# Rumex Beschreibung

Stefan Blechschmidt

2013, 2014, 2015

## Vorwort

Auf dieser Seite versuche ich meinen Rumex Baukasten zu beschreiben. Eine Beschreibung zu erstellen schiebe ich schon seit Anfang an. Da ich Rumex mittlerweile schon seit drei Jahren verwende und in letzter Zeit keine großen Veränderungen mehr einbauen musste ist es Zeit sich der Beschreibung zu widmen.

**Beweggründe**

Es gibt immer mehr Systeme zum Erstellen und Verwalten von Internetseiten. Probiert habe ich schon viele und mit den meisten war ich auch sehr zufrieden. Was mich aber immer gestört hat waren die ständigen Updates.

*Irgendwie hat sich alles immer gebissen*

Einmal brauchte das, ich nenne es stellvertretend für alle Systeme, CMS eine besondere Version eines Programms. Dann war es wieder umgekehrt ein Update des Programms konnte nicht gemacht werden weil das CMS noch nicht damit zurecht kam.

Ab und zu kam es mir so vor als ob ich die meiste Zeit damit verbrachte das CMS und dessen Plattform dazu zu überreden mit einander zu arbeiten.

Ich wünschte mir immer mehr eine einfache Internetseite, eine HTML Seite, so wie in früheren Zeiten.

Irgendwann bin ich dann auf `markdown` und `pandoc` gestoßen und die Rumex Idee ist entstanden.

### Der Name Rumex

Rumex ist die lateinische Bezeichnung für den Ampfer und dieser taucht in der Natur dann auf, wenn der Boden Überdüngt, Verdichtung und Beschädigt ist. Rumex gehört zu den sogenannten Pionier Pflanzen. Er ist ein Lückenfüller.

Genau das soll Rumex auch sein ein Lückenfüller für alle die diese Systeme satt haben.

# Kurzbeschreibung

# Die Index Seite

# Seiten Arten

Rumex drei Arten von Seiten ein. Diese Verwendung wird über die Dateiendung gesteuert. Eine Zahl in dieser Dateiendung bestimmt außerdem ob ein Inhaltsverzeichnis erstellt werden soll und bis welcher Tiefe es angezeigt wird. Der Aufbau der Datei Endung ist dabei folgender

- Die ersten beiden Zeichen `rx` kennzeichnen eine Rumex Datei
- Das dritte Zeichen, immer eine Zahl, kennzeichnet ob die Datei ein Inhaltsverzeichnis besitzt und bis welcher Ebene dieses eingebunden werden soll.
- Das vierte Zeichen zeigt die Verwendung, sprich Einbindung an.

**Bedeutung der Zahl in der Dateiendung**

0 Datei ohne Inhaltsverzeichnis.

1 Datei mit Inhaltsverzeichnis bis zur ersten Ordnung.

2 Datei mit Inhaltsverzeichnis bis zur zweiten Ordnung.

3 Datei mit Inhaltsverzeichnis bis zur dritten Ordnung.

4 Datei mit Inhaltsverzeichnis bis zur vierten Ordnung.

5 Datei mit Inhaltsverzeichnis bis zur fünften Ordnung.

6 Datei mit Inhaltsverzeichnis bis zur sechsten Ordnung.

### Verwendungskennzeichnung

### rx0w

### rx0x

## Sonderseiten

### index

### suche

### start

### weblog

### sitemap

### rss

## Installation

Rumex kann in zwei Stufen verwendet werden. Einmal als lokale, private Installation[1] und als dann noch als öffentliche Installation.

### Vorbereitung des Rechners

Rumex ist auf ein *nix System ausgerichtet. Auf diesem sollten folgende Programme installiert sein:

- bash
- make
- perl
- git
- pandoc
- imagemagick
- wget
- sitecopy

[1] Rumex braucht generell keine online Verbindung. Alle benötigten öffentlichen Scripte werden statisch gespeichert, sind in der Installation enthalten. Nicht einmal ein Webserver muss installiert werden.

- vim (g)vim
- texlive

Wer mit dem Editor `vim` zurecht kommt sollte sich auch `gvim` installieren. Rumex besitzt eine `gvim` Erweiterung die, die Arbeit bzw. die Suche nach dem richtigen Befehl am Anfang um einiges erleichtert.

`texlive` wird nur gebraucht wenn man auch PDF Dateien erstellen möchte.

`sitecopy` wird nur gebraucht wenn man die Daten per FTP hoch laden möchte.

Bei Debian dürfte dies die nachfolgender Zeile erledigen.

```
sudo apt-get install make perl git-core \
pandoc imagemagick sitecopy gvim texlive
```

## Die lokale Installation

Für die Installation auf deinem Rechner musst zu erst das ZIP bzw. das tar.gz Archiv vom github Server holen und entpacken werden.

```
wget https://github.com/itbayer/rumex/archive/gh-pages.zip
unzip gh-pages.zip
```

bzw.

```
wget https://github.com/itbayer/rumex/archive/gh-pages.tar.gz
tar -xzvf gh-pages.tar.gz
```

Nach dem entpacken wechselt man in das Verzeichnis `rumex-gh-pages/.rx` und starte die Befehle `make install` und `make show`.

```
cd rumex-gh-pages/.rx/
make install
make show
```

Rumex ist jetzt lokal installiert und zeigt die erste Seite an. Auf dieser findet man eine kurz Information der Schritte die man noch machen sollte. Diese Schritte sind vor allen wichtig wenn man seine Seiten veröffentlichen möchte.

## Veröffentlichen der Seite

In der zweiten Stufen kann die Seite auf drei Arten veröffentlicht werden.

1. Hosting auf Github

2. Hosting auf einem Server mit git Unterstützung

3. Hosting klassisch per FTP

#### Veröffentlichung auf github

TODO hier fehlt noch was

#### Veröffentlichung auf einem Server mit git Unterstützung

#### Veröffentlichung auf einem Server mit FTP

# Konfiguration

## Rumex auf dich einstellen

Nach der Installation muss Rumex noch auf dich eingestellt werden. Genauer gesagt sollten folgende Angabe für deine neuen Seite angepasst werden.

- Url deiner Seite
- Impressum
- Kopf / Fusszeile
- Logo

Eine Kurzbeschreibung dieser Anpassungen findest du, nach der Installation, auf der ersten Seite[2] die dir Rumex zeigt.

## Rumex Konfigurationsvariable

In diesem Abschnitt werden alle Varablen die Rumex verwendet beschrieben. Diese Variablen befinden sich in zwei Dateien. Zu einem ist das die Vorgabe Datei `.rumex/makefile/config.mk` und die Benutzerdatei `.rx/config.mk`, wobei die Einträge der Vorgabe Datei von den Einträgen der Benutzer Datei ,soweit vorhanden, überschrieben werden.

Es sind nicht gewünschten Einträge in der Benutzerdatei vorhanden, braucht man die entsprechende Variable nur in die Benutzerdatei einzutragen.

**URL** Definiert die URL über die die Seite erreichbar ist. Die URL muss hier ohne abschließendes `/` eingegeben werden.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

[2] Diese Seite, die bei der Installation kopiert wird trägt den Namen `istart.rx0v`.

Beispiel:

```
URL="http://www.it-bayer.de/rumex"
```

---

**INDEX__TITEL** Definiert den Titel der Seite. Es können auch HTML Tags verwendet werden.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel: (eine Zeile)

```
INDEX_TITEL = "IT Bayer's rumex <sup style=\"font-size: .4em;\">
(github.com Version)</sup>"
```

---

**INDEX__AUTOR** Definiert den Autor der Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
INDEX_AUTOR = "Stefan Blechschmidt"!
```

---

**INDEX__DATUM** Definiert den Stand, Datum der Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
INDEX_DATUM = $(shell ls index.rx0x -l --time-style=+%Y-%m-%d | awk '{print $$6}')
```

---

**RSS__TITEL** Definiert den Titel der RSS Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
RSS_TITEL = "Neuigkeiten von rumex Baukasten"
```

---

**CSSALL** Definiert die CSS Datei für die Formatierung aller Seiten.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
CSSALL = "rxtpl/css/all.css"
```

---

**CSSSCREEN** Definiert die CSS Datei für die Formatierung der Bildschirmausgabe.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
CSSSCREEN = "rxtpl/css/screen.css"
```

---

**CSSPRINT** Definiert die CSS Datei für die Formatierung der Druckausgabe.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
CSSPRINT = "rxtpl/css/print.css"
```

---

**CSSHANDHELD** Definiert die CSS Datei für die Formatierung Mobiler Endgeräte.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
CSSHANDHELD = "rxtpl/css/handheld.css"
```

---

**SEITENBANNER** Definiert die CSS Datei für das Seitenbanner.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
SEITENBANNER = "rxtpl/img/rumex.png"
```

---

**MOOTIT** Kontoname für die Diskusionserweiterung mittels moot.it. Wird diese Variable kommentiert ist diese Erweiterung ausgeschaltet.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
# MOOTIT = "rumex"
```

---

**WEBLOGAUTOR** Diese Variable definert den Autor der Weblog Einträge.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
WEBLOGAUTOR = "IT-Bayer"
```

---

**META_PUBLISHER** Angaben zum Autor der HTML Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
META_PUBLISHER = "IT-Bayer"
```

---

**META_CREATOR** Angaben zum Autor der HTML Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
META_CREATOR = "IT-Bayer (Stefan Blechschmidt)"
```

---

**RUMEXSUCHE** Verzeichnis in dem sich das JavaScript für die Suche befindet.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
RUMEXSUCHE = "rxtpl/js"
```

---

**FAVICON** Definiert das Favicon Bild der Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
FAVICON = "favicon.gif"
```

---

**SITECOPY** Definiert das Programm für die FTP Upload Funktion. Ist diese Variabel kommentiert wird git verwendet.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
#SITECOPY = /usr/bin/sitecopy
```

Die TEMPLATE Dateien definieren den Seitenaufbau. Hierzu werden vier Dateien verwendet.

1. HTML_TEMPLATE
2. KOPF_TEMPLATE
3. HEADER_TEMPLATE
4. FUSS_TEMPLATE

Ist im Verzeichnis `rx/` eine entsprechende Datei vorhanden wird diese verwendet. Ansonst wird die Datei aus dem Verzeichnis `.rumex/default` eingebunden.

**HTML_TEMPLATE** HTML Vorlage Datei für die Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel: (eine Zeile)

```
HTML_TEMPLATE = $(shell if [ -f ../.rx/html.template ];
then echo \"../.rx/html.template\";
else echo \"../.rumex/default/html.template\";
fi)
```

**KOPF_TEMPLATE** Definiert die Kopfdatei der Seite.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel: (eine Zeile)

```
KOPF_TEMPLATE = $(shell if [ -f ../.rx/kopf.html ];
then echo \"../.rx/kopf.html\";
else echo \"../.rumex/default/kopf.html\";
fi)
```

**HEADER_TEMPLATE** Definiert die Datei in der Header Einträge der Seite vorhanden sind.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel: (eine Zeile)

```
HEADER_TEMPLATE = $(shell if [ -f ../.rx/header.html ];
then echo \"../.rx/header.html\";
else echo \"../.rumex/default/header.html\";
fi)
```

---

**FUSS_TEMPLATE** Definiert die Datei der Informationen des Fußbereiches.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel: (eine Zeile)

```
FUSS_TEMPLATE = $(shell if [ -f ../.rx/fuss.html ];
then echo \"../.rx/fuss.html\";
else echo \"../.rumex/default/fuss.html\";
fi)
```

Jede der drei Seitentypen bekommt in HEADER der HTML Datei die `meta_robots` Variable mitgeliefert. Anhand der nachfolgenden drei Konfigurationsvariablen werden diese Parameter für jeden Seitentyp gesetzt.

---

**META_ROBOTS_STANDARD** Meta Angabe für die Standard Seiten `rx?s`.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
META_ROBOTS_STANDARD = "all"
```

---

**META_ROBOTS_VERSTECKT** Meta Angabe für die versteckten Seiten `rx?v`.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
META_ROBOTS_VERSTECKT = "noindex,nofollow,noarchive"
```

---

**META_ROBOTS_WEITERLEITUNG** Meta Angabe für die Seiten die weiterleiten `rx?w`.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
META_ROBOTS_WEITERLEITUNG = "noindex,follow,noarchive"
```

---

**INDEX** Programm zum erstellen der `index.rx0x` Datei.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
INDEX="../.rumex/bin/index.pl"
```

---

**RX2RSS** Programm zum Erstellen der RSS Feed Datei.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
RX2RSS = "../.rumex/bin/rx2rss.pl"
```

---

**RSS_FILE** Name der RSS Datei die in den Quelltext eingebaut wird.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
RSS_FILE = "rss.xml"
```

**RSS_EXTERN** Verwenden einer externen RSS Seite. Ist diese Seite gesetzt wird die RSS Seite der angegebenen URL verwendet.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | ja |

Beispiel:

```
#RSS_EXTERN = "http://www.it-bayer.de/rss.xml"
```

---

**SITEMAP_XML** Programm zum erstellen der XML Sitemap Datei

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
SITEMAP_XML = "../.rumex/bin/sitemap_xml.pl"
```

---

**SITEMAP_XML_FILE** Zieldatei der Sitemap Datei.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
SITEMAP_XML_FILE = "../sitemap.xml"
```

---

**SITEMAP_RX** Programm zum erstellen der rx0v Sitemap Datei.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
SITEMAP_RX = "../.rumex/bin/sitemap_rx.pl"
```

---

**SITEMAP_RX_FILE** Zieldatei der HTML Sitemap Datei.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
SITEMAP_RX_FILE = "sitemap.rx0v"
```

---

**BVZ** Verzeichnis in dem sich die Bilder befinden.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
BVZ="../bilder"
```

---

**PANDOC** Pandoc Programm.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

---

**SUCHE** Programm zum Erstellen der `rumexsuche_config.js`.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
SUCHE = "../.rumex/bin/suche.pl"
```

---

**SUCHE_JS_CONFIG** Ziel Datei für die JavaScript Suche.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
SUCHE_JS_CONFIG="../$(RUMEXSUCHE)/rumexsuche_config.js"
```

---

**META_GENERATOR** Rumex Versionshinweis für die HTML Dateien.

| Datei | Vorgabe | Benutzer |
|---|---|---|
| vorhanden | ja | nein |

Beispiel:

```
META_GENERATOR = "rumex "$(shell cat ../.rumex/default/version.txt)
```

# Diskussion

## moot.it

TODO hier fehlt noch die Beschreibung

- ein/ausschalten
- Kontoname einstellen

# Weblog

# RSS Feed

# Vim Kurztasten

Alle Rumex Kurztasten, für den Editor VIM, beginnen mit einem `,r`[3]. Für die Bedienung von Rumex braucht mein nur ein paar. Viele der Kurztasten beinhalten eine Kombination einzelner Kurztasten bzw. Befehle.

Am Anfang ist es sicher einfacher GVIM zu verwenden, da hier ein Menü eingebaut ist welches unter anderem auch die Kurztasten anzeigt.

Wer jedoch mit der tilda Unterstützung arbeitet sollte sich schon ein paar Kurztasten einprägen.

[3] Ausnahmen bestätigen die Regel. So wurde für die zwei spaltige Darstellung `,spn2` verwendet.

## Textbausteine

Mittels dieser Kurztasten können Textbausteine eingebunden werden.

### `,rnd` (RumexNeueDatei)

Erstellt ein neues Datei Gerüst. Dabei wird der Dateiname gleich mit eingebunden.

**So schaut's aus**

```
% vim-kurztasten
%
%


<!--

 [vim-kurztasten](vim-kurztasten.html)
 ======================================================

Vortext INDEX

-->

Vortext INDEX und SEITE

<!-- schnipp -->

TEXT DER SEITE

<!-- vim: set ft=pandoc: -->
```

### `,rnn` (RumexNeueNachricht)

Erstellt einen neuen News Eintrag. Diese Kurztaste macht eigentlich nur in der Datei `rss.rx0x` Sinn. Das Datum hinter dem Titel, Überschrift 1, wird von Rumex gesetzt, ist also immer die aktuelle Systemzeit beim ausführen von `,rnn`.

```
 # Neue Nachricht <!-- 2013/11/10 00:30 -->

<!--
!> Link: http://www.it-bayer.de/rumex/
!> Autor: IT-Bayer
```

```
!> Kategorie: Neues
-->

Ab hier geht die neuen Nachricht los.
```

### `,rwl` (RumexWeiterLeitung)

Erstellt einen neue Weiterleitungsseite.

**So schaut's aus**

```
% Weiterleitung nach ....html
%
%


<script language="javascript">
<!--
//window.location.href="....html";
// -->
</script>
```

Anmerkung: Die Zeile `window.location.href=....html;` wurde hier Kommentiert da sonst die javascript Weiterleitung greift. Normalerweise findet man kein `//` vor der Zeile `window.location.href=....html`.

### `,rmk` (RumexMootitKommentar)

Erstellt einen Moot.it Kommentar Abschnitt. Als Kennzeichnung wird der Dateiname ohne `rx??` eingebaut / angehängt.

```
 # Kommentare

<a class="moot" href="https://moot.it/i/rumex/blog/vim-kurztasten"></a>
```

### `,rnb` (RumexNeuenweBlog)

Erstelle einen neuen Weblog Eintrag.

```
 # Rumex WebLog

_am 09.09.2013 um 15:21 schrieb IT-Bayer_
```

```
Text für den Eintrag

<div class="weblog">
Text der vorerst ausgeblendet ist.
</div>
```

## Git Kommandos

### `,rgp` (RumexGitPull)

Git pull

### `,rgs` (RumexGitStatus)

Git status

## Text Formatierung

### `,rff` (RumexFormatFett)

Markierte Textstelle fett darstellen

### `,rfk` (RumexFormatKursiv)

Markierte Textstelle kursiv darstellen

### `,rfl` (RumexFormatListe)

Markierte Zeilen in eine Liste wandeln

### `,rfn` (RumexFormatNummernliste)

Markierte Zeile in eine Nummernliste wandeln

### `,rfb` (RumexFormatBlock)

Markierte Zeile in einen Block wandeln

**,rfc** Markierte Zeile als Code darstellen

**,spn2** Text mit 2er Spalten Formatierung umschließen

**,spn3** Text mit 3er Spalten Formatierung umschließen

**,spn4** Text mit 4er Spalten Formatierung umschließen

## Make Befehle

### `,rma` (RumexMakeAll)

Speichert alle offenen Dokumente und erstellt daraus die HTML Datei.

### `,rmb` (RumexMakeBild)

Erstellt die unterschiedlichen Bildgrößen

### `,rmca` (RumexMakeCleanAll)

Löscht alle html Dateien, alle Bildergrößen und alle xml Dateien. beinhaltet die drei Befehle `make bclean`, `make hclean`, `make xclean`.

### `,rmcb` (RumexMakeCleanBilder)

Löscht alle Bildgrößen aus dem `bilder/` Verzeichnis. Es werden nur die Bilder gelöscht die von Rumex erstellt wurden siehe `make bilder`.

### `,rmch` (RumexMakeCleanHtml)

Löscht alle `.html` Dateien die von Rumex erstellt wurden.

### `,rmcx` (RumexMakeCleanXml)

Löscht alle `.xlm` Dateien die von Rumex erstellt wurden.

### `,rmcf5` (RumexMakeCleanF8htm)

Löscht alle `.htm` Dateien die mittels der Gvim Taste F5, siehe HTML und PDF Dateine mit pandoc und gvim erstellen, erstellt wurden.

### `,rmh` (RumexMakeHtml)

Erstellt die `.html` Dateien.

#### `,rmi` (RumexMakeIndex)

Erstellt die index Datei.

#### `,rmm` (RumexMakesiteMap)

Erstellt die sitemap Datei.

#### `,rmo` (RumexMakeOnline)

Speichert alle Dateien und stellt diese online.

#### `,rmr` (RumexMakeRss)

Erstellt die rss Datei.

#### `,rms` (RumexMakeSuche)

Erstellt die Rumex Suche, bzw. die Dateiliste die für die Suche verwendet werden soll.

### Vorschau

#### `,rsf` (RumexShowFile)

Zeigt eine Vorschau der Seite der Datei `file:///` im Standardbrowser.

#### `,rsw` (RumexShowWww)

Zeigt eine Vorschau der Seite **Online** im Standardbrowser.

#### `,rsl` (RumexShowLocal)

Zeigt eine Vorschau der Seite auf dem eigenen Rechner. Diese Kurztaste kann aber nur verwendet werden wenn auf dem Rechner der Apache installiert und entsprechend eingerichtet ist.

### Sonstiges

#### `,rku`

Öffnet das Unterverzeichnis `.rx` in einem neuen VimTab Fenster. Keine Ahnung warum ich diese Kurztaste so benannt habe.

#### `,ros` (RumexOpenStart)

Öffnet die `start.rx0s` in einen neuen VIM Tab Fenster.

#### `,ror` (RumexOpenRss)

Öffnet die `rss.rx0x` in einen neuen VIM Tab Fenster.

## Statik Funktion

Rumex verwendet pandocs markdown weil man damit sehr einfach und schnell Text erstellen und in verschiedenen Formate wandeln kann. Für die Erstellung von Denkschriften[4] wurde zusätzlich eine, ich nenne sie *statik Funktion* eingebaut. Mit dieser Funktion ist es möglich innerhalb eines Unterverzeichnisses verschiedenen Ausgabe Formate zu erstellen. Zur Zeit werden, von Rumex, folgenden Formate unterstützt:

- .html
- .pdf
- .odt OpenDokument
- .epub E-Book
- .mobi E-Book (Kindle)

Erstellt werden die einzelnen Formate über die innerhalb (g)vim[5] zur Verfügung stehen.~~Eine Besonderheit ist dass die `.htm` Datei auch ohne die zusätzlichen Dateien wie Bilder oder die CSS Datei funktionieren. Alle Daten werden in die `.htm` Datei eingebunden.~~[6]

[4] Im Neudeutschen würde man die Denkschrift als Memorandum bezeichnen.

[5] Die Funktionstasten sind im gvim Rumex Menü nicht eingebaut. Man sollte sie sich also merken :-).

[6] Diese Eigenschaft wurde wieder entfernt da eine Datei mit sehr vielen Bildern oder gar Videos, die alle in die HTML Datei eingebunden werden, sehr groß wird. Dieses erhöht die Ladezeit der HTM Datei. Bei der Verwaltung durch git wirkt sich diese Eigenschaft auch ungünstig aus.

Auch wurde die von Pandoc eingebaut sodass Verweise auf anderen Quellen in den Denkschriften verwendet werden können.

Wer sich Rumex nicht installieren möchte aber dennoch eine einfache Möglichkeit sucht HTML Seiten zu erstellen, kann sich einmal cirsium anschauen. Cirsium ist eine Auskopplung aus Rumex, unterteilt aber die Quelltextdatei in verschiedene Abschnitte. Am besten einfach mal auf github.com anschauen.

Ab der Rumex Version 0.8.2 sind die Funktionstasten in Rumex enthalten. Folgende F-Tasten wurden belegt.

**F5** Erstellt die `.htm` Datei ohne Inhaltsverzeichnis.

**Shift+F5** Erstellt die `.htm` Datei mit Inhaltsverzeichnis bis zur Ebene 3.

**Alt+F5** Erstellt die `.htm` Datei mit kompletten Inhaltsverzeichnis bzw. bis Ebene 5 da LaTeX auch nicht mehr Ebenen unterstützt und die beiden Ausgabe Format annähernd identisch sein sollten.

**Ctrl+F5** Öffnet die `.htm` Datei.

**F6** Erstellt die `.pdf` Datei ohne Inhaltsverzeichnis.

**Shift+F6** Erstellt die `.pdf` Datei mit Inhaltsverzeichnis bis zur Ebene 3.

**Alt+F6** Erstellt die `.pdf` Datei mit kompletten Inhaltsverzeichnis. Hier ist anzumerken dass LaTeX nur Inhaltsverzeichnisse bis zur Ebene 5 unterstützt. Das HTML Format wurde entsprechend angepasst, siehe oben.

**Ctrl+F6** Öffnet die `.pdf` Datei. Zur Zeit wird nur zathura unterstützt.

**F7** Erstellt die restlichen Formate, `.epub`, `.odt` und `.mobi`. Voraussetzung für das `.mobi` Format ist `calibre`.

**Ctrl-F7** Öffnet die Literatur Verwaltung `rumex.bib`. Voraussetzung, `jabref` ist installiert.

## HTML Formatierung

Die erzeugte HTML Datei besitzt Standardmäßig keine Formatierung bzw. verwendet die Standard Darstellung des Browsers.

Kopf- und Fusszeile werden dadurch nicht, vom restlichen Text, unterschieden. Auch das Inhaltsverzeichnis ist im ersten Moment als solches nicht gleich zu erkennen. Dieses kann mit ein wenig CSS geändert werden. Diese CSS Datei ist ab Rumex Version 0.8.2 enthalten muss aber unter Umständen noch eingerichtet werden.

```
cd .rx
ln -s ../.rumex/default/statik.css statik.css
```

Beim Lesen des Artikels *"PDF-Dokumente schreiben mit Pandoc und Markdown" [ˆstenderprolinux] ist mir die Idee gekommen die Rumex* statik Funktion* mit einem Literaturverzeichnis, die ja auch in pandoc zur Verfügung steht, zu versehen.

## Installation pandoc manuell

Für die Verwendung der Literaturfunktion muss pandoc um das Zusatzprogramm `pandoc-citeproc` erweitert werden. Wer Pandoc über die Paketverwaltung installiert hat braucht hier nichts zu machen. Wer Pandoc manuell, so wie ich, installiert hat muss dieses Programm nachinstallieren.

Dazu erweitert man die Installationszeile um das neue Programm

```
cabal update
cabal install pandoc pandoc-citeproc
```

Zu guter Letzt erstellt man noch die symbolischen Links der beiden Programme.

```
sudo ln -s /home/USER/.cabal/bin/pandoc /usr/local/bin/.
sudo ln -s /home/USER/.cabal/bin/pandoc-citeproc /usr/local/bin/.
```

**Nachinstallation Rumex** Wer Rumex schon im Einsatz hat muss für die Erweiterung ein wenig Hand anlegen. Zu erste holt man sich die neue Version[7] von rumex.

Dann braucht man noch drei zusätzliche Dateien im Verzeichnis `.rx`.

- `rumex.bib`
- `rumex.csl`
- `statik.css`

Wobei der und die CSS Datei nur verlinkt wird. In der `rumex.bib` werden dann die Literatur Verweise verwaltet.

```
cd .rx
touch rumex.bib
ln -s ../.rumex/default/din-1505-2.csl rumex.csl
ln -s ../.rumex/default/statik.css statik.css
```

Als Literatur Stil kommt `din-1505-2.csl` zum Einsatz. Andere Stile findet man im git Repository https://github.com/citation-style-language/styles.git. Als Name für die Stil Vorlage wurde `rumex.csl` gewählt damit mit eine Änderung des Stils einfach über den Symlink gemacht werden kann.

[7] Die Literatur Erweiterung ist ab der Rumex Version 0.8.2 enthalten.

**Literatur Verwaltung** Für die Verwaltung der Literatur Datenbank verwende ich Jabref.

```
sudo apt-get install jabref
```

Der Aufruf des Programms wurde auch auf einen F Taste gelegt. Wer eine anderes Programm verwenden will muss diesen entsprechend anpassen.

### Tipps

- Auf Wikipedia (2013) findet man eine schöne Beschreibung über die Literatur Verwaltung mit BibTex.
- Die BibTex Einträge muss man sich unter Umständen nicht einmal selber erstellen. Da verschiedene Seiten entsprechende Dienste anbieten. Gelungen finde ich die Seite von http://www.literatur-generator.de/ aber auch auf http://lead.to/amazon/en/?op=bt findet man BibTeX Einträge. Zwar muss man diese unter Umständen noch ein wenig überarbeiten aber das Grund Gerüst wird einem sozusagen frei Haus geliefert.
  Wer einen Wikipedia Artikel zitieren dem wird unter `Werkzeuge -> Seite zitieren` weiter geholfen.
- Für das zitieren von Internetseiten verwende ich folgende Formate, siehe dazu Abschnitt .

```
@ELECTRONIC{ wiki:bibtex,
    author = "Wikipedia",
    title = "BibTeX --- Wikipedia{,} Die freie Enzyklopädie",
    year = "2013",
    url = "http://de.wikipedia.org/w/index.php?title=BibTeX&oldid=124228120",
    note = "[Online; Stand 18. Dezember 2013]"
}
```

. . . oder

**Achtung:** Das Formate `@WWW` wird von jabref nicht unterstützt und gegen ein anders ausgetauscht. Bei dem Einsatz von Jabref am besten `@ELECTRONIC` verwenden.

```
@WWW{ wiki:bibtex,
    author = "Wikipedia",
    title = "BibTeX --- Wikipedia{,} Die freie Enzyklopädie",
    year = "2013",
    url = "http://de.wikipedia.org/w/index.php?title=BibTeX&oldid=124228120",
    note = "[Online; Stand 18. Dezember 2013]"
}
```

## Verwendung in- und außerhalb von Rumex?

Innerhalb von Rumex erstellt man in einem separatem Unterverzeichnis die entsprechende markdown Datei und dann kann es auch schon los gehen.

Außerhalb von Rumex kann man diese Funktion natürlich auch verwenden. Mit Außerhalb meine ich Denkschriften die nicht veröffentlicht werden. Dazu gibt es zwei Möglichkeiten.

1. Die Datei bzw. das Verzeichnis in `.gitignore` hinterlegen. Somit wird diese nicht verwaltet und auch nicht, bei einem `make online`, hoch geladen.
2. Eine zweite lokale Rumex Installation die nur für Denkschriften verwendet wird.
3. ... und dann wäre da noch Cirsium, eine Auskopplung aus Rumex, mit der man einfache html und pdf Seiten erstellen kann. Die Literaturverzeichnis-option ist auch enthalten. Die Formate odt, epub und mobi sind jedoch nicht eingebaut.

### Setzen der Überschrift für das Literaturverzeichnis

Die Überschrift für das Literaturverzeichnis muss immer am Ende des Artikels gesetzt werden.

**Beispiel**

```
# Literaturverzeichnis
```

### Einbinden von Bildern

Bei dem Einbinden der Bilder muss man beachten dass die Erstellung der statik Datei vom Verzeichnis `.rx` ausgeht.

Will man also ein Bild, dass im Ordner der Statik Datei liegt einbinden so muss auch auf das Bild aus der Sicht des `.rx` Verzeichnisses eingebunden werden.

Beispiel:

Das Bild liegt im Ordner `statik` somit müsste der Markdown Befehl so aussehen.

```
![Beispielbild](../statik/beispiel.png)
```

In Rumex kann man diese Funktion natürlich auch verwenden. Am besten erstellt man sich dazu ein eigenes Unterverzeichnis und dort die Datei `index.md` mit den Texten.

### Statik Dateien im `.rx` Verzeichnis

Es wird sicher passieren dass man die Funktionstasten der Statik Seiten auch bei der Bearbeitung der eigentlichen Rumex Dateien drückt. Durch entsprechende Einträge in der `.gitignore` Datei werden solche Dateien von einem Upload ausgeschlossen. Mit den Aufruf von `make statikclean` können die erstellten statik Dateien Schlussendliche aus dem `.rx` Verzeichnis entfernt werden. Dieser Befehl wird auch bei `make clean` ausgeführt.

### Tipps

Das PDF Programm `zathura` hat die Eigenschaft dass wenn sich die Datei ändert diese automatisch nachgeladen wird. Eine schöne Funktion wenn man seinen Text, an dem man gerade arbeitet, immer wieder einmal im Ausgabe Format betrachten will. Einfach die Taste F6 drücken, die Datei wird auch gleich gespeichert, und mit ALT-TAB das Programm Fenster wechseln.

Bei Format HTML geht das natürlich auch. Nur muss hier eine Erweiterung installiert werden. Für die Browser Chromium und Firefox habe ich mit `Auto Refresh Plus`$^{Chromium}$ bzw. `Tab Auto Reload`$^{FireFox}$ gute Erfahrungen gemacht.

---

Die PDF Datei dieser Beschreibung kann man sich hier ansehen. Die Markdown Quelldatei kann man sich hier holen.

---

## Lager

Zwischenlager der Texte die noch nicht aufgearbeitet wurden.

### gvim unterstützung einrichten

Blechschmidt (2013a): hier fehlt noch die beschreibung@

- erstellen der startdatei
- einrichten

### tipps zur benutzung gvim

**vim kurztasten** eine übersicht der rumex kurztasten für den editor vim findest du auf der seite vim-kurztasten.

**gvim menü** eine übersicht des rumex menüs für den editor gvim findest du auf der seite gvim-menü.

**dateiname ergänzen** will man in die datei einen dateinamen einbauen, weiß aber nicht mehr genau wie er heißt, kann man folgenden trick verwenden. in diesem beispiel wird ein bildname gesucht. im text schreibe man `../bilder/tw` und drückt dann die tastenkombination `c-x + c-f`, gvim öffnet ein dialogfeld in dem alle dateien die auf dieses muster übereinstimmen geöffnet. gibt es nur einen treffer wird dieser gleich eingefügt.

**Wort innerhalb des Dokumentes suchen** Sucht man ein Wort das man im Dokument schon einmal verwendet hat, um zum Beispiel darauf zu verweisen. Schreibt man den Wortanfang und drückt dann `c-p`. Es öffnet sich ein Dialogfeld in dem alle Wörter die auf dieses Muster passen angezeigt werden. Gibt es nur einen Treffer wird dieser gleich eingefügt. Diese Wortsuche funktioniert auch Dateiübergreifend. Die Datei muss jedoch geöffnet sein.

**Nützliche Erweiterungen** Vim bietet ein paar nützliche Erweiterungen in Form von Plugins an. Hier eine Liste, der Plugins, die ich gerne verwende.

**pathogen.vim** diese Erweiterung macht das Installieren weitere Erweiterungen einfach. Dabei ist die Installation von `pathogen` schnell erledigt.

**vim-pandoc** Erweiterung rund um Pandoc.

**fuzzyfinder** Dateien schnell zum editieren öffnen. Installiert ist diese Erweiterung, vorausgesetzt man verwendet pathogen, mit dem Befehl:

```
wget -o /tmp/vim-fuzzyfinder.zip http://www.vim.org/scripts/download_script.php?src_id=
mkdir ~/.vim/bundle/vim-fuzzyfinder
unzip /tmp/vim-fuzzyfinder.zip -d ~/.vim/bundle/vim-fuzzyfinder/
```

für das öffnen dies datei dialogs sollte man sich dann noch eine kurztaste konfigurieren.

```
" ------------------------------------------------------------
" fuzzyfinder file suche auf <f12> binden
map <f12> :fuzzyfinderfile <cr>
```

### seite auf einem github.com server einrichten

Blechschmidt (2013a)

- einrichten eines github.com zugangs
- arbeits- repository auf den ap[8] holen
- die rumex zip datei
- die dateien des zip archives in das arbeits- repository kopieren
- grund dateien anpassen
  - Blechschmidt (2013b): angabe welche dateien fehlt noch@
- erste änderungen vornehme
- `make online` - fertig.

### seite auf einem nicht github.com server einrichten

**datei upload per git** Blechschmidt (2013a)

Auf der Seite rumex.it-bayer.de findet man eine Beschreibung wie man den rumex Baukasten auf einen **nicht** `github.com` Server installiert.

**datei upload per ftp** Blechschmidt (2013a)

### root verzeichnis

Im `root` Verzeichnis findet man alle `html` Dateien der Seite. Diese werde vom Baukasten erstellt und müssen nicht von Hand verändert werden. Zusätzlich findet man noch ein folgende Systemdateien:

**rss.xml** news feed datei, wird vom system erstelle

**readme.md** beschreibungsdatei die von github.com gebraucht wird

**robots.txt** datei für die suchmaschinen

**favicon.ico** icon für den browser

**.htaccess** Konfiguration für den Apache Server

**cname** Blechschmidt (2013b): beschreibung@

[8] mit ap ist der **a**rbeits**p**latz rechner gemeint.

### Unterverzeichnisse

**.rumex/** Blechschmidt (2013b): beschreibung@

**.rx/** Blechschmidt (2013b): beschreibung@

**rxtpl/** Standard Template Verzeichnis. In diesem Verzeichnis befinden sich die Dateien die für das aussehen der Seite verantwortlich sind.

Folgende Dateinen und Verzeichnisse sind hier zu finden

- index.html (Weiterleitung zum root Verzeichnis)
- css/
- js/
- img/

**bilder/** in diesem Verzeichnis werden alle Bilder der Seite abgelegt.

### Steuerung durch Dateiendung

Das aussehen der Dateien bezüglich des Inhaltsverzeichnisses könnte auch durch die Dateiendung gesteuert werden.

### Beispiel

**datei.rx0s** würde eine html Datei ohne Inhaltsverzeichnis erstellen.

**datei.rx1s** erstellt die html Datei mit dem Inhaltsverzeichnis aus den Einträgen der h1 Überschriften.

**datei.rx2s** erstellt eine html Datei mit dem Inhaltsverzeichnis aus den Einträgen der h1 und h2 Überschriften.

Zusätzlich könnte die Dateiendung auch eine unterschiedliche Verwendung der Dateien ermöglichen.

**datei.rx0s, datei.rx1s ...** Standard Datei.

**datei.rx0x, datei.rx1x ...** Datei die nicht in die Liste auf `index.html` eingebunden wird.

**datei.rx0v, datie.rx1v ...** Versteckte Datei. Diese taucht weder in der `index.html` noch in der `sitemap.xml` auf.

**datei.rx0w** Datei mit einer Weiterleitung. Die Weiterleitung wird dabei mittels `javascript` realisiert, da github keine `.htaccess` Weiterleitung unterstützt.

```
% weiterleitung nach beschreibung.html
%
%

<script language="javascript">
<!--
window.location.href="beschreibung.html";
// -->
</script>
```

Bei der Änderung der Dateiendung bleibt der eigentliche `html` Name gleich. Nur die Funktion der Einbindung ändert sich.

### sonderseiten

Im Verzeichnis `pandoc` befinden sich Sonderseiten.

**rumex/index.rx0x** Diese Datei wird vom Programm `.bin/make_index.pl` erstellt, muss somit nicht vorhanden sein.

**rumex/start.rx0s** Die `start.rx0s` wird als Vortext in die `index.rx0x` eingebunden. Mit ihr kann man oberhalb der Seiten Liste einen extra Text in die `index.html` eingebunden werden.
**diese Datei ist erforderlich und muss vorhanden sein.**

**rumex/rss.rx0x** Datei für die RSS FEED Funktion.

**rumex/impressum.rx0x** Datei für die Impressumsangaben.
**ist nicht zwingend erforderlich. Jedoch muss die `.inc/fuss.html` entsprechende bearbeitet, der link muss raus genommen, werden.**

**rumex/makefile** sie `make` Steuerdatei.

### Aufbau der Startseite

Die Startseite `markdown/start.rx0s` muss vorhanden sein. Es reicht auch ein ‘touch markdown/start.rx0s.

Der normale Aufbau könnte so ausschauen. Die `pandoc` kopfzeile sind nicht zwingend erforderlich.

```
% start.rx0s
%
%

hier kommt auch schon der vortext für die index.html
```

### aufbau der einzelseiten

Die Einzelseiten liegen alle im Verzeichnis `rumex` und zwar in der Sprache `markdown` bzw. der Erweiterung von `pandoc`.

Diese Einzelseiten werden in chronologischer Reihenfolge in die Startseite `index.html` eingebunden und bilden sozusagen das Inhaltsverzeichnis der Seite. In jeder Einzelseite wird dazu ein sogenannter *"vortext"* hinterlegt. Die Seite bzw. der Kopf der Seite hat dabei folgenden Aufbau.

```
% seiten überschrift 1
% seiten überschrift 2
% seiten überschrift 3

<!--


# listen-überschrift

überschrift und text der in der listenübersicht
auf index.html angezeigt wird.

alles was innerhalb der html kommentar marken
steht wird nur auf der index.html seite angezeigt.

-->

alles was sich außerhalb der html marken
befindet wird auch auf der eigentlichen seite angezeigt.

durch das schlüsselwort "schnipp", das auch in html
kommentar marken stehen muss, wird der vortext beendet.
auf der index.html erscheint an dieser stelle der link
"... weiter lesen".

<!-- schnipp -->

ab hier geht dann der inhlat der eigentlichen seite los.
```

#### template

Blechschmidt (2013a)

Änderung gegenüber des original `pandoc` templates.

## rss feed funktion

Blechschmidt (2013a)

Wird nicht aus den einzelnen dateien erstellt sondern muss manuell editiert werden, datei `.rx/rss.rx0x`.

Jede Überschrift eines Eintrags muss mit einem `{.nn1}` enden.

Danach kommen die angaben zu:

**link:** *verweis zur seite mit weiteren informationen*
**autor:** *autor der den eintrag geschrieben hat*
**kategorie:** *kategoie des eintrags*
**datum:** *datum des eintrags.* das richtige format bekommt man mit dem Befehl `date -r`.

die in html kommentar marker eingeschlossen sind.

anschließend folgt die meldung. zur zeit werden folgende `pandoc` formatierungen unterstützt.

- überschriften ab der stufe 3 `###`
- aufzählungen `-`
- aufzählungen `*`
- zitate `>`
- links `[link](http://muster.tdl)`. diese dürfen nicht am anfang einer zeile stehen.
- bilder `![bild](../bilder/muster.png muster.png)`, diese dürfen nicht am anfang einer zeile stehen.
- code ‘code‘.

#### beispiel einer rss feed seite

#### rss dateiname

der dateiname ist mit `rss.xml` vorbelegt und kann über die variable `rss_file` in der `config.md` geändert werden.

**rss titel**

der title des rss feed wird durch die variable `rss_titel` angepasst.

```
rss_titel = "neuigkeiten von rumex baukasten"
```

**rss auslagern**

den rss link kann man auch auslagern so dass dieser auf eine andere seite zeigt. dazu setzt man die variable `rss_extern` mit dem entsprechenden link. die variabel `rss_file` wird dadurch nicht mehr verwendet. auch der rss lauf wird dadurch ausgeschaltet und durch eine meldung ersetzt.

**rss kurztaste**

für die einzelnen einträge steht auch eine kurztaste `.rnn` zur verfügung.
in gvim unter

```
rumex -> textbausteine -> neuernews eintrag
```

eingefügt wird dann folgende vorgabe. der wert hinter datum wird von system ausgelesen und entsprechende gesetzt.

```
 # neue nachricht{.nn1}

<!--
| link: http://www.it-bayer.de/rumex/
| autor: it-bayer
| kategorie: neues
| datum: mon, 28 oct 2013 07:36:56 +0100
-->

ab hier geht die neue nachricht los.
```

**interna**

durch die beschriebenen rss variablen wird die erstellung des rss feed gesteuert. es wird in jede html datei nachfolgender header abschnitt eingebaut wenn die `rss_title` variable gesetzt wurde. `rss_file` bzw. `rss_extern` steuern den `href` eintrag.

## Rumex auf einem USB Stick

Rumex kann auch auf einem USB Stich installiert werden. Der Stick muss aber ein Dateiformat besitzt
welches mit Dateirechten und symbolische links umgehen kann.

**USB Sticks im vfat Format funktionieren nicht.** man kann zwar die Daten darauf ablegen. Das arbeiten über den Stick funktioniert nicht wirklich. Auch wenn man die Daten, von einem vfat Stick, auf ein *nix System kopiert werden muss händisch nach gebessert werden.

Linktipp: USB Stick unter linux verschlüsseln

## make Steuerung

Blechschmidt (2013a)

Gesteuert wird der Baukasten mittels `make` im Unterverzeichnis `rumex`. Folgende `make` Befehle stehen dabei zur Verfügung.

**make html** erstellt die einzelnen html dateien. hier kann auch nur `make` verwendet werden.

**make index** erstellt die `index.md` datei aus der dann die `index.html` datei erstellt wird.

**make all** eine zusammenstellung aus `make index` und `make html`.

**make online** daten auf github hoch laden.

**make bilder** erstellt bilder in verschiedenen auflösungen.

## update

Blechschmidt (2013a)

## Homepage Änderung, schnell und immer aktuell

Wer wünscht es sich nicht einen schnellen Zugriff auf seine Homepage. Sei es um schnell was zu ändern oder genauso schnell was neues einzustellen. Rumex bietet in Verbindung mit `tilda` eine, so finde ich, schöne, einfache und auch, schnelle lösung.

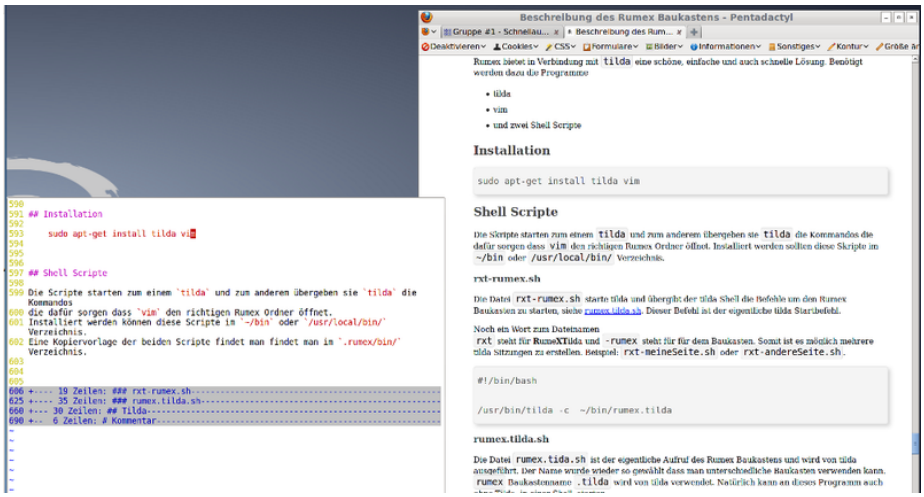


Abbildung 1: rumex im tilda Fenster. Mit einem Tastendruck öffnet sich das tilda Fenster und man kann die Texte eintippen. Ein erneuter Tastendruck schließt das tilda Fenster wieder und der Bildschirm ist wieder frei. Hat man seine Änderung abgeschossen kann man mit den Rumex Kurztasten die Änderung schnell online stellen.

### Installation der tilda Unterstützung

Die Installation ist nicht umfangreich. Man braucht vim und tilda und dann noch zwei bash script. Eine Kopiervorlage der beiden Scripte findet man findet man im `.rumex/bin/` Verzeichnis.

```
sudo apt-get install tilda vim
cp .../rumex/.rumex/bin/rxt-rumex.sh ~/bin/.
cp .../rumex/.rumex/bin/rumex-tilda.sh ~/bin/.
```

Diese beiden Bash Scripte müssen anschließend noch angepasst werden.

**Shell Scripte** Die Scripte starten zum einem `tilda` und zum anderem übergeben sie `tilda` die Kommandos die dafür sorgen `vim` im richtigen rumex Ordner zu öffnet. Installiert werden können diese scripte im `~/bin` oder `/usr/local/bin/` Verzeichnis.

**rumex-tilda.sh** Die Datei `rumex-tilda.sh` starte tilda und übergibt der tilda Shell die Befehle um den rumex Baukasten zu starten, siehe .

```
#!/bin/bash
```

```
/usr/bin/tilda -c  ~/bin/rumex-vim.sh
```

**rumex-vim.sh** Mit dem Befehl `rumex-vim.sh` wird der rumex Baukastens aufgerufen. Dieser Befehl wird unter anderem auch von `rumex-tilda.sh` verwendet. `rumex-vim.sh` kann natürlich auch in einem Shellfenster ausgeführt werden.

**rumex-gvim.sh** Mit dem Befehl `rumex-gvim.sh` wird der rumex Baukasten mit dem Editor gvim gestartet.

#### tilda einrichten

Nach dem ersten Start wird tilda in linken oberen Bildschirm Bereich eingeblendet. Man sollte tilda nun noch an seine Bedürfnissen anpassen. Dazu in das tilda Fenster mit der rechten Maustaste klicken und `eigenschaften` aus wählen.

**Übrigens:** Man kann `tilda` mehrfach starten. Somit kann auf mehreren rumex Installationen parallel über diese weiße zugegriffen werden. Man sollte nur jede `tilda` Sitzung ein wenig anders konfigurieren.

**Nachteil:** Ein Nachteil von `tilda` darf man aber nicht verschweigen. Bei wechseln zwischen den Fenstern kann man die Tastenkombination `<alt>+<tab>` nicht verwenden bzw. man kommt mit dieser Kombination nicht mehr zurück nach `tilda`. Schließt und öffnet man `tilda` mit der definierten Taste bekommt man aber den Fokus wieder in das Fenster.

Will man die `<alt>+<tab>` Kombination doch verwenden muss man die Standardeinstellung von tilda ändern. Den erforderlichen Schalter findet man in der Konfiguration, Reiter *allgemein* -> schalter *nicht in der taskleiste anzeigen.*

## Über dieses Dokument

Dieses Dokument wurde mit der Statik Funktion von Rumex erstellt. Dadurch ist es natürlich möglich diesen Text in verschiedene Formate anzubieten.

## Lizenz

### Namensnennung

Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.

### Keine kommerzielle Nutzung

Dieses Werk bzw. dieser Inhalt darf nicht für kommerzielle Zwecke verwendet werden.

### Weitergabe unter gleichen Bedingungen

Wenn Sie das lizenzierte Werk bzw. den lizenzierten Inhalt bearbeiten oder in anderer Weise erkennbar als Grundlage für eigenes Schaffen verwenden, dürfen Sie die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

Außerdem müssen die Lizenzen der Programme, die zum Erstellen des jeweiligen Dokumentes verwendet werden, berücksichtigt werden.

## Impressum / Kontakt

**Verwendung:** Privat

**Anschrift:** Stefan Blechschmidt
Auggenbach 9
94357 Konzell
oder per E-Mail an sb(at)it-bayer.de.

Blechschmidt, Stefan: *Hier fehlt noch was.* URL http://www.it-bayer.de. — [Stand Mai 2014]

Blechschmidt, Stefan: *Hier fehlt noch was.* URL http://www.it-bayer.de. — [Stand Mai 2014]

Wikipedia: *BibTeX — Wikipedia, Die freie Enzyklopädie.* URL http://de.wikipedia.org/w/index.php?title=BibTeX&oldid=124228120. — [Online; Stand 18. Dezember 2013]