

Notizen zu L^AT_EX

Jörg Desch
CC-BY-SA

Revision R.1

4. Juli 2016

Dieses Dokument ist keine Einführung in L^AT_EX. Es ist vielmehr eine lose Sammlung von Notizen zur Nutzung von L^AT_EX. Sie ist im Laufe einiger Jahre mehr oder weniger intensiver Nutzung entstanden, und war ursprünglich nicht für eine Veröffentlichung vorgesehen. Für mich ist dieses Dokument mehr ein Schmierblock mit Schnipseln und Tipps. Nichtsdestotrotz versuche ich nach der Veröffentlichung dieses Dokument weiter zu ergänzen und zu überarbeiten. Die Struktur des Dokuments ist über die Jahre gewachsen und sicherlich Gegenstand von Änderungen, sobald sich einmal die Zeit dazu bietet.

Inhaltsverzeichnis

1. Nutzung des Randbereichs	2
1.1. Notizen im Randbereich	2
1.2. Bilder im Randbereich	3
2. Besondere Darstellungen	3
2.1. Umranden von Text	3
2.2. Symbolische Darstellung von Tasten und Menüs	4
2.3. Eine Tastenliste	4

3. Von Boxen und anderen Schachteln	5
3.1. Eine Verbatim-Box	5
3.2. Eine leichtgewichtige Anmerkungsbox	6
3.3. Die große Lösung für Boxen	6
4. Eigene Zähler verwenden	8
4.1. Testpunkte automatisch erzeugen	8
5. Das Spiel mit den Einheiten	9
5.1. Listen von Zahlen	11
6. Silbentrennung	12
6.1. gezielte Hilfe bei der Trennung	13
6.2. Globale Silbentrennung	14
6.3. Verhindern der Silbentrennung	14
7. Das Spiel mit Sprachen	15
8. Das Spiel mit Farben	16
8.1. Eine pastellige Farbpalette	16
8.2. Überschriften einfärben	17
9. Verschiedene Kleinigkeiten	17
9.1. Leerzeichen sichtbar darstellen	17
9.2. Spacing der Paragraphen anpassen	18
A. Anhang	19
A.1. Historie	19

1. Nutzung des Randbereichs

1.1. Notizen im Randbereich

Der neue Befehl `marginlabel` erzeugt eine Randnotiz. Da diese nicht so sehr auffallen sollte, ist die kleiner gesetzt. Um Trennungsprobleme zu minimieren und den Abstand zum Fließtext, ist der Text rechtsbündig. Durch das führende `\hspace{0pt}` kann auch schon das erste Word getrennt werden. Dies ist wegen der geringen Breite sinnvoll.

*Eine kleine
Anmerkung
am Rande*

```
\newcommand{\marginlabel}[1]%  
  {\mbox{}\marginpar{\raggedleft\footnotesize%  
    \itshape\hspace{0pt}\#1}\ignorespaces}
```

1.2. Bilder im Randbereich

Der neue Befehl `marginfig` erzeugt ein Bild im Randbereich. Die `minipage` mit dem `vspace` ist wegen der Ausrichtung nötig. Ohne sie wäre der untere Rand des Bildes gleich dem der Zeile, in dem `marginfig` eingefügt wird. Der Trick mit `minipage[t]` und `vspace{-\baselineskip}` stammt aus „`epslatex.ps`“ (Part III Abschnitt 11.4.2).

`marginfig` bekommt den Namen des Bildes als Parameter übergeben. Des weiteren erlaubt der Befehl den Gebrauch eines optionalen Parameters, der unverändert an den Befehl `\includegraphics` durchgereicht wird.

```
\newcommand{\marginfig}[2] []%  
  {\mbox{}\marginpar{%  
    \begin{minipage}[t]{\marginparwidth}%  
    \par\vspace{-\baselineskip}%  
    \includegraphics[#1]{#2}%  
    \end{minipage}}}
```

Hinweis: Will man die Bilder auf die Breite des Randbereich skalieren, so sollte als Basis für diese Angabe die Variable `\marginparwidth` verwendet werden.

2. Besondere Darstellungen

2.1. Umranden von Text

Um zum Beispiel eine Taste zu stilisieren, kann man einen mit einem Oval umrahmten Text verwenden. Nachfolgendes Code-Schnipsel ist aus Kofler's Buch „Linux“ entnommen. Es zeichnet ein Oval um einen verkleinerten Text, ohne dass der Zeilenabstand verändert wird. Für die symbolische Darstellung von Tasten ist das unter Abschnitt 2.2 beschriebene Paket `menukeys` aber wesentlich besser geeignet.




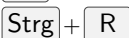

```

\newbox\mybox%
\newcount\laenge%
\newcount\laengehalbe%
\newcommand{\txtframe}[1]{%
\setbox\mybox=%
\hbox{\textsf{\footnotesize #1}}}%
\laenge=\wd\mybox%
\advance\laenge by 370000%
\laengehalbe=\laenge%
\divide \laengehalbe by 2%
\unitlength1sp%
\begin{picture}(\laenge,600000)(0,140000)%
  \put(\laengehalbe,300000){\oval(\laenge,600000)}%
  \put(185000,140000){\unhbox\mybox}%
\end{picture}}

```

2.2. Symbolische Darstellung von Tasten und Menüs

Das sehr interessante Paket **menukeys** erlaubt das einfache Setzen von Menühierarchien und Tasten. Mit dem Befehl kann man einen Menueintrag oder durch > getrennt ganze Hierarchien angeben. Für Tasten gibt den Befehl **keys**, der Einzeltasten und durch + verbunden auch Tastenkombinationen darstellen kann. Verzeichnispfade kann man mit **directory** abbilden.

<code>\menu{Extras}</code>	
<code>\menu{Extras > Setup > Leiste}</code>	
<code>\keys{A}</code>	
<code>\keys{Strg + R}</code>	
<code>\directory{/home/Your Name/Documents}</code>	

2.3. Eine Tastenliste

Um in einer Dokumentation eine ganze Reihe von Tasten zu beschreiben, bietet sich nochfolgende modifizierte Liste an. Die Umbegung **keylist** verwendet den unter Abschnitt 2.2 aufgezeigten Befehl **key** zur Darstellung von Tasten. Zudem werden der hängende Text nach der Taste (dem item) eingerückt. Das Maß der Einrückung wird

aus der Breite des als Parameter übergebenen Textes ermittelt. Man muss hier also die längste Tastenbeschriftung übergeben.

Das ganze ist eine Modifikation der Liste `Ventry` aus dem “LaTeX Begleiter”.

```
\newenvironment{Keylist}[1]%
{\begin{list}{}{%
  \renewcommand{\makelabel}[1]{\keys{##1}\hfill}%
  \setlength{\labelwidth}{#1}%
  \setlength{\leftmargin}{\labelwidth}%
  \addtolength{\leftmargin}{\labelsep}}}%
{\end{list}}}
```

3. Von Boxen und anderen Schachteln

3.1. Eine Verbatim-Box

Die Umgebung `verbatim` wird normalerweise immer linksbündig ausgegeben. Zudem lässt sie sich nicht zentrieren. Das kann man umgehen, in dem man die Umgebung `verbatim` in eine `minipage` packt. Dies erlaubt nun die Zentrierung oder auch die Verwendung einer `figure` Umgebung, so dass man nun auch Referenzen verwenden kann. Das nachfolgende Schnipsel zeigt eine solche Verwendung. In diesem Beispiel wird die Breite der `minipage` auf die Hälfte der Zeilenbreite reduziert.

```
\begin{figure}[htbp]
\centering
\begin{minipage}{0.5\linewidth}
\begin{verbatim}
DIES IST TEXT IN DER VERBATIM-BOX
\end{verbatim}
\end{minipage}
\caption{Die Bildunterschrift}
\label{fig:ein-label}
\end{figure}
```

3.2. Eine leichtgewichtige Anmerkungsbox

Kommentar: Dies ist eine Anmerkungsbox, wie sie mit dem *neuen* Environment `anmerkung` erzeugt wird. Das ganze bezieht sich auf die Zeilenbreite, und sollte somit auch bei zweispaltigem Satz korrekt arbeiten.

Diese hübsche und leichtgewichtige (da von Paketen unabhängige) Box erreicht man durch folgende neue Umgebung. Der optionale Parameter ist der “Titel” des Textes, der dem nachfolgenden Satz vorangestellt wird. Der Default ist **Anmerkung:**. Das ganze basiert übrigens auf einem Posting von Bernd Raichle. Ich habe nur ein paar kleine Änderungen vorgenommen.

```
\newenvironment{anmerkung}[1][Anmerkung:]{%
  \noindent%
  \begin{minipage}{\linewidth}%
    \smallskip%
    \rule{\linewidth}{.4pt}%
    \smallskip% == \vspace{\smallskipamount}%
    \newline\noindent\textbf{#1}%
    \enspace\ignorespaces%
  }{%
    \smallskip
    \rule{\linewidth}{.4pt}%
    \smallskip%
  }\end{minipage}%
}
```

3.3. Die große Lösung für Boxen

Mit dem Paket `mdframed` gibt es ein komplettes Paket das Rahmen zeichnet. Der Befehl `newmdenv` erlaubt eine einfache Definition einer neuen ”gerahmten” Umgebung. In den Optionen des Befehls kann die Umgebung umfangreich konfiguriert werden. Dies reicht von Sichtbarkeit oder Farbe der Linien, Abständen, Titel bis hin zur Farbe der Hintergründe der Box und des Titels.

Das Modul erlaubt verschiedene Zeichenmodule zu nutzen. Diese werden beim Einbinden des Pakets mit der Option `framemethod` gewählt. Nutzt man zum Beispiel `framemethod=tikz`, dann ist es möglich Boxen mit abgerundeten Ecken zu zeichnen.

Neben der Definition eigener Umgebungen mittels `newmdenv` kann man auch eine Standardbox mit lokalen Optionen verwenden. Die globalen Einstellungen des Pakets lassen sich mit `mdfsetup` verändern. Auf diese Features gehe ich aber nicht weiter ein (`\begin{mdframed}[OPTIONEN]`).

Die Farben in den Beispielen verwenden die Farbpalette aus Abschnitt 8.1.

Beispielbox

```
\newcounter{beispiel}
\newmdenv[
  linecolor   = myGreen ,
  linewidth   = .5ex ,
  rightline   = false ,
  leftline    = false ,
  frametitle  = {\refstepcounter{beispiel}\textbf{Beispiel~\thebeispiel.}}
]{beispiel}
```

Beispiel 1.

Das hier ist eine Beispielbox die nur horizontale Linien besitzt. Diese Box verwendet einen eigenen Zähler, welcher automatisch im Titel der Box verwendet wird. Innerhalb der Box sind Fußnoten^a erlaubt und werden innerhalb der Box dargestellt.

^awie diese hier

Infobox

```
\newmdenv[
  frametitle = {\textbf{Info}}
]{info}
```

Info

Das hier ist eine Infobox. Hier wird nur der Titel der Box als Option übergeben. Der Rest entspricht der Standardeinstellung.

Definitionsbox

```
\newmdenv[
  backgroundcolor = myYellowBrighter,
  roundcorner    = 10pt,
  linecolor      = myYellowDark,
  leftmargin     = 2em ,
  rightmargin    = 2em ,
  skipabove      = \baselineskip ,
  skipbelow      = \baselineskip ,
  frametitle     = Definition ,
  frametitlefont = \large\sffamily\scshape ,
  frametitlebackgroundcolor = myYellow
]{definition}
```

DEFINITION

Das hier ist eine Definitionsbox. Diese hat einen Hintergrund für den Titel und einen für die Box selbst. Der Titel ist in *small caps* gesetzt. Zudem wurden 2em linker und rechter Rand eingebaut. Als Besonderheit sind hier die Ecken der Box abgerundet.

4. Eigene Zähler verwenden

4.1. Testpunkte automatisch erzeugen

In vielen Bereichen werden Testpunkte, Anforderungen oder andere Punkte durchnummeriert, um diese Eindeutig referenzieren zu können. Das folgende L^AT_EX-Schnipsel zeigt einen Befehl, der zur Definition von Testpunkten dient. Er erlaubt die optionale Angabe

eines Labels, so dass ein Referenzieren möglich wird. Diese Version verzichtet absichtlich auf das Paket `ifthen` und verwendet die \TeX -Befehle `\ifx` und `\else`.

```
\newcounter{testnr}
\newcommand{\Testpunkt}[1]%
{\def\tmpval{#1}%
\refstepcounter{testnr}%
{\bfseries\par\medskip\noindent Test~\arabic{testnr}%
\ifx\tmpval\empty\else\label{\tmpval}\fi:\enspace\ignorespaces}}
```

Ein Beispiel für das Ergebnis dieses neuen Befehls sind die folgenden Testpunkte.

Test 1: Dies wäre der erste Testpunkt. Da es sich um einen ganz normalen Absatz handelt, kann der Text also ruhig länger werden.

Test 2: Dies ist der zweite Testpunkt.

Test 3: ...ist nun aber wirklich der letzte Punkt. Er folgt direkt auf Testpunkt 2.

Verwendet wird der Befehl durch `\Testpunkt{}` gefolgt von dem normalen Fließtext. Man kann diesem Befehl ein Label übergeben (`\Testpunkt{tp:xx}`), so dass sich mit `\ref{tp:xx}` auf die Nummer dieses Testpunkts beziehen kann.

Anmerkung: In einer eigenen QSM-Klasse sollte der Name “Testpunkt” über eine Variable konfigurierbar gemacht werden. Des weiteren sollte die Referenz nicht über “`\Testpunkt~\ref{}`” erfolgen. hierfür sollte ein eigener Befehl existieren, der den entsprechenden Namen aus der Variable des Pakets ließt.

5. Das Spiel mit den Einheiten

Um Zahlen mit Maßeinheiten zu verwenden, bieten sich die Pakete `siunitx` oder `units`. Das Erstere ist zu bevorzugen.

Zahlen wie 10 ms^{-1} oder 10 m sind mit dem Befehl `\SI{ }{ }` schnell geschrieben. Der Befehl wertet den Zahlenwert aus und nutzt das zur Formatierung der Zahl. Das macht Probleme, wenn man statt einer Zahl einen Bruch mit `\frac` angeben will. Um das zu

tun, muss man `SI` per Option das Untersuchen der Zahl verbieten. Das geschieht durch Angabe der Option `parse-numbers`.

Will man nur die Einheit formatieren, dann kann den Befehl `si` verwenden. Der arbeitet wie “der große Bruder” `SI`, erwartet aber einen Zahlenparameter.

Einfache Zeichen wie μ oder θ (`μ` oder `θ`) lassen sich aber auch direkt in einer eingebetteten Mathe-Umgebung mit deren Namen angeben. Ein Großbuchstabe im Namen erzeugt dann auch den passenden Großbuchstaben. Diese Zeichen werden dann aber wie in Formeln typisch kursiv geschrieben. Das Wikibook [LaTeX-Kompendium](#) liefert eine nette Übersicht.

Toleranzen werden dem Zahlenwert durch \pm getrennt nachgestellt. Die Angabe erfolgt allerdings in weniger geläufiger Schreibweise. Damit auch wirklich ein \pm erscheint, muss man die Option `separate-uncertainty=true` verwenden. Im Falle von `\SI` reicht das aber nicht. Hier wird der Zahlenwert zusammen mit der Toleranz geklammert ausgegeben. Die Option `multi-part-units=single` schaltet dies aber bei Bedarf wieder ab. Alternativ kann man mit `repeat` statt `single` erreichen, dass die Einheit sowohl dem Zahlenwert als auch der Toleranz angehängt wird.

Beispiele:

```
* \SI{10}{\meter}
* \SI{0.0001}{\meter\per\second}
* \SI[per-mode=fraction]{0.0001}{\meter\per\second}
* \SI[parse-number=false]{\frac{1}{5}}{\micro\meter}
* \si{\micro\meter}
* \si{\meter/\second^2}
* \SI{50 x 180 x 200}{mm}
* \ang{180,5} ist ein formatierter Winkel.
* \num{47+-0,5} oder \num[separate-uncertainty=true]{47+-0,5} sind
  Zahlenwerte mit Toleranzen.
* Zahlenwerte mit Toleranzen und Einheiten: \SI{7+-0,1}{\metre} oder
  \SI[separate-uncertainty=true,multi-part-units=single]{7+-0,1}{\metre}.
  Alternativ \SI[separate-uncertainty=true,multi-part-units=repeat]{7+-0,1}{\metre}
  oder \SI[separate-uncertainty=true]{7+-0,1}{\metre}

• 10 m
• 0,0001 m s-1
```

-
- $0,0001 \frac{\text{m}}{\text{s}}$
 - $\frac{1}{5} \mu\text{m}$
 - μm
 - m/s^2
 - $50 \text{ mm} \times 180 \text{ mm} \times 200 \text{ mm}$
 - $180,5^\circ$ ist ein formatierter Winkel.
 - $47,00(5)$ oder $47,00 \pm 0,05$ sind Zahlenwerte mit Toleranzen.
 - Zahlenwerte mit Toleranzen und Einheiten: $7,00(1) \text{ m}$ oder $7,00 \pm 0,01 \text{ m}$. Alternativ $7,00 \text{ m} \pm 0,01 \text{ m}$ oder $(7,00 \pm 0,01) \text{ m}$

Wem die Einheiten als Bruch besser gefallen, der kann das mit `per-mode=fraction` selektiv als Option angeben (siehe Beispiele). Global lässt sich das in die Präambel nach dem Einfügen des Paketes die folgende Konfigurationssequenz einbauen.

Des weiteren sollte man bei der Verwendung von Toleranzen die Klammerung durch das schönere \pm ersetzen. Dies erreicht man mit `separate-uncertainty=true`. Ohne weiteres zutun werden Werte mit Toleranzen allerdings geklammert. Dies lässt sich mit der Option `multi-part-units=single` abschalten. Wer die Einheit gerne hinter dem Zahlenwert *und* der Toleranz sehen will, der nimmt statt `single` einfach `repeat`.

```
\sisetup{
  per-mode=fraction,
  fraction-function=\tfrac,
  separate-uncertainty=true,
  multi-part-units=single
}
```

5.1. Listen von Zahlen

Gibt man eine Liste oder einen Bereich von Zahlen an, dann gehört die Einheit an jede Zahl. Das kann ziemlich nerven, weswegen die Entwickler von `siunitx` die Befehle `SIl`ist und `S`Irange eingebaut haben.

```
%\numlist{<Zahl;Zahl;Zahl>} formatiert Zahlenlisten
\numlist{10;20;30}
%\SIlist{<Zahl;Zahl;Zahl>}{<Einheit>} formatiert Zahlenlisten mit Einheit
\SIlist{10;20;30}{\meter}
%\SIRange{<Zahl>}{<Zahl>}{<Einheit>} formatiert Zahlenbereiche mit Einheiten
\SIRange{10}{20}{\meter}
```

Beispiele:

- 10, 20 und 30
- 10 m, 20 m und 30 m
- 10 m bis 20 m

Anmerkung: Für Bereiche und Listen setzt `siunitx` automatisch die passenden Begriffe “bis” oder “und” ein. Damit das auch in der richtigen Sprache erfolgt, reicht es *nicht* aus, einfach nur das Paket `babel` zu laden. Man muss hierfür die Sprache an Klassenoption an die Dokumentenklasse übergeben. Dadurch bekommen alle später geladenen Paket die Sprache mit. Das hat dann den Nebeneffekt, dass *KOMA-Script* auch gleich Babel mitlädt.

Beispiel: `\documentclass[ngerman]{scrartcl}`

Will man die Zahlenwerte den Spracheinstellungen anpassen, muss man auch die Option `locale=DE` an `siunitx` übergeben. Dies sorgt dann dafür, dass die Darstellung der Zahlen “eingedeutscht” wird.

6. Silbentrennung

Die Silbentrennung erfolgt in L^AT_EXan sich automatisch, mit einem speziellen Algorithmus. Dieser versagt jedoch ausnahmsweise, sowie systematisch in folgenden Fällen:

- In zusammengesetzten Wörtern. Z. B. wird **Staatsvertrag** nicht an der Wortfuge getrennt. Abhilfe leistet das Paket *hyphsubst* ¹

¹`\RequirePackage[ngerman=ngerman-x-latest]{hyphsubst}`

-
- Wörter mit Sonderzeichen (etwa: Umlaut) werden nicht immer getrennt, je nach den verwendeten Paketen und Schriften ([siehe Dante-FAQ: \(Silben-\)Trennung, Absatz-, Seitenumbruch](#))
 - Wörter, die einen Bindestrich enthalten, werden ausschließlich am Bindestrich getrennt, etwa `Karl-Franzensuniversität`.

6.1. gezielte Hilfe bei der Trennung

Mit folgenden Sequenzen lässt sich die Silbentrennung steuern:

- `\-`: Mit dieser Anweisung teilt man L^AT_EX mit, dass das Wort nur an der betreffenden Stelle getrennt werden kann. `Staats\-\ver\-\trag`. **Achtung!** Ist in einem Wort dieses Zeichen gesetzt, wird es an keiner anderen Stelle mehr getrennt werden. Daher sollte man, wenn diese Anweisung in einem Wort verwendet wurde, an jeder Stelle, an der man in diesem Wort einen Umbruch dulden möchte, diese Anweisung setzen.
- `"-`: Mittels dieser Anweisung wird eine zusätzliche Trennstelle angegeben. `Staats-vertrag` wird also von L^AT_EX je nach Bedarf auch nach der ersten Silbe getrennt werden. *Hinweis:* Dieser Befehl erfordert das Paket `german` oder `ngerman`.
- `"=`: Dies erzeugt einen Bindestrich, der die sonstigen Trennstellen im Wort weiterhin erlaubt: `Karl-Franzensuniversität` erlaubt die Trennungen `Karl-Fran-zens-uni-ver-si-`. *Hinweis:* Dieser Befehl erfordert das Paket `german` oder `ngerman`.
- `"`: Diese Anweisung erlaubt einen Umbruch, ohne dass ein Bindestrich gesetzt wird – etwa für den Fall, dass dieser bereits steht: `Karl-Fran\-\zens\-\uni\-\ver\-\si\-\tät`. *Hinweis:* Dieser Befehl erfordert das Paket `german` oder `ngerman`.
- `"~`: Mit dieser Anweisung setzt man einen Bindestrich, an dem nicht getrennt werden soll, beispielsweise bei Wortergänzungen in Klammern: `(Haupt"~)Straße`
- `\discretionary{Anfang}{Ende}{Ungetrennt}`: Definiert eine Trennhilfe für Wörter, deren Trennung an einer Stelle anders als erwartet aussieht. Ende ist dabei das Ende des ersten Wortteils in der getrennten Form, Anfang der Anfang des zweiten Wortteils in der getrennten Form, und Ungetrennt ist die betreffende Stelle in ungetrennter Form. Durch die Angabe aller Varianten bleibt LaTeX frei in der Platzierung und Trennung des betreffenden Wortes. Wörter mit “ck” hatten nach

alter Rechtschreibung (also vor 1996) eine ungewöhnliche Trennung, wie das Wort `\discretionary{rück-}{ken}{rücken}` zeigt.

6.2. Globale Silbentrennung

Am Anfang des Dokuments kann man eine Liste mit Worten inklusive Trennhilfen eintragen. Da diese Liste intern auf 300 Worte begrenzt ist, macht es keinen Sinn eine solche Liste zentral für alle Texte zu pflegen. Die Liste wird an den Befehl `hyphenation` übergeben. Wichtig ist, dass keine Sonderzeichen erlaubt sind. Umlaute muss man also als `\"a` schreiben. **CHECK:** ist das mit XeTeX wegen der UTF8-Nutzung anders?

```
\hyphenation{Staats-ver-trag Stau-be-cken Ver-st\"ar-ker-aus-gang}
```

Bei Zusammengesetzten Worten, deren Trennung komplexer ist (wie z.B. “LaTeX-Wörterbuch”), sollte man am einfachsten eigene Befehle definieren.

```
\newcommand{\LW}{LaTeX=Wörter"-buch\xspace}
```

Tipp: mit `\xspace` kann man ein “intelligentes” Leerzeichen einbauen. Danach kommt auch mit `\LW ...` ein Leerzeichen (wie bei `\LW{}` ...). Folgt dem Befehl aber ein Satzzeichen, dann wird kein Leerzeichen eingefügt!

6.3. Verhindern der Silbentrennung

Will man die Silbentrennung komplett verhindern, dann muss man dazu das Wort einfach mit `\mbox` einpacken. Das ist nicht sichtbar, verhindert aber die Trennung.

7. Das Spiel mit Sprachen

Das Handling der Sprachen in einem Dokument übernimmt das Paket `babel`. Für "Deutsch" sollte man die "neue" Rechtschreibung verwenden. Dazu gibt man dem Paket die Option `ngerman` mit auf den Weg. \LaTeX kümmert sich dann mit Hilfe von `babel` auch gleich um die Formatierung von länderspezifischen Dingen wie Zahlen oder dem Datum.

Jetzt kommt es nicht selten vor, dass in einem Text verschiedene Sprachen wendet werden müssen. Bei `babel` gibt man die zusätzlichen Sprachen durch Komma getrennt als weitere Paketoptionen an. **Wichtig:** die *letzte* Sprache ist die Sprache, die als Vorgabe verwendet wird.

Die Umschaltung von Sprachen erfolgt mit `selectlanguage`. Hierbei kann man den Textteil als zweiten Parameter angeben, oder man rahmt den Textbereich mit geschweiften Klammern und setzt den Befehl dort als erstes ein. Das nachfolgende Schnippsel zeigt die Umstellung:

```
\usepackage[english,ngerman]{babel}  %& default ist ngerman
...
Dies ist ein Test. Heute ist \today{}.
Ein Beispiel für \foreignlanguage{english}{small regions}.
{\foreignlanguage{english} Sample for a longer sequence}
\foreignlanguage{english}
Toggle to English for the following text.
\foreignlanguage{ngerman}
und zurück auf Deutsch. Beides Explizit.
```

Quellen:

1. [babel](#) auf CTAN.
2. [namsu.de: Babel Paket](#)

8. Das Spiel mit Farben

Um die Welt von \LaTeX farbiger zu gestalten, muss man das Paket `xcolor` laden. Dieses stellt ein Bündel standardfarben zur Verfügung. Die Namen der Farben sind teilweise nicht sehr eingänglich, weshalb man dem Paket eine Option für verschiedene Bezeichnungen anderer Werkzeuge aufzuzwingen. Man kann aber auch eigene Farben definieren. Das ist besonders interessant, da es im Internet eine Vielzahl von Generatoren für hübsch aussehende Farbkombinationen gibt. Mein persönlicher Favorit ist paletton.com. Hiermit habe ich eine [online erzeugte Palette](#) entworfen, die meinen Anforderungen für verschiedene Aufgaben erfüllt. Die RGB-Werte kann man dann mit Hilfe des Befehls `definecolor` einem Namen zuordnen.

Bei der Verwendung eigener Farben ist es sinnvoll, wenn man sich die Farben mit Namen definiert, die nicht mit anderen Namen kollidieren. Ich habe hier den Prefix `my` vorangestellt. In Anwendungen bei denen den Farben bestimmte bedeutungen zugeordnet werden, sollte der Farbname diese Bedeutung widerspiegeln. So kann man später die Farbe ändern, ohne den ganzen Text durchzusehen.

8.1. Eine pastellige Farbpalette

```
\definecolor{myRed}{RGB}{176, 63, 61}
\definecolor{myRedBrighter}{RGB}{255,174,172}
\definecolor{myRedBright}{RGB}{217,113,110}
\definecolor{myRedDark}{RGB}{138, 29, 26}
\definecolor{myRedDarker}{RGB}{ 97,  5,  3}
\definecolor{myYellow}{RGB}{176,139, 61}
\definecolor{myYellowBrighter}{RGB}{255,229,172}
\definecolor{myYellowBright}{RGB}{217,183,110}
\definecolor{myYellowDark}{RGB}{138,103, 26}
\definecolor{myYellowDarker}{RGB}{ 97, 67,  3}
\definecolor{myBlue}{RGB}{ 41, 90,111}
\definecolor{myBlueBrighter}{RGB}{113,148,164}
\definecolor{myBlueBright}{RGB}{ 72,117,137}
\definecolor{myBlueDark}{RGB}{ 20, 67, 87}
\definecolor{myBlueDarker}{RGB}{  4, 44, 61}
\definecolor{myGreen}{RGB}{ 44,126, 77}
\definecolor{myGreenBrighter}{RGB}{125,186,150}
```

```
\definecolor{myGreenBright}{RGB}{ 79,156,110}  
\definecolor{myGreenDark}{RGB}{ 19, 99, 51}  
\definecolor{myGreenDarker}{RGB}{ 2, 70, 29}
```



Abbildung 1: Beispiel einer erzeugten Farbpalette. Neben den Grundfarbton gibt es zwei hellere und zwei dunklere Farbtöne.

8.2. Überschriften einfärben

Da ich hauptsächlich die KOMA-Skripte verwende, beschreibe ich hier auch nur die elegante Lösung für dieses Paket. Es gibt einen Befehl `addtokomafont`, mit dem man einer Schriftengruppe ein neues Attribut zu einer bestehenden Konfiguration hinzufügen kann. Damit macht man nichts kaputt, und verändert keine anderen Einstellungen. Relevante Gruppen sind `sectioning` und `sectionentry`. Alternativ könnte man mit `setkomafont{disposition}` den gesamten Schriftensetup ersetzen.

```
\addtokomafont{sectioning}{\color{myBlueDark}}  
\addtokomafont{sectionentry}{\color{myBlueDark}}
```

9. Verschiedene Kleinigkeiten

9.1. Leerzeichen sichtbar darstellen

Um ein Leerzeichen als solches sichtbar in einem Text darzustellen verfügt \LaTeX über den Befehl `\textvisiblespace`. So zum Beispiel nachfolgend mit `a\textvisiblespace b`.

Ein explizit sichtbares Leerzeichen: a \LaTeX b

Die Breite dieses *sichtbaren Leerzeichens* ist aber fest definiert. Will man eine Variante mit definierbarer Breite, dann muss man sich einen eigenen Befehl bauen. Mit dem lassen sich dann nicht nur Leerzeichen ersetzen, sondern er eignet sich auch für die Darstellung von auszufüllenden Feldern in Formularen.

Während der originale Befehl ohne Parameter wie folgt aussieht,

```
\DeclareTextCommandDefault{\textvisiblespace}{%
  \mbox{\kern.06em\vrule \@height.3ex}%
  \vbox{\hrule \@width.3em}%
  \hbox{\vrule \@height.3ex}}
```

könnte eine Variante mit optionalem Parameter `\Vtextvisiblespace` so deklariert werden:

```
\newcommand\Vtextvisiblespace[1][.3em]{%
  \mbox{\kern.06em\vrule height.3ex}%
  \vbox{\hrule width#1}%
  \hbox{\vrule height.3ex}}
```

Ein sichtbares Leerzeichen: a_b (im Original)

Ein sichtbares Leerzeichen: a_b (Variante)

Ein sichtbares Leerzeichen mit 1em: a__b

Ein sichtbares Leerzeichen mit 1cm: a____b

9.2. Spacing der Paragraphen anpassen

Ohne zutun verwenden die meisten L^AT_EX Dokumentenklassen zur optischen Trennung der Paragraphen eine Einrückung ohne einen vertikalen Zeilenabstand. Um dass zu ändern, kann man in den *KOMA-Script* Dokumentenklassen die Klassenoption `parskip=half` übergeben. Für andere Dokumentenklassen ist das Paket `parskip` einfügen. Wer ein wenig mehr Kontrolle haben möchte, der kann stattdessen die nachfolgenden Einstellungen verwenden.

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1.3ex plus 0.5ex minus 0.3ex}
```

Mit `parindent` wird die Einrückung deaktiviert. Die Einstellung mit `parskip` erfolgt auf Basis der Größeneinheit `ex`. Dies entspricht der Höhe eines kleinen 'x' in der aktuellen Schriftgröße des Fließtextes. Die hier angegebenen `1.3ex` sind also 30% höher als das kleine 'x'. Mit den Angaben *plus* und *minus* wird L^AT_EX der mögliche Spielraum bei der Variation des Abstandes angegeben. Der Bereich der Einstellung liegt also hier bei `1ex` bis `1.8ex`.

Alternativ könnte man auch mit der fixen Größeneinheit `pt` arbeiten.

```
\setlength{\parskip}{6pt plus 2pt minus 1pt}
```

A. Anhang

A.1. Historie

Revision	Kommentar
R.1	erste Veröffentlichung