Howto zu pandoc und knitr

Formatierung in markdown

Überschriften

```
# Überschriftebene 1

## Überschriftebene 2

### Überschriftebene 3
```

Schrifttype ändern

Manchmal möchte man Dinge hervorheben. Dies kann ebenfalls in Markdown vorgenommen werden.

```
**fett**,
`monospace`,
_kursiv_,
__fettunterstreichen__,
\*so wird ein * erzeugt
```

Ausgabe:

- **fett**,
- `monospace`,
- *kursiv*,
- **fettunterrichen**,
- * so wird ein * erzeugt

Tabellen

Eine Tabelle lässt sich wie folgt darstellen. Dabei sind die Breiten der kurzen Striche entscheidend für die Spalten. Die Ausrichtung innerhalb der Zellen lässt sich durch einrücken (zentriert) oder ganz links (ohne Space! linksbündig) oder ganz rechts (ohne Space! rechtsbündig) herstellen. Die Zeilen der Tabelle trennt man durch Leerzeilen von einander. So können Texte über mehrere Zeilen fließen.

```
-------------------------------------------------------------------------
-
Code  SIP           HTTP          Request/Response  Klasse / Erläuterung
----- ------------- ------------- ----------------- ------------------------
-
100   Trying        Continue      Response          Informational

180   Ringing                     Response          Informational
```

```
200   OK            OK            Response          Success

401   Unauthorized  Unauthorized  Response          Client-Error

404   Not Found     Not Found     Response          Client-Error
      SUBSCRIBE                   Request           Presence Management
      NOTIFY                                  Presence Management
-------------------------------------------------------------------------
-
```

Bilder

```
![Bildbeschrifung im Text](./pfad/dateiname.png "reference")
```

Variablen

Variablen können im yaml-Format angegeben werden und im Dokumentenkopf eingebunden werden. Diese lassen sich dann z.B. in einem Latex-Template verwenden.

```
---
titlename:  'VoIP-Vertiefung'
filename: 'W02-VoIP 02 SIP Vertiefung'
username: 'Name: _________________'
classname: 'Klasse: ________'
docdate: 'Datum: ________'
---
```

Fußnoten

Fußnoten lassen sich Inline (also direkt hinter dem Begriff) wie folgt bilden:

```
SIP^[SIP: Session Initiation Protocol; Signalierungsprotokoll ähnlich
aufgebaut wie HTTP-Standard]
```

Dies würde im Text wie folgt aussehen, wobei die Fußnoten automatisch durchnummeriert werden und jeweils im unteren Bereich der Seite, auf der sie eingefügt wurden angezeigt werden.

Beispiel: SIP[1)]

Aufzählungen

Aufzählungen können mit * am Zeilenanfang dargestellt werden.

```
* Standard: IETF RFC 2543
* Protokoll ist ähnlich wie HTTP/SMTP aufgebaut (s. Fehler/Statuscodes)
```

So würde dies im Text aussehen:

- Standard: IETF RFC 2543
- Protokoll ist ähnlich wie HTTP/SMTP aufgebaut (s. Fehler/Statuscodes)

Latex-Template

Die Kopf-/Fusszeilen habe ich über das Latex-Template in Zusammenhang mit den oben erläuterten Variabeln gesteuert.

```
\usepackage{fancyhdr} %Paket laden
%\pagestyle{plain} %nur Seitenzahl in der Fußzeile (LaTeX-Standard)
\pagestyle{fancy} %eigener Seitenstil
\fancyhf{} %alle Kopf- und Fußzeilenfelder bereinigen
\fancyhead[L]{{$username$}  \\{$titlename$}} %Kopfzeile links
\fancyhead[C]{{$classname$}\\Werner-von-Siemens-Schule} %zentrierte
Kopfzeile
\fancyhead[R]{{$docdate$} \\Arbeitsblatt} %Kopfzeile rechts
\renewcommand{\headrulewidth}{0.4pt} %obere Trennlinie
\fancyfoot[L]{\today} %Seitennummer
\fancyfoot[C]{\thepage} %Seitennummer
\fancyfoot[R]{{$filename$}}
\renewcommand{\footrulewidth}{0.4pt} %untere Trennlinie
```

Um die Überschriften etwas kompakter zu gestalten habe ich folgende Latex-Zeilen in die Template-Datei eingefügt.

```
\usepackage{titlesec}
\titlespacing\section{0pt}{12pt plus 4pt minus 2pt}{0pt plus 2pt minus 2pt}
\titlespacing\subsection{0pt}{12pt plus 4pt minus 2pt}{0pt plus 2pt minus
2pt}
\titlespacing\subsubsection{0pt}{12pt plus 4pt minus 2pt}{0pt plus 2pt minus
2pt}
```

knitr - Der kleine Helfer

Tool zu Vereinfachung der Ausgabe kann folgendes Tool verwendet werden. http://yihui.name/knitr/demo/pandoc/ Damit man es nutzen kann wird die Programmiersprache R benötigt.

```
sudo apt-get install r-base
```

Die Command-Line von R erreicht man durch eintippen von R in die Shell.

Nun muss in R noch das Modul `knitr` installiert werden.

```
install.packages('knitr', dependencies = TRUE)
```

Sollte es dabei zu Fehlern kommen, dann kann mit folgendem Befehl evtl. Abhilfe geschaffen werden.

```
update.packages()
```

Damit `knitr` weiss, welches Template und welcher Converter verwendet werden soll, kann man dies entweder in einer eigenen Datei angeben oder direkt in die Markdown-Datei schreiben. Hier ein Beispiel für die zweite Variante, die den Vorteil von wenigen Dateien hat.

```
<!--pandoc
t: latex
latex-engine: xelatex
s:
number-sections:
template: default.latex
output: "W02-VoIP 02 Vertiefung.pdf"

t: odt
s:
number-sections:
template: default.odt
output: "W02-VoIP 02 Vertiefung.odt"

-->
```

Die Parameter hinter `t:` werden für den `format` Parameter des `knitr` Aufrufs verwendet. Alle anderen Zeilen stellen jeweils einen Parameter dar, der normalerweise hinter `pandoc` in der Kommandozeile getippt wird.

Um nun ein Dokument zu konvertieren können folgende Befehle verwendet werden.

```
library(knitr)
pandoc('W02-VoIP 02 Vertiefung.md', format='latex')
```

Als Ergebnis wird nun dieser Befehl ausgeführt:

```
pandoc 'W02-VoIP 02 Vertiefung.md'  -s --number-sections --latex-
engine=xelatex --template=default.latex -o 'W02-VoIP 02 Vertiefung.pdf'
```

Es ist einleuchtend, dass der kleine Umweg über `knitr` viel Schreibarbeit auf der Konsole spart. Zumal man die R-Umgebung bei verlassen speichern kann und damit die History der R-Befehle erhalten bleiben.

Um R wieder verlassen zu können muss man folgendes Command eingeben:

```
q()
```

IDE für markdown mit knitr Support: https://www.rstudio.com/products/rstudio/download/

Template für ein Arbeitsblatt

Das Rstudio ist sehr komfortabel, allerdings möchte man nicht bei jedem neuen Dokument alles erneut einstellen. Daher helfen ein paar kleine Eingriffe, um schnell neue Dokumente generieren zu können. Rstudio unterstützt Templates. Diese liegen unter Ubuntu in folgendem Ordner:

Unter Linux:

```
/usr/local/lib/R/site-library/rmarkdown/rmarkdown/templates
```

Unter Windows:

```
C:\Programme\RStudio\R-3.2.3\library\rmarkdown\rmarkdown\templates
```

Dort befindet sich pro Template ein Ordner.

Der Ordner besteht aus folgender Struktur

```
template_name
    resources -> z.B. css-Dateien für HTML-Output
    skeleton -> hier liegen alle Dateien, die für ein neues Dokument
benötigt werden
    template.yaml -> Name und Beschreibung des Templates für Rstudio
```

Ich habe mir ein Satz zusammengestellt, um meine Arbeitsblätter zu generieren. Diese sind sicherlich nicht perfekt, aber für den Anfang ausreichend. Arbeitsblatt_template Das Archive enthält die Ordnerstruktur, die in den `templates`-Ordner von R als `root` kopiert werden muss. Danach steht die neue Vorlage im Rstudio zur Verfügung.

knitr und pandoc automatisch in Rstudio laden

Da ich es als lästig empfinde die hauptsächlich eingesetzten Bibliotheken immer wieder neu laden zu müssen, habe ich nach einer Lösung gesucht, um diese während des Starts von R gleich mit zu laden. Da Rstudio die Standard R-Umgebung als Konsole anbietet, schlägt man gleich zwei Fliegen mit einer Klappe.

Die Lösung ist denkbar simpel. R lädt u.a. eine Datei `.Rprofile`, die im Hauptordner des Users liegen muss. Dort kann man häufig benötigte Bibliotheken einbinden.

Unter Linux:

```
/home/USERNAME/
```

Unter Windows:

```
C:\Users\USERNAME\Documents\
```

In der folgende Beispieldatei wird `knitr` geladen, so dass der `pandoc`-Befehl unter R direkt

ausgeführt werden kann.

.Rprofile

```
library(knitr)
```

pandoc/latex/knitr unter Windows

Folgende Tools werden benötigt bzw. damit geht es am schnellsten.

- R-Programmiersprache (https://cran.r-project.org/bin/windows/base/)
- RStudio (https://www.rstudio.com/products/rstudio/download/)
- pandoc (https://github.com/jgm/pandoc/releases)
- MikTex (http://miktex.org/2.9/setup)

Die Reihenfolge ist sinnvoll, damit keine Fehlermeldungen entstehen. Über das RStudio können weitere Pakete für MikTex bzw. R nachgeladen werden.

pandoc output

Math und das Format-Problem

https://github.com/jgm/pandoc/issues/1938

Formelsatz in `markdown` kann sehr unschöne Ergebnisse hervorbringen, um diese in den Griff zu bekommen sind einige Handgriff notwendig. Hier wird versucht eine Lösung für folgende Kombinationen anzubieten.

- `markdown` → `PDF`
- `markdown` → `dokuwiki`
- `markdown` → `epub`
- `markdown` → `HTML`

Als Beispiel-Code soll folgender Schnipsel dienen:

```
Ein wenig Mathematik

$$
\begin{align}
x&=\frac{3}{5}\label{eq:1}\\
y&=2
\end{align}
$$
```

```
End of math. $\eqref{eq:1}$
```

```
pandoc math.txt -t html -s -o test.html --
mathjax=https://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS-MML
_HTMLorMML -H mathjax-header-include.txt
```

mathjax-header-include.txt

```
<script type="text/x-mathjax-config">
    MathJax.Hub.Config({ TeX: { equationNumbers: {autoNumber: "all"} }
});
  </script>
```

1)

SIP: Session Initiation Protocol; Signalierungsprotokoll ähnlich aufgebaut wie HTTP-Standard

From:
http://www.kopfload.de/ - **kopfload - Lad Dein Hirn auf!**

Permanent link:
http://www.kopfload.de/doku.php?id=allgemein:howto:pandoc&rev=1457367965

Last update: **2025/11/19 16:13**