Markdown und Mermaid

bja @ https://coffee21.de/?p=672

24. Dezember 2021

Version	Beschreibung
0.1	Initiale Version

1. Einführung

Denken wir heute an ein Schreibprogramm kommt uns sofort *Word* in den Sinn. Es ist sogar üblich das Wort *Word* als Synonym für das Wort *Schreibprogramm* zu verwenden. Ich finde das problematisch, weil Sprache eine Realität schafft, die schwer zu durchbrechen ist. Wir erschaffen damit eine Barriere die uns daran hindert den Blick zu weiten. Denn ist ein Schreibprogramm der Art von *Word* oder *Libreoffice Writer* immer sinnvoll? Es kommt eben auf den Einsatzzweck an. In diesem Artikel möchte ich eine andere Herangehensweise vorstellen, welche auf Textdateien basiert und einige Vorteile mitbringt.

Jeder hat sicherlich von dem Textsatzsystem TeX gehört, welches 1978 von dem herausragendem Informatiker Donald E. Knuth veröffentlicht wurde. Dieses Programm liest eine Textdatei ein und erzeugt daraus eine binäre Dokumentdatei. Dies kann z.B. ein PDF-Dokument oder eine Libreoffice Writer Datei sein. Dieses Vorgehen hat den Vorteil, dass ich unabhängig bin von meiner binären Dokument-Datei. Ich bin in der Lage aus der Textdatei jede beliebige Dokumentdatei zu erstellen. Die Textdatei ist sozusagen meine Quelldatei und diese ist sehr nachhaltig, weil ich diese mit Betriebssystem-Werkzeugen wie einem Texteditor auch nach 50 Jahren noch öffnen und lesen kann. Außerdem kann ich eine Textdatei mit einem Versionsverwaltungssystem wie Git sehr gut versionieren und somit sehr einfach mit mehreren Personen an einem Dokument arbeiten. Zusätzlich kann ich eine Textdatei öffnen ohne zeitgleich eventuell darin enthaltenen Code auszuführen. Textdateien haben viele Vorteile, aber Textdateien mit TeX-Auszeichnungen haben den Nachteil, dass sie für Menschen schwer lesbar sind. Aus diesem Grund haben sich Textsatzsysteme wie Markdown oder AsciiDoc entwickelt. Deren Ziel ist eine Textdatei, welche leicht lesbar und gut überschaubar ist. Sie sind weniger mächtig wie TeX, aber in vielen Anwendungsszenarien völlig ausreichend.

2. Markdown

Markdown wurde im Jahr 2004 von John Gruber und Aaron Swartz veröffentlicht. Das Ziel war eine vereinfachte und leicht lesbare Auszeichnungssprache zu erstellen. In der Praxis ist die Auszeichnung von Text leicht einprägbar und wird von vielen Texteditoren durch z.B. Syntax-Highlighting unterstützt. Einen guten Überblick über die Syntax liefert die Projektseite unter https://daringfireball.net/projects/markdown/syntax. Seinen Siegeszug begann Markdown genau wie AsciiDoc in der Softwareentwicklung zur Dokumentation von Programmen. Hier spielte es seine Vorteile der Versionierbarkeit mit Versionsverwaltungssystemen wie z.B. Git und des einfachen öffnens und lesens einer Textdatei auf Serversystemen voll aus. Erst später kam Markdown beim Schreiben von Büchern am Anfang einer langen Prozesskette zum Einsatz. Hier kommt ein weiterer Vorteil von Textdateien zum Tragen. Textdateien lassen sich sehr gut mit Betriebssystemsoftware weiterverarbeiten, zusammenfügen oder trennen. Ist die Syntax bekannt, lässt sich sogar auf einzelne Felder wie z.B. der Titel oder die Überschriften zugreifen, die dann wiederum weiterverarbeitet werden können.

Markdown wandelt Textdateien mit Markdown-Syntax in XHTML-Dateien um. Zum ersten Testen empfehle ich den Online-Editor Dillinger. Auf der linken Seite steht der eigene Text in Markdown-Syntax und auf der rechten Seite aktualisiert sich permanent die Vorschau. Das fertige Dokument lässt sich am Ende als Markdown-, PDF- oder HTML-Datei herunterladen.

3. Pandoc

Pandoc ist ein mächtiger universeller Markup-Dokumentkonverter. Da Microsoft Word (docx) und LibreOffice (odt) auch nur in einem ZIP-Archiv verpackte Markup-Dateien sind, ist es mit Pandoc möglich diese Art von Dateien in Markdown, AsciiDoc oder LaTeX umzuwandeln und zurück. Aber auch PDF-Dokumente lassen sich generieren (nur in diese eine Richtung). Folgendes Kommando wandelt eine Textdatei mit Markdown-Syntax in eine PDF-Datei um:

```
pandoc -s quelldatei.md -o zieldatei.pdf
```

Pandoc bedient sich dafür im Hintergrund der Programme `pdflatex`, `lualatex`, `xelatex`, `latexmk`, `tectonic`, `wkhtmltopdf`, `weasyprint`, `prince`, `context` oder `pdfroff`. Auf meinem System, einem openSUSE Tumbleweed, verwendete pandoc `pdflatex`.

4. Mermaid

Einen ähnlichen Ansatz wie Markdown verfolgt das Werkzeug Mermaid. Nur eben nicht für Textdokumente, sondern für Diagramme, Charts und Grafen. Das

Projekt bietet ebenfalls einen Online-Editor an. Auch hier ist auf der linken Seite der Quellcode und auf der rechten Seite die Vorschau angeordnet. Kleiner Tipp: In der ersten Zeile steht das *TD* für die Aufbaurichtung des Diagramms. Im Beispiel des Online-Editors also *Top-Down*. Es gibt noch *LR* für *Left-Right*.

5. Installation

Ziel sollte es sein, den Mermaid-Sourcecode in ein Markdown-Dokument zu integrieren und daraus mit pandoc ein PDF-Dokument zu erstellen. Pandoc benötigt dazu einen Filter. Dieser lässt sich wie folgt installieren:

```
sudo npm -g install mermaid-filter
```

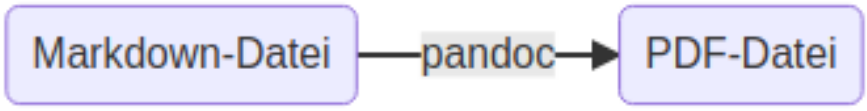
Im Markdown-Dokument könnte dann folgender Text enthalten sein:

````
> Beliebiger Text.
>
> ```mermaid
> graph LR
>    A(Markdown-Datei) -->|pandoc| B(PDF-Datei)
> ```
````

Anschließend lässt sich mit folgendem Befehl ein PDF-Dokument generieren:

```
pandoc --variable papersize=a4paper \
       --variable lang=de \
       -F mermaid-filter \
       -s quelldatei.md \
       -o zieldatei.pdf
```

Das PDF-Dokument enthält jetzt den automatisch fertig gerenderten Grafen. Ein Vorteil, was andere vielleicht als Nachteil empfinden, ist, dass der Erstellende sich keine Gedanken um das Layout machen muss. Außerdem lassen sich Änderungen an dem Grafen mit sehr geringem Aufwand durchführen.



6. Conclusion

Ziel war es einen kleinen Einstieg in Markdown zu geben und einen alternativen Weg bei der Dokumenterstellung darzustellen. Sehr empfehlen kann ich auch AsciiDoc und natürlich LaTeX.

Nachtrag

Auf einem anderen System (auch openSUSE Tumbleweed) schlug die Installation des mermaid-filters fehl. Eine ältere Version aus einem anderem Repository behob das Problem. Hinweis: Führe diese Kommandos nur aus, wenn Du ihren Sinn und Zweck verstehst!

```
sudo npm -g install mermaid-filter
/usr/local/bin/mermaid-filter -> /usr/local/lib/node_modules/
mermaid-filter/index.js

> puppeteer@13.0.1 install /usr/local/lib/node_modules/
mermaid-filter/node_modules/puppeteer
> node install.js

ERROR: Failed to set up Chromium r938248! Set
"PUPPETEER_SKIP_DOWNLOAD" env variable to skip download.
[Error: EACCES: permission denied, mkdir
'/usr/local/lib/node_modules/mermaid-filter/node_modules/
puppeteer/.local-chromium'] {
  errno: -13,
  code: 'EACCES',
  syscall: 'mkdir',
  path: '/usr/local/lib/node_modules/mermaid-filter/
  node_modules/puppeteer/.local-chromium'
}
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! puppeteer@13.0.1 install: `node install.js`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the puppeteer@13.0.1 install script.
npm ERR! This is probably not a problem with npm. There is
likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR!     /root/.npm/_logs/2021-12-24T11_43_06_794Z-debug.log

sudo rm /usr/local/bin/mermaid-filter
sudo npm -g i @shanewholloway/mermaid-filter
```