

```
Überschrift Ebene 1
===================

Überschrift Ebene 2
-------------------
```

```
<h1>Überschrift Ebene 1</h1>

<h2>Überschrift Ebene 2</h2>
```

Überschrift Ebene 1

Überschrift Ebene 2

5.2.1 Überschriften als interne Adresse (Erweiterung)

Überschriften sind ein gutes Mittel, um sein Dokument zu strukturieren. In einem gedruckten Buch findet man Hinweise im Text oder das Inhaltsverzeichnis, um auf andere Kapitel zu verweisen. Markdown stellt über eine Erweiterung eine ähnliche Möglichkeit bereit, mit der (aus technischer Sicht) sogar etwas mehr Komfort verbunden wird. Da ein Verweis letzten Endes ein Link ist, ist es nicht verwunderlich, dass Sprungmarken im Text ähnlich formuliert werden.

Das Beispiel zeigt alle Möglichkeiten, die als Verweis auf `Meine Überschrift` dienen.

```
# Meine Überschrift

Hier könnte nun sehr viel Text folgen, aber...

[Meine Überschrift]
[Meine Überschrift][]
[zurück zu 'Meine Überschrift'][Meine Überschrift]
[zurück zu 'Meine Überschrift'](#meine-überschrift)
```

```
<h1 id="meine-überschrift">Meine Überschrift</h1>
<p>Hier könnte nun sehr viel Text folgen, aber…</p>
<p><a href="#meine-überschrift">Meine Überschrift</a><br>
```

```
<a href="#meine-überschrift">Meine Überschrift</a><br>
<a href="#meine-überschrift">zurück zu ‘Meine Überschrift’</a><br>
<a href="#meine-überschrift">zurück zu ‘Meine Überschrift’</a></p>
```

Meine Überschrift

Hier könnte nun sehr viel Text folgen, aber…

Meine Überschrift Meine Überschrift zurück zu ‘Meine Überschrift’ zurück zu ‘Meine Überschrift’

5.2.2 Überschriften-IDs (Erweiterung)

Wer ein Buch schreibt und hierin auf Kapitel verweisen möchte, möchte dies eventuell nicht in der Form machen, als dass er den Text der Überschrift zitiert, sondern im Text eigene Markerierung setzt, auf die er verweisen kann. Ein Möglichkeit hierfür ist die Definition einer eigenen ID, indem hinter dem Text der Überschrift mithilfe des Ausdrucks `{#<id-der-Überschrift>}` eine solche vergeben wird. Dass im Beispiel der Ausdruck `<id-der-Überschrift>` durch die eigene IDzu ersetzen ist, versteht sich hoffentlich von selbst.

```
# Überschrift Ebene 1 {#überschrift-1}

## Überschrift Ebene 2 {#überschrift-1-2}
```

```
<h1 id="überschrift-1">Überschrift Ebene 1</h1>

<h2 id="überschrift-1-2">Überschrift Ebene 2</h2>
```

Überschrift Ebene 1

Überschrift Ebene 2

5.3 Zitate

Wer inhaltlich korrekt unterwegs sein möchte, wird Zitate als solche auszeichnen wollen. Das geht mit `>`, das jeder Zeile eines Zitats vorangestellt wird. Spätestens jetzt sollte dem Leser auffallen, dass Markdown seinen Ursprung in den Klartextmails hat.

Längere Zitate lassen sich über mehrere Zeilen verteilen. Dabei ist auf die Einrückung zu achten, wenn man, wie im nachfolgenden Beispiel zu sehen, auf das vorangestellte Auszeichnungssymbol verzichtet. Außerdem ist eine beliebige Verschachtelung möglich. Dabei definiert die Anzahl der vorangestellten Größerzeichen `>` die Verschachtelungstiefe.

```
Meine beiden Lieblingszitate, die das heutige Leben (also die ersten beiden Jahrzehnte
```

```
> Ich habe mir immer gewünscht, dass mein Computer so einfach zu bedienen sein sollte, wie mein
  Mein Wunsch wurde wahr.
  Ich weiß nicht mehr, wie mein Telefon funktioniert.
> (Bjarne Stroustrup, C++ Entwickler)

> > Zwei Dinge sind unendlich: Das Universum und die menschliche Dummheit. Aber beim Universum
> > (Albert Einstein, Physiker)
```

Meine beiden Lieblingszitate, die das heutige Leben (also die ersten beiden Jahrzehnte des 21. Jahrhundert) am besten beschreiben, sind folgende:

> Ich habe mir immer gewünscht, dass mein Computer so einfach zu bedienen sein sollte, wie mein Telefon. Mein Wunsch wurde wahr. Ich weiß nicht mehr, wie mein Telefon funktioniert. (Bjarne Stroustrup, C++ Entwickler)
>
> > Zwei Dinge sind unendlich: Das Universum und die menschliche Dummheit. Aber beim Universum bin ich mir nicht ganz sicher. (Albert Einstein, Physiker)

5.4 Aufzählungen, Listen

Immer nur alles in einem Fließtext zu verpacken, wird auf Dauer langweilig, insbesondere dann, wenn man Inhalte anders strukturiert, besser vermitteln kann. Hierzu zählen geordnete und lose (unsortierte) Listen.

5.4.1 Unsortierte Listen

Jedem Eintrag einer unsortierten Auflistung wird ein Bindestrich (Minuszeichen) `-`, Plussymbol `+` oder ein Stern `*` vorangestellt. Um Listen zu verschachteln, sprich mit Untereinträgen zu versehen, werden die Kindelemente mit mindestens 2 Leerzeichen eingerückt.

Mehrzeilige Listeneinträge sind zulässig. Zeilenumbrüche und Abschnitte werden auch innerhalb von Listen unterstützt. In dem nun folgenden Beispiel, finden wir unter im Listeneintrag “Punkt 2.2” sowohl einen Zeilenumbruch, als auch einen Absatz innerhalb der Aufzählung. Auch wurden hier die Symbole gemischt, worauf man im realen Markdown-Leben zur Verbesserung der Lesbarkeit verzichten sollte.

```
- Punkt 1
+ Punkt 2
  - Punkt 2.1
  - Punkt 2.2
```

```
    nächste Zeile

    und eine weitere Zeile
  - Punkt 2.3
nächste Zeile
* Punkt 3
+ Punkt 4
```

```
<ul>
    <li>Punkt 1</li>
    <li>Punkt 2
        <ul>
            <li>Punkt 2.1</li>
            <li>
                <p>Punkt 2.2<br>
                   nächste Zeile
                </p>
                und eine weitere Zeile
            </li>
            <li>Punkt 2.3 nächste Zeile</li>
        </ul>
    </li>
    <li>Punkt 3</li>
    <li>Punkt 4</li>
</ul>
```

- Punkt 1
- Punkt 2
  - Punkt 2.1
  - Punkt 2.2
    nächste Zeile
    und eine weitere Zeile
  - Punkt 2.3 nächste Zeile
- Punkt 3
- Punkt 4

5.4.2 Geordnete Listen

Geordnete Listen, auch sortierte Listen genannt, definieren eine Reihenfolge, indem sie jedem Eintrag eine fortlaufende Bezeichnung (Zahl, Buchstabe, römische Ziffer) voranstellen. Das Beispiel zeigt auch, dass man die Nummerierung nicht vorgeben muss, weil diese automatisch erfolgt. Dieser Automatismus verhindert aber zugleich eine individuelle Ordnung, was zu Schwierigkeiten bei Zahlenstrahlen führen kann.

```
1. Punkt 1
1. Punkt 2
    a. Punkt 2.1
    a. Punkt 2.2
        i. Punkt 2.2.1
        i. Punkt 2.2.2
1. Punkt 3
1. Punkt 4
```

```
<ol type="1">
    <li>Punkt 1</li>
    <li>Punkt 2
        <ol type="a">
            <li>Punkt 2.1</li>
            <li>Punkt 2.2
                <ol type="i">
                    <li>Punkt 2.2.1</li>
                    <li>Punkt 2.2.2</li>
                </ol>
            </li>
        </ol>
    </li>
    <li>Punkt 3</li>
    <li>Punkt 4</li>
</ol>
```

1. Punkt 1
2. Punkt 2
    a. Punkt 2.1
    b. Punkt 2.2
        i. Punkt 2.2.1
        ii. Punkt 2.2.2
3. Punkt 3
4. Punkt 4

Darüber hinaus, lässt sich die Auflistung weiter verfeinern und die Auflistung mit einem bestimmten Wert beginnen. Das kann hilfreich sein, wenn aus einem bestimmten Grund eine Liste an beliebiger Stelle fortgeführt werden soll. Beginnt eine Zeile mit einer Zahl, gefolgt von einem Punkt und einem Leerzeichen, wird dies als Beginn einer neuen Liste verstanden. Wer dieses Verhalten unterbinden möchte, muss den Punkt "escapen".

```
1618\. Der Dreißjährige Krieg beginnt
Der Krieg erfasst erst im Jahr 1626\. das reiche Städtchen Jüterbog. Bis dahin
blieb die Stadt vom Krieg verschont und war mit seinen 4000 Einwohnern eine für
damalige Zeiten relativ große Stadt.
Aufgrund seiner geografischen Lage war es unter anderem Veranstaltungsort
von Fürstentreffen. Jüterbog gehörte seinerzeit noch zu Sachsen, was sich
```

```
erst nach dem Krieg änderte.

1648\. Der 30-Jährige Krieg ist zu Ende
Aus der einst blühenden Stadt Jüterbog ist ein Dorf geworden. Ganze 300 Seelen
zählt die Stadt nach dem Krieg. Die meisten von ihnen sind gestorben oder in
weniger umkämpfte Gebiete umgesiedelt. Auch haben sich viele von ihnen
armutsbedingt den Armeen angeschlossen.
```

```
<p>1618. Der Dreißjährige Krieg beginnt<br>
Der Krieg erfasst erst im Jahr 1626. das reiche Städtchen Jüterbog. Bis dahin
blieb die Stadt vom Krieg verschont und war mit seinen 4000 Einwohnern eine für
damalige Zeiten relativ große Stadt.<br>
Aufgrund seiner geografischen Lage war es unter anderem Veranstaltungsort von
Fürstentreffen. Jüterbog gehörte seinerzeit noch zu Sachsen, was sich erst nach
dem Krieg änderte.</p>
<p>1648. Der 30-Jährige Krieg ist zu Ende<br>
Aus der einst blühenden Stadt Jüterbog ist ein Dorf geworden. Ganze 300 Seelen
zählt die Stadt nach dem Krieg. Die meisten von ihnen sind gestorben oder in
weniger umkämpfte Gebiete umgesiedelt. Auch haben sich viele von ihnen
armutsbedingt den Armeen angeschlossen.</p>
```

1618. Der Dreißjährige Krieg beginnt
Der Krieg erfasst erst im Jahr 1626. das reiche Städtchen Jüterbog. Bis dahin blieb die Stadt vom Krieg verschont und war mit seinen 4000 Einwohnern eine für damalige Zeiten relativ große Stadt.
Aufgrund seiner geografischen Lage war es unter anderem Veranstaltungsort von Fürstentreffen. Jüterbog gehörte seinerzeit noch zu Sachsen, was sich erst nach dem Krieg änderte.
1648. Der 30-Jährige Krieg ist zu Ende
Aus der einst blühenden Stadt Jüterbog ist ein Dorf geworden. Ganze 300 Seelen zählt die Stadt nach dem Krieg. Die meisten von ihnen sind gestorben oder in weniger umkämpfte Gebiete umgesiedelt. Auch haben sich viele von ihnen armutsbedingt den Armeen angeschlossen.

5.4.3 Definitions-/Datenlisten (Erweiterung)

Eine Spracherweiterung, die nicht zum Original-Markdown gehört, sind Datenlisten. Sie ermöglicht, einem Schlüsselwort oder einer Wortgruppe eine oder mehrere Beschreibungen zuzuweisen. In HTML spricht man hierbei von einer “datalist” (HTML-Tag `<dl>`), die aus einem Titel besteht (“datatitle”, HTML-Tag `<dt>`), dem Beschreibungselemente (“datadescription”, HTML-Tag `dd`) folgen. Jeder `<dt>`-Tag leitet semantisch ein neues Element ein, dem die folgenden `<dd>` als Beschreibungen zugesprochen werden.

```
Linux
: Opensource-Betriebssystem
```

```
: von Linus Torvald

Windows
: Betriebssystem aus dem Hause Microsoft
: von Bill Gates

iOS
: Betriebssystem von Apple
: von Steve Wozniak, Ron Wayne und Steve Jobs
```

```
<dl>
    <dt>Linux</dt>
    <dd>Opensource-Betriebssystem</dd>
    <dd>von Linus Torvald</dd>
    <dt>Windows</dt>
    <dd>Betriebssystem aus dem Hause Microsoft</dd>
    <dd>von Paul Allen und Bill Gates</dd>
    <dt>iOS</dt>
    <dd>Betriebssystem von Apple</dd>
    <dd>von Steve Wozniak, Ron Wayne und Steve Jobs</dd>
</dl>
```

Linux Opensource-Betriebssystem
: von Linus Torvald

Windows Betriebssystem aus dem Hause Microsoft
: von Bill Gates

iOS Betriebssystem von Apple
: von Steve Wozniak, Ron Wayne und Steve Jobs

5.5 Tabellen (Erweiterung)

Auch Tabellen gehören zu den Elementen, die im Markdown-Original nicht vorkommen. Auch sie entstammen einer der vielen Erweiterungen. Ihre Auszeichnung gestaltet sich äußerst einfach. Selbst die Eigenschaften, wie die rechts- oder linksbündige Ausrichtung einer Spalte, lassen sich äußerst einfach definieren.

Vernünftig formatierte Tabellen bestehen aus einer oder mehreren Kopfzeilen, gefolgt von den Inhalten. Diesem Muster folgen Tabellen in Markdown. Allerdings wurden “tablefooter” (Tabellenfußzeilen) in der freien Markdonw-Wildbahn noch nicht gesehen, was durchaus als Anreiz verstanden werden darf, sich an einer der existierenden Erweiterung zu beteiligen[1], um dies nachzuholen.

[1] Auch wenn Vielfalt belebt, sollte man versuchen, etablierte Dinge zu verbessern, als sie unübersichtlich werden zu lassen, weil es zigtausende Alternativen um ihre jeweilige Daseinsberechtigung streiten.

Das nachfolgende Beispiel zeigt, wie eine Tabelle in Markdown definiert wird und für einzelne Spalten Eigenschaften vorgegeben werden. Die Spalte mit der Überschrift “Standard” wird ohne Formatvorgaben “normal” dargestellt, was je nach Sprache links (lateinische Sprachen) oder rechtsbündig (beispielsweise Chinesisch) sein kann.

Den übrigen Spalten (der Beispieltabelle) wurde das Format der Spalteninhalte vorgegeben. Diese erfolgt in der Zeile, die den Tabellenkopf vom Inhalt trennt. Die Position des Doppelpunktes gibt Auskunft darüber, ob die Fließrichtung des Inhalts linksbündig (Doppelpunkt ist links platziert), rechtsbündig (Doppelpunkt befindet sich rechts), zentriert (je ein Doppelpunkt auf beiden Seiten) oder überhaupt nicht vorgegeben wird.

```
| Rechts  | Links   |  Zentriert   | Standard
| -------:|:------- |:------------:| ---------
| 12      | 12      | 12           | 12
| 123     | 123     | 123          | 123
| 1       | 1       | 1            | 1
```

```
<table>
    <thead>
        <tr class="header">
            <th style="text-align: right;">Links</th>
            <th style="text-align: left;">Rechts</th>
            <th style="text-align: center;">Zentriert</th>
            <th>Standard</th>
        </tr>
    </thead>
    <tbody>
        <tr class="odd">
            <td style="text-align: right;">12</td>
            <td style="text-align: left;">12</td>
            <td style="text-align: center;">12</td>
            <td>12</td>
        </tr><tr class="even">
            <td style="text-align: right;">123</td>
            <td style="text-align: left;">123</td>
            <td style="text-align: center;">123</td>
            <td>123</td>
        </tr><tr class="odd">
            <td style="text-align: right;">1</td>
            <td style="text-align: left;">1</td>
            <td style="text-align: center;">1</td>
            <td>1</td>
        </tr>
    </tbody>
</table>
```

Rechts	Links	Zentriert	Standard
12	12	12	12
123	123	123	123
1	1	1	1

Im übrigen ist es egal, ob man die Spaltenbreite über alle Zeilen gleich hält. Das verbessert die Lesbarkeit des Markdown-Dokuments. Zulässig ist aber auch die folgende Schreibweise, bei der wir neben der Spaltenbreite auch auf das vorangestellte "Pipe"-Symbol (|) verzichten.

```
Rechts | Links | Zentriert | Standard
-:|:-|:-:|-
12 | 12 | 12 | 12
123 | 123 | 123 | 123
1 | 1 | 1 | 1
```

Rechts	Links	Zentriert	Standard
12	12	12	12
123	123	123	123
1	1	1	1

5.6 Fußnoten (Erweiterung)

Sobald Bücher technischer werden oder Zusatzinformationen unablässig erscheinen, kann der Einsatz von Fußnoten hilfreich werden. Mit ihrer Hilfe können Dinge erklärt werden oder Hinweise auf weiterführende Informationen gegebenen werden, ohne dass der FLuß des Textes beeinträchtigt wird.

Fußnoten lassen sich in zweierlei Form in den Text einbetten und orientieren sich in ihrer Notation an den Links.

```
Wer die Antwort auf die Frage "nach dem Leben, dem Universum und dem ganzen Rest" finden möchte,
sollte bei wikipedia[^wiki42] vorbeischauen oder,
und das ist wirklich ein wärmstens zu empfehlendes Buch,
den Roman "Per Anhalter durch die Galaxis"^[Per Anhalter durch die Galaxis wurde von Douglas Adam
lesen, aus dem dieses Zitat stammt.

[^wiki42]: (https://de.wikipedia.org/wiki/42_(Antwort))[https://de.wikipedia.org/wiki/42_(Antwort

<p>Wer die Antwort auf die Frage "nach dem Leben, dem Universum und dem ganzen Rest" finden möcht
sollte bei wikipedia<a href="#example-fn1" class="footnote-ref" id="example-fnref1"><sup>1</sup><
und das ist wirklich ein wärmstens zu empfehlendes Buch,
den Roman "Per Anhalter durch die Galaxis"<a href="#example-fn2" class="footnote-ref" id="example
```

```
lesen, aus dem dieses Zitat stammt.</p>

<ol>
    <li id="example-fn1"><p><a href="https://de.wikipedia.org/wiki/42_(Antwort)">https
    <li id="example-fn2"><p>Per Anhalter durch die Galaxis wurde von Douglas Adams ges
</ol>
```

Wer die Antwort auf die Frage “nach dem Leben, dem Universum und dem ganzen Rest” finden möchte, sollte bei wikipedia1 vorbeischauen oder, und das ist wirklich ein wärmstens zu empfehlendes Buch, den Roman “Per Anhalter durch die Galaxis”2 lesen, aus dem dieses Zitat stammt.

https://de.wikipedia.org/wiki/42_(Antwort)]

Per Anhalter durch die Galaxis wurde von Douglas Adams geschrieben

Kapitel 6

Sonstige Sprachmittel

Wir haben in den vorangegangenen Abschnitten besprochen, wie man alles schön formattiert. Doch was ist, wenn man in seinem Text bestimmte Zeichen unverändert darstellen möchte, also einen Stern als Stern, eine Tilde als Tilde und so weiter? Gleiches gilt für die gestalterischen Möglichkeiten. Wie kann man aus dem eingeschränkten Sprachumfang ausbrechen und bestimmte Dinge umsetzen, für die Markdown selbst keine Möglichkeiten bereithält und für die es auch keine Erweiterungen gibt? Nun, auch daran wurde gedacht.

6.1 Escaping - das besondere Entschärfen

Insbesondere bei den Spanelementen dürfte man erkannt haben, dass man mit bestimmten Zeichen im Text vorsichtig umgehen muss, da diese eine Formattierung bewirken und nicht ausgegebenen werden. Das kann man verhindern, indem man den jeweiligen Symbolen einen Backslash "\" voranstellt.

```
Lorem ipsum dolor \*sit amet\*, consetetur _sadipscing elitr_,
sed diam nonumy eirmod tempor invidunt ut _**labore et** dolore_
magna aliquyam erat, sed diam voluptua.
```

Lorem ipsum dolor *sit amet*, consetetur *sadipscing elitr*, sed diam nonumy eirmod tempor invidunt ut ***labore et*** *dolore* magna aliquyam erat, sed diam voluptua.

Das war auch schon die ganze Magie.

6.2 HTML einbetten

Wir sprachen bisher immer nur davon, dass Markdown sich am Web orientiert. Alle Markdown-Converter generieren mindestens die HTML-Code. Daher ist es nur nachvollziehbar, dass man innerhalb von Markdown auch HTML-Fragmente platzieren kann. Bei der Verwendung von HTML ist nichts weiter zu beachten und es besteht auch keine Limitierung. Ein guter Autor achtet allerdings stets darauf, dass das Ergebnis-HTML semantisch korrekt ist.

Auch wenn Browser meist visuell das erzeugen, was man sich mit Tricksereien zusammengebastelt hat, ist es zum Beispiel nicht zulässig, Block-Elemente in von Span-Elementen zu verpacken. So sollten Ausdrücke wie `<b><table>...</table></b>` (bzw. in Markdown geschrieben `*<table>...</table>*`) unbedingt verhindert werden.

Ansonsten gibt es wirklich keine Einschränkung bei der Verwendung von HTML-Fragmenten. Allerdings ist zu empfehlen, möglichst sparsam damit umzugehen, um die großen Vorteile[1] der Auszeichnungssprache Markdown nicht zu stören oder gar zunichte zu machen.

Wir wollen den Abschnitt Markdown mit einem Tipp abschließen, der eine in Markdown nicht enthaltende Formattierung mit Hilfe von HTML doch möglich macht: dem Unterstreichen.

```
Lorem ipsum dolor *<u>sit amet</u>*, consetetur <u>_sadipscing elitr_</u>,
sed diam <span style="border-bottom:1px double #a00">nonumy</span> eirmod tempor invidu
```

```
Lorem ipsum dolor <em><u>sit amet</u></em>, consetetur <u><em>sadipscing elitr<em></u>
sed diam <span style="border-bottom:1px double #a00">nonumy</span> eirmod tempor invidu
```

Lorem ipsum dolor *sit amet*, consetetur *sadipscing elitr*, sed diam nonumy eirmod tempor invidunt ut.

[1] Einfachheit und ein ungestörter Lesefluss im Rohdokument (Quelltext)

Teil III

Pandoc und KindleGen

Nachdem wir uns mit den Möglichkeiten von Markdown auseinandergesetzt haben, wollen wir uns abschließend mit der Übersetzung unserer Markdown-Ergebnisse in andere Formate beschäftigen, von denen viele für eBooks verwendet werden. Das ist vor allem dann wichtig, wenn der potentielle Leserkreis aus Personen besteht, die Markdown nicht kennen (wollen). Wir holen sie ab, indem wir ihnen Dateien bereitstellen, die zum Lesen elektronischer Lektüre besser geeignet ist. Man denkt spontan an EPub oder gar dem nativen Amazon-Kindle-Format. Dass ggf. auch HTML-, PDF- oder gar DOCX-Dateien möglich sind, wird manchem vielleicht ein "gut zu wissen" entlocken und für andere eine Mindestanforderung darstellen. Der vorliegende Abschnitt wird genau dieses Thema behandeln.

Ziel ist, den Funktionsumfang und eventuelle Besonderheiten der Werkzeuge aufzuzeigen, mit denen wir aus unserem Rohmaterial ein eBook erstellen. An der Bedienung der Kommandozeile werden wir nicht vorbeikommen, da es sich bei den vorgestellten Lösungen um reine Kommandozeilenwerkzeuge handelt. Wer dies vermeiden möchte, sollte einen Blick auf Calibre werfen, muss dann aber auf die Unterstützung von Markdown, HTML und anderen, eher technischen Ausgabeformate verzichten. Neben der Möglichkeit zur Konvertierung stellt das Programm allerdings auch eine Bibliothek bereit, mit der die eBooks verwaltet werden können.

Im folgenden fokussieren wir uns auf die Kommandozeilenwerkzeuge PanDoc und KindleGen. Wir verfolgen nämlich ein bisher unerwähntes Sekundärziel und wollen unsere Dokumente automatisch (also ohne weitere Interaktion) in mehr als nur ein Format zu konvertieren. Ein mögliches Einsatzszenario ist, ein auf Markdown aufsetzendes CMS- oder Bloggingsystem, mit Exportfunktionen auszustatten, die dem Besucher ihre Inhalte zum Offlinelesen bereitstellen.

Kapitel 7

Pandoc

Wir wollen uns für's erste dem deutlichen mächtigeren Werkzeug "PanDoc" zuwenden. Dieses Tool ist modular aufgebaut. Es besteht aus einem Kern, der über sogenannte Filter um die Unterstützung der verschiedenen Ein- und Ausgabeformate erweitert wird. Die Funktionsweise der Filter kann sehr einfach folgendermaßen erklärt werden:

1. Übersetzung des Eingabeformats in ein PanDoc-internes Datenmodel
2. Übersetzung des PanDoc-internen Datenmodels in das Ausgabeformat

Dabei stellen das Eingabe- und Ausgabeformat jeweils eines der vielen, von Haus aus unterstützten Formate dar, die wir im folgenden für unser Vorhaben verwenden. Wer weiterführende Details zum internen Ablauf benötigt oder PanDoc mit Hilfe von Haskell oder Python um eigene Filtermechanismen erweitern möchte, findet sie unter anderem unter https://pandoc.org/filters.html.

Uns soll dieser kurze "Blick unter die Motorhaube" genügen, um zu verstehen, wie PanDoc intern arbeitet.

Etwas ausführlicher gehen wir nun auf die Verwendung von PanDoc ein, und zwar in dem Umfang, als dass wir die Grundfunktionen besprechen und Anreize geben, mit denen jeder möglichst schnell ans Ziel kommt. Aufgrund des enormen Funktionsumfang müssen jedoch hier Abstriche in Kauf genommen werden, die mit dem offiziellen Handbuch[1] sehr schnell kompensiert werden können.

7.1 Installation

PanDoc kann von pandoc.org für viele Betriebssysteme (Windows, diverse Linux'e) heruntergeladen werden.

[1] Das Handbuch steht als Online als HTML-Seite (https://pandoc.org/MANUAL.html) und alternativ in Form eines PDF-Dokuments (https://pandoc.org/MANUAL.pdf) bereit.

Unter Windows verwendet lädt man entweder die `msi`-Datei (als Installationspaket) oder das `zip`-Archiv (für eine manuelle Installation) herunter. Unter Linux kann man meist auf die Version aus dem jeweiligen Repository des Betriebssystems zurückgreifen. Hier ist jedoch Vorsicht geboten, denn unter Ubuntu und Linux Mint wurde bis Anfang 12/2018 noch eine alte 1.9.x Version angeboten. Seinerzeit war mit Version 2.5 bereits eine deutlich umfangreichere Version bereits im Umlauf. Aus diesem Grund wird auch für Linux auf die manuelle Installation zurückgegriffen.

Für Windows und Linux gelten bei manueller Installation diese Schritten:

1. Download des zum System passenden Archivs https://github.com/jgm/pandoc/releases/tag/2.5
2. Entpacken in ein Verzeichnis, dass wir als PanDoc-Installationsordner ausgesucht haben

Das war es eigentlich schon. Wir sollten aber noch einen Test vornehmen, bevor wir uns freuen. Dafür empfiehlt sich bei den meisten Kommandozeilenprogrammen die Abfrage ihrer Versionsnummer:

```
<pathto>/pandoc --version
```

Das Programm sollte ohne weitere Fehlermeldungen[2] unter anderen die Versionsnummer ausgeben. Wir haben dabei die Langform des Parameters verwendet. Das hat den Grund, dass sich aus dem Parameternamen bereits dessen Bedeutung ableiten lässt. Wir werden später noch weitere kennenlernen, bei denen aus der abgekürzten Form nicht unbedingt abgeleitet werden kann, um was es sich handelt. Oder würden sie bei der Verwendung von `-f` sofort auf das Wort `--from` schließen?

Doch bevor wir uns mit der Verwendung auseinandersetzen wollen, verfeinern wir unsere Umgebung dahingehend, als dass wir uns in Zukunft sparen wollen, ständig den gesamten Pfad zu `pandoc` eingeben zu müssen. Dafür erweitern wir den Suchpfad um das Installationsverzeichnis unseres neuen Werkzeugs. Wer das nicht mag, kann den gesamten Abschnitt #path-erweitern überspringen und gleich mit den dem Kapitel #erste-schritte fortfahren.

7.1.1 PATH erweitern

Die nachfolgenden Schritte sind nicht verpflichtend. Vielmehr erleichtert er nur die Verwendung von PanDoc auf der Kommandozeile in der Form, als dass man sich den absoluten Pfad zu PanDoc sparen kann.

[2] Unter Linux kann es erforderlich sein, mittels `chmod +x <pathto>/pandoc` das Programm ausführbar zu machen

7.1.1.1 Linux

Wer unter Linux Programme installiert, muss sich nicht darum kümmern, den Suchpfad zu erweitern, damit das Programm auch ohne Angabe des absoluten Pfades gefunden wird. Es gibt viele Möglichkeiten, die sich von Desktop zu Desktop unterscheiden. Da wir PanDoc aber über die Kommandozeile bedienen wollen, beschreiben wir hier das Setzen der Variable für die BASH. Andere Shells verwenden andere Dateien, in denen unter anderem die PATH-Variable gesetzt wird, die alle Verzeichnisse aufnimmt, in denen nach Programmen gesucht werden soll. Welches es sind, beantwortet die Dokumentation oder kann unter Zurhilfenahme der Suchmaschine seines Vertrauens sehr schnell ermittelt werden. Die Suchworte "Linux PATH erweitern" sollten dabei helfen.

In der BASH kann der PATH sogar an mehreren Stellen eingestellt werden. Es bietet sich aber an, diesen in der Datei .bashrc anzupassen, welche sich im Verzeichnis des Benutzers befindet.

Datei `$HOME/.bashrc` öffnen und am Ende derselbigen folgende Zeile hinzufügen, wobei `<pathto-pandoc>` durch den vollqualifizierten/absoluten Pfad zum Verzeichnis ersetzt werden soll, indem sich das Kommandozeilenprogramm `pandoc` befindet.

```
PATH=$PATH:<pathto-pandoc>
```

Der Erfolg kann sofort getestet werden, indem man PanDoc erneut mit der Ausgabe der Versionsnummer sinnfrei beschäftigt. Allerdings verzichten wir dieses mal auf die absolute Pfadangabe und rufen das Programm nur noch mit seinem Namen auf.

```
pandoc --version
```

7.1.1.2 Windows

Auch Windows kennt PATH-Variablen und verwendet sie analog zu Linux. Unter Windows 7 findet man die Option zum Setzen/Erweitern der PATH-Variable im Dialig "Erweiterte Systemeinstellungen anzeigen" und darin den Button "Umgebungsvariablen". In diesem Dialog wird zwischen benutzerspezifischen und globalen (Systemvariablen) Umgebungsvariablen unterschieden. Wer PanDoc jedem eingerichteten Windowsbenutzer komfortabel im Suchpfad bereitstellen möchte, hängt die Pfad des Installationsverzeichnisses der Systemvariable PATH hinten an. Ansonsten ist die Benutzervariable PATH damit Installationspfad (ausschließlich das Verzeichnis) zu erweitern.

Änderungen von Variablen zeigen sich in der Kommandozeile erst, wenn man diese NACH dem Editieren öffnen. In laufenden Instanzen werden diese Änderungen nicht übernommen. Daher öffnen wir nun nach der PATH-Erweiterung eine neue CMD- oder PowerShell-Box, bevor wir unsere Veränderung testen können.

```
pandoc --version
```

7.2 Erste Schritte

Die von PanDoc unterstützten Eingabe-/Inputformate kann man sich mit dem nachfolgenden Befehl anzeigen lassen:

```
$> pandoc --list-input-formats
```

Analog dazu erhält man mit dem Argument `--list-output-formats` eine Liste aller Ausgabeformate:

```
$> pandoc --list-output-formats
```

Insbesondere die letztgenannte PanDoc-Funktion ist interessant, wenn man aus einer Eingabedatei alle erdenklichen Ausgaben erzeugen möchte. Das Skript `build.sh` [3], mit denen das in Markdown verfasste Buch in umgewandelt wurde, zeigt, wie schnell man damit ans Ziel kommen kann.

7.3 Optische Aufwertung

Nun haben wir aber erfahren, dass PanDoc sehr viele Ausgabeformate kennt, bei denen sehr wohl optische Belange zum Tragen kommen. DOCX, PDF und HTML[4] sind nur einige. HTML haben wir in unsere Aufzählung bewusst aufgenommen, weil neueste Versionen der verschiedenen eBook-Formate (also EPUB & Co.) HTML als Basis verwenden. Es liegt daher auf der Hand, dass wir uns im Folgenden dieses Exportformat genauer betrachten und vor allem auf dessen optische Aufwertung beleuchten.

7.3.1 Export als HTML

Um aus Markdown ein HTML-Dokument zu erstellen, bedarf es nicht zwangsläufig PanDoc. Hier könnte man auch auf das Perlskript zurückgreifen, dass die Markdown-Begründer bereitstellen. Wir wollen aber ein eBook erstellen und setzen hierfür `pandoc` ein.

Doch wie teilen wir PanDoc mit, dass wir: - Markdown als Quelle verwenden? - ein HTML-Dokument erstellen wollen? - unsere eigenen Formatvorgaben in das Dokument eingebunden bekommen? - Bilder einbetten, auf die wir im Markdown-Dokument verwiesen haben?

[3] Ein für Windows verwendbares `build.cmd` wird nachgereicht

[4] HTML ist zwar auch nur eine andere Auszeichnungssprache, unterstützt aber durch einbindung von CSS und JavaScript eine optische Aufwertung des Erscheinungsbildes, wie es sich im Interpreter (Browser) zeigt.

All das verrät uns der nachfolgende Befehl:

```
$> pandoc ebook.md --standalone --toc --toc-depth=4 --number-
sections --top-level-division=part --css=resources/styles.css --
from=markdown --to=html
```

Wer sich das Ergebnis im Quelltext anschaut, wird feststellen, dass sehr viele Headerinformationen eingespeist werden. Das liegt am Parameter `--standalone`. Wer HTML ohne dem ganzen Schnickschnack drumherum benötigt, weil er das Ergebnis beispielsweise in ein eigenes HTML-Template einbetten möchte, verzichtet einfach auf diese Option. Als Ergebnis wird alles ausgegeben, was wir innerhalb des `<body>`-Tags finden.

Betrachten wir uns nun die Übersetzung unseres Markdowns in HTML an, finden wir neben dem sauber aufbereiteten Text noch ein Inhaltsverzeichnis, das in einem Tag mit der `id="TOC"` eingebettet wurde. Es beinhaltet Links alle Überschriften, die wir in unserem Dokument definiert haben und erleichtert damit die Navigation innerhalb unserer Arbeit ungemein. Wer auch das nicht mag, befreit den Befehl von den Argumenten `--toc` und `--toc-depth=4`. Im übrigen schränken wir mit `--toc-depth=4` das Inhaltsverzeichnis auf die obersten 4 Überschriften-Ebenen ein. Überschriften der fünften oder sechsten Ebene werden nicht mit ins Inhaltsverzeichnis aufgenommen. Durch verändern der Zahl auf 6 oder weglassen des Attributs `--toc-depth` wird die Limitierung aufgehoben.

Dass die Übersetzung unseres Markdown-Dokuments semantisch korrekt ist, sehen wir an der übrigen Struktur. PanDoc bettet all unsere Abschnitte in `<section>`-Tags ein, wobei die jeweilige Überschrift das erste Element ist, gefolgt von dessen Inhalt. Nicht nur, dass solche sauberen Strukturen für Suchmaschinen optimal sind (Stichwort SEO[5]), hilft uns das später auch bei der optischen Gestaltung.

7.3.2 Cascading Stylesheets

Um die optischen Belange kümmert man sich in HTML mit den “Cascading Stylesheets” (kurz: “CSS”). Auch hierfür stellt PanDoc einen Parameter bereit, über den man seine individuellen Gestaltungsregeln einspeisen kann. Diese werden von PanDoc “nur” verlinkt und müssen deshalb gemeinsam mit dem HTML weitergegeben werden, wenn man das Ergebnis auf einem anderen PC begutachten möchte.
Im einfachsten Fall genügen uns die Vorgaben, die PanDoc mitliefert. Wer nur ohne `--standalone` exportiert hat, reduziert die Formatierung seines Ergebnisses auf die Standardeinstellungen des Browsers. Wir wollen aber mehr, nicht wahr? Welche Dinge müssen wir beachten, wenn wir unsere eigenen Styles formulieren?

[5] Bei der Search Engine Optimization (kurz SEO, dt. Suchmaschinenoptimierung) achtet man unter anderem auf eine klare, semantisch korrekte Strukturierung seiner Internetseiten, um das Suchmaschinenraking positiv zu beenflussen. Je besser das Ranking ausfällt, desto besser (weiter vorn) erscheint die Seite in den Suchmaschinenergebnissen.

7.3.2.1 CSS für Standard-Markdown

Einen ersten Ansatz geben uns alle HTML-Tags, die wir im Kapitel Markdown behandelten. Auf das Markdown in seiner ursprünglichen Form reduziert, müssen wir folgende Elemente berücksichtigen:

- Spanelemente `<emp>`, `<strong>`, `<a>`, `<img>`, `<code>`, `<pre>`
- Überschriften und Abschnitte `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`, `<p>`
- Horizontale Linien `<hr>`
- Zitate `<blockquote>`
- Listen `<ul>`, `<ol>`, `<li>`, sowie `<dl>`, `<dt>`, `<dd>`

```
/* Inline-/Spanelemente */
emp    { /* Formatierung des <emp>-Tags  */ }
strong { /* Formatierung des <strong>-Tags */ }
a      { /* Formatierung des <a>-Tags    */ }
img    { /* Formatierung des <img>-Tags  */ }
code   { /* Formatierung des <code>-Tags */ }
pre    { /* Formatierung des <pre>-Tags  */ }

/* Blockelemente */
h1     { /* Formatierung des <h1>-Tags */ }
h2     { /* Formatierung des <h2>-Tags */ }
h3     { /* Formatierung des <h3>-Tags */ }
h4     { /* Formatierung des <h4>-Tags */ }
h5     { /* Formatierung des <h5>-Tags */ }
h6     { /* Formatierung des <h6>-Tags */ }

hr     { /* Formatierung des <hr>-Tags */ }

blockquote { /* Formatierung des <blockquote>-Tags */ }

ul     { /* Formatierung des <ul>-Tags */ }
ol     { /* Formatierung des <ol>-Tags */ }
li     { /* Formatierung des <li>-Tags */ }

dl     { /* Formatierung des <dl>-Tags */ }
dt     { /* Formatierung des <dt>-Tags */ }
dd     { /* Formatierung des <dd>-Tags */ }
```

Wer HTML in sein Markdown einbettet, sollte natürlich auch an die darin enthaltenden HTML-Tags bei der Gestaltung seines Layouts denken.

Auch wenn durchaus buchspezifische Belange definiert wurden, lohnt ein Blick in die Datei https://github.com/hofrichter@@@styles.css@@@. Hierin fällt auf, dass bestimmte Tags kontextabhängig gestaltet wurden. So unterscheiden wir beispielsweise beim

<code>-Tag, ob er als Element innerhalb eines Fließtextes vorkommt oder als Kindelement des <pre>-Tags mehrzeilige Codebeispiele umklammert. Dem wurde beispielsweise mit dieser Anweisung Rechnung getragen:

```
code {
    background-color: #eee;
}
pre {
    border: 1px solid #999;
    background-color: #f3f3f3;
}
pre code {
    background-color: transparent;
}
```

7.3.2.2 CSS für PanDoc's Standard-Markdown-Interpretation

Wie wir eingangs festgestellt haben, werden die einzelnen Kapitel schön säuberlich in <section>'s verpackt. Die einleitende Überschrift ist darin das erste Element, gefolgt vom Fließtext (<p>-Tags) oder den anderen Blockelementen. Wer sich den Quelltext aus unserem ersten PanDoc-Lauf genauer anschaute, wird feststellen, dass jeder <section>-Tag über zwei Attribute verfügt:

- `class` mit der Information, welcher Überschriftenebene der Abschnitt zugeordnet wurde
- `id` mit leicht abgewandelter Schreibweise der Überschrift

Die CSS-Klasse nimmt je nach Überschriftenebene folgende Werte an: - level1 - level2 - level3 - level4 - level5 - level6

Bezüglich der `id` sei erwähnt, dass der HTML-Standard ID's keine Leerzeichen enthalten dürfen[6]. PanDoc verwendet den Text der Überschriften als ID und ersetzt Leerzeichen durch Bindestriche `-`. Auch wird der gesamte Text kleingeschrieben. Verweise auf eine Überschrift müssen diese Regel befolgen, um nach Übersetzung in ein anderes Format noch gültig zu sein. Ein Beispiel soll dies veranschaulichen:

```
# Meine erste Überschrift

# Ende
...
Zurück zum Abschnitt [#meine-erste-überschrift](Meine erste Überschrift).
```

Um für unser eBook aus den Vollen schöpfen zu können, müssen wir bei unserer optischen Aufbereitung auch jene Dinge berücksichtigen, die wir im Abschnitt

[6] Gemäß https://www.w3.org/TR/html4/types.html#type-id dürfen IDs nur aus Buchstaben ([A-Za-z]), Zahlen ([0-9]), Bindestrichen ("-"), Unterstrichen ("_"), Doppelpunkten (":"), und Punkten (".") bestehen. Eine ID muss zudem mit Buchstaben beginnen.

Markdown als Erweiterung kennzeichnet haben. Hierzu gehören vor allem Fußnoten und Tabellen.

7.3.2.3 CSS für Tabellen (Markdown Erweiterung)

Die Formatierung von Tabellen möchten wir nicht im Detail besprechen, weil es hierfür einfach zu viele Möglichkeiten und Anforderungen gibt. Es lohnt sich aber noch einmal einen genaueren Blick auf die HTML-Struktur zu werfen, die PanDoc aus dem Markdown erzeugt. Viele vergessen, dass zwischen dem `<table>`-Tag und dem ersten `<tr>` noch `<thead>` und `<tbody>` in dem DOM-Baum integriert werden sollte, um zwischen Tabellenüberschrift und -inhalt zu unterscheiden. Semantisch korrekt wird die Tabelle allerdings erst, wenn für die Zellen des Tabellenkopfes `<th>`-Tags verwendet werden und nicht `<td>`. Auch das wird von PanDoc berücksichtigt.

Somit können wir unsere CSS-Regeln wie folgt erweitern, um die Möglichkeiten der Tabellengestaltung vollständig zu erfassen:

```
table { /* Formatierung des <table>-Tags */ }
thead { /* Formatierung des <thead>-Tags */ }
tbody { /* Formatierung des <tbody>-Tags */ }
thead tr { /* Formatierung des <tr>-Tags innerhalb des Tabellenkopfes */ }
tbody tr { /* Formatierung des <tr>-Tags innerhalb des Tabellenrumpfes */ }
th { /* Formatierung des <th>-Tags als einzelne Zelle im Tabellenkopf */ }
td { /* Formatierung des <th>-Tags als einzelne Zelle im Tabellenkopf */ }
```

7.3.2.4 CSS für Fußnoten (Markdown Erweiterung)

PanDoc übersetzt Fußnoten in folgende HTML-Fragmente

```
Im Fließtext platzierte Fußnote<a href="#fn1" class="footnote-ref" id="fnref1"><sup>1</
und hier<a href="#fn2" class="footnote-ref" id="fnref2"><sup>2</sup></a>.
```

```
<section class="footnotes">
  <hr>
  <ol>
    <li id="fn1"><p>Text der Fußnote 1<a href="#fnref1" class="footnote-back"></a></p>
    <li id="fn2"><p>Text der zweiten Fußnote<a href="#fnref2" class="footnote-back"></a
    ...
```

Wie wir sehen, kommen Elemente zum Einsatz, die auch im Standard-Markdown verwendet werden können. Allerdings werden diese um `id`- und `class`-Attribute erweitert. Die Werte der IDs sollen uns beim Layout nicht primär interessieren.

Uns geht es vor allem um die CSS-Klassen, die wir in unserem Cascading Style Sheet ansprechen, um ihnen einen individuellen Touch zu geben.

```
.footnote-ref { /* Formatierung des <a>-Tags, der im Fließtext auf eine Fußnote verweist */ }
.footnotes ol { /* Formatierung des <ol>-Tags aller Fußnoten */ }
.footnotes li { /* Formatierung des <li>-Tags einer einzelnen Fußnote */ }
.footnotes li p { /* Formatierung des <p>-Tags einer einzelnen Fußnote */ }
.footnotes a.footnote-back { /* Formatierung des <a>-Tags (Rücksprungmarke) einer einzelnen Fußno
```

Wichtig ist, dass wir in unserem zwischen `.footnotes a.footnote-back` und `.footnotes a` unterscheiden sollten, weil letztere Option auch in Fußnoten platzierte Verweise auf andere Seiten (bspw. bei Quellenangaben) mitformatieren würde, was wir unter Umständen nicht wollen.

7.3.3 Weitere PanDoc-Funktionen und -Eigenschaften

7.3.3.1 Externe Ressourcen

Wer sich das vorliegende Buch einmal im Rohform anschaut, wird bemerken, dass wir an der einen oder anderen Stelle bewusst darauf verzichteten, Beispielgrafiken lokal vorzuhalten und stattdessen mit externen Adressen (erkennbar am Präfix "https://" oder "http://") arbeiten. Das hat den Grund, dass wir damit vor allem nachweisen wollen, dass PanDoc solche externen Ressourcen verarbeiten kann. Im HTML-Ergebnis kann man keine Änderung feststellen, im finalen eBook-Format werden diese Dinge jedoch in interne Verweise umgewandelt und die Grafik mit in die Datei integriert. PanDoc lädt die Datei somit herunter, um sie dem eBook beizufügen. Das kann zwei Probleme verursachen:

1. PanDoc arbeitet nicht fehlerfrei, wenn ihm der Zugriff auf das Internet verwehrt wird. In diesem Fall sollte man die Datei herunterladen und mit lokalen Pfaden arbeiten.
2. Um Urheberrechtsverletzungen zu vermeiden, sollten vor eBook-Veröffentlichungen die Eigentumsrechte geklärt oder die Zustimmung vom Urheber eingeholt werden

7.3.3.2 Cover und Metadaten

Wir haben bisher erfahren, wie wir unsere Inhalte technisch umsetzen. Bücher bestehen allerdings auch aus einem hübsch gestalteten Buchdeckel und einleitenden Seiten, auf denen wir vor allem Angaben, wie Erscheinungsdatum, Autor, Titel oder auch Untertitel finden.

PanDoc stellt hierfür einen eigenen Mechanismus bereit. Das ermöglicht eine individuelle und vom Inhalt strikt getrennte Gestaltung des Rahmens zu. Man kann sie als Parameter des `pandoc`-Befehls mitgeben, in einer separaten `yaml`-Datei definieren oder in das Markdown-Dokument einbetten. Bei beiden Formen weist

man PanDoc-spezifischen Attributen Werte zu. Dabei ist keines der Attribute Pflicht. In wie weit diese Aussage relativiert werden muss, kann sich jeder selbst denken. So kann man es als weitestgehend sinnfrei ansehen, seinem eBook keinen Titel (`title`) zu geben.

Doch genug des Wortspiels. Das folgende Beispiel zeigt die Metadaten der initialen Version des Buches.

```
---
title: Mit Markdown zum eBook (Betaversion)
subtitle: Die alternative Technik der eBooks-Erstellung
date: 01/2019
rights: © 2019 Sven Hofrichter, ebooks@sugarfree.im, CC BY-NC
author: Sven Hofrichter
author.affiliation: JEE-Architect @ Finanzinformatik Solutions Plus GmbH
publisher: Sven Hofrichter
keywords: [eBook,epub,Markdown,pandoc,kindlegen,erstellen]
cover-image: resources/cover.jpg
abstract: |
  Dieses Buch beschreibt, wie man mithilfe von Markdown, Pandoc und KindleGen
  eBooks erstellen kann und diese in unterschiedlichen Formaten bereitstellt.
lang: de-DE
---
```

Abschießend sei erwähnt, dass die gezeigten Attribute nur eine Teilmenge möglicher Angaben darstellen. Die vollständige Liste findet man im PanDoc-Manual hier: http://pandoc.org/MANUAL.html#epub-metadata.

Kapitel 8

KindleGen

Ein Manko von PanDoc ist, dass mit ihm keine eBooks im Kindle-Format (Dateiendung `.azw`) erstellt werden kann. Das ist zwar ein Wermutstropfen, aber kein allzu großes Manko. Denn Amazon bietet selbst ein Kommandozeilentool an, mit dem man aus einer geringen Zahl an Eingabeformate (unter anderem EPUB) das genannte proprietäre Format überführen kann.

8.1 Installation

Anders als bei PanDoc muss man für KindleGen leider ein wenig suchen, denn Amazon stellt dieses Werkzeug zwar bereit, verweist jedoch (eine Vermutung) nur in seinen Hilfeseiten darauf. Man findet es aber meist sehr schnell mit Hilfe der Suchmaschine seines Vertrauens. Im Dezmeber 2018 konnte man es unter dieser Adresse finden https://www.amazon.com/gp/feature.html?docId=1000765211. Auch dieses Werkzeug steht für verschiedene Betriebssysteme bereit. Seine Installation erfolgt durch einfaches Entpacken eines ZIP-Archivs (Linux) oder durch Starten der Installationsroutine (Windows).

Sobald das Werkzug auf dem Zielrechner ausgerollt wurde, sollte wieder geprüft werden, ob selbiges fehlerfrei gestartet werden soll. Da das Werkzeug leider nicht über einen Parameter zur Abfrage der Versionsnummer verfügt, uns aber sein "usage" ausgibt, wenn wir es ohne Argumente aufrufen, greifen wir zu Prüfung darauf zurück:

```
$> kindlegen
```

8.2 Erste Schritte

KindleGen unterstützt eine deutlich geringere Zahl an Inputformaten, was für uns aber kein Problem darstellt, weil wir uns mit PanDoc weiterhelfen können. Dass dabei Verluste entstehen können, die sich negativ auf das Endergebnis auswirken können, ist verständlich und auch Grund dafür, dass wir bereits am Anfang Markdown zum Erfassen des Buches verwenden.

Um nun auf die Zielgerade einzubiegen und auch das Kindle-Format bedienen zu können, stellen wir unser Ergebnis als `epub` bereit.

Ausgehend davon, dass wir eine Quelldatei namens `ebook.md` haben und es mit pandoc's Hilfe in das epub-Format überführen wollen, ergibt sich folgende, vorbereitende pandoc-Anweisung:

```
$> pandoc ebook.md --toc --toc-depth=4 --number-sections --top-
level-division=part --from=markdown --css=resources/styles.css --
to=epub --output=ebook.epub
```

Wir gehen an dieser Stelle davon aus, dass der Autor das EPUB-Ergebnis bereits geprüft und für korrekt umgesetzt befunden hat und gehen gleich zum Folgeschritt über, der das eBook im das Kindle-native Format übersetzt.

```
$> kindlegen ebook.epub -o ebook.awz
```

Als Zieldatei haben wir im Beispiel `ebook.@@@` angegeben. Nun unterziehen wir diese Datei einer optischen Prüfung und kopieren sie hierfür auf einen AmazonKindle. Wer ein solches Gerät nicht hat, kann auf die Kindle-Applikationen zurückgreifen, die für verschiedene Betriebssysteme bereitstehen. Es gibt auch ander eBook-Reader-Applikationen, die unter anderem mit dem Kindleformat umgehen können, die man alternativ verwenden kann. Es wird allerdings empfohlen, auf die Produkte von Amazon zurückzugreifen, da wir nur dadurch sichergehen können, dass die Anwendungen das Ergebnis auch wirklich anzeigen und nicht aufgrund irgendwelcher Neuerung die Darstellung verweigern.

Teil IV

Abschluß

Wie wir gesehen haben, muss man nicht immer nur mit großen und aufwändigen Werkzeugen hantieren, um ans Ziel zu kommen. Manchmal sind es die schlichten Dinge, die uns dem Ziel näher bringen und auch Freiheitsgrade öffnen, die wir in anderen Lösungen schmerzlich vermissen, es aber eventuell erst zu spät realisieren.

Ich hoffe, dass ich mit dieser kleinen Buch ein wenig Interesse wecken konnte. Dass dieses eBook aus der Not heraus entstand, um mein eigentliches Thema zu stemmen, darf an dieser Stelle gern als Versprechen oder Drohung betrachtet werden. Der Fokus war aus aus diesem Grund auch vorrangig technisch geprägt. Eine überarbeitete Version ist zu erwarten, sobald das ursprünglich anvisierte Buchthema mit den hier beschriebenen Mitteln umgesetzt wurde.

Wer das Vorwort überlesen hat und nun gespannt ist, welches Thema kurzfristig zurückgestellt wurde, darf sich auf etwas freuen, dass mit den drei Buchstaben `PWA` abgekürzt wird. Es handelt sich hierbei eher um eine Sammlung von Funktionen und Definitionen aus dem Webumfeld, was die Behandlung in Form einer Lektüre deutlich umfangreicher gestaltet. Umso wichtiger war es, sich mit den Werkzeugen zu beschäftigen, mit denen die Notizen strukturiert und in etwas Lesbares gebracht werden und zwar in einer Form, bei der der Autor nicht von seiner Arbeit abgelenkt wird.