Sdoc

Frank Seitz, http://fseitz.de/

28. Januar 2018

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument beschreibt die Syntax und den Gebrauch der Auszeichnungsprache Sdoc. Sdoc ist eine Metasprache zum Verfassen von Dokumenten. Ein Dokument, das in Sdoc geschrieben ist, kann portabel in eine Online-Fassung (HTML) und in eine Druck-Fassung (PDF) übersetzt werden.

Die Online-Fassung ist für die Wiedergabe in einem Browser vorgesehen. Das Aussehen kann via CSS gestaltet werden.

Die Druck-Fassung wird von LaTeX gesetzt und besitzt eine entsprechend hohe Qualität für die Wiedergabe auf Papier. In einem PDF-Viewer kann die Druck-Fassung auch interaktiv genutzt werden, da sie mit Hyperlinks versehen ist.

Dieses Dokument ist in Sdoc geschrieben. Wie es übersetzt wird, ist in Abschnitt Dieses Dokument übersetzen auf Seite 31 beschrieben.

2 Beispiel

2.1 Sdoc

```
%Document:
  title="Ein Beispieldokument"
  author="Nico Laus"
  date="today"
  latexGeometry="includeheadfoot,margin=2.6cm"

= Einleitung

Dies ist eine I{Einleitung}. Mit wenig C{Aufwand} erstellen wir in
Sdoc ein B{Dokument}. Als Textauszeichnungen sind auch Kombinationen
wie B{I{fett und kursiv}} oder C{I{monospace und kursiv}} möglich.

== Ein Beispiel

Mit dem Satz des Pythagoras M~a^2+b^2=c^2~ berechnen wir die Länge der
Hypotenuse: M~c=\sqrt{a^2+b^2}~.

%Graphic:
  file="+/sdoc-graphic-illusion"
  align="center"
  scale=0.3

Programm-Quelltexte können wir mit Syntax-Highlighting verschönern:

%Code: lang=Perl ln=1 indent=1
  open(my $fh, 'ls -l |') or die "open failed ($!)\n";
  while (<$fh>) {
      s/\n/<BR>\n/;
      print;
  }
  close($fh) or die "close failed ($!)\n";
.

Auf eine andere Stelle im Dokument zu verweisen ist einfach. Hier ist
ein Verweis auf die L{Einleitung}.

* Apfel
* Birne
* Pflaume

Das ist eine einfache Aufzählung. Aufzählungen können beliebig
geschachtelt werden.

# eof
```

2.2 PDF

Ein Beispieldokument

Nico Laus

28. Januar 2018

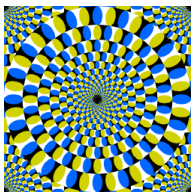
Inhaltsverzeichnis

1 Einleitung

Dies ist eine *Einleitung*. Mit wenig `Aufwand` erstellen wir in Sdoc ein **Dokument**. Als Textauszeichnungen sind auch Kombinationen wie ***fett und kursiv*** oder ***`monospace und kursiv`*** möglich.

1.1 Ein Beispiel

Mit dem Satz des Pythagoras $a^2 + b^2 = c^2$ berechnen wir die Länge der Hypotenuse: $c = \sqrt{a^2 + b^2}$.



Programm-Quelltexte können wir mit Syntax-Highlighting verschönern:

```
1 open(my $fh, 'ls -l |') or die "open failed ($!)\n";
2 while (<$fh>) {
3     s/\n/<BR>\n/;
4     print;
5 }
6 close($fh) or die "close failed ($!)\n";
```

Auf eine andere Stelle im Dokument zu verweisen ist einfach. Hier ist ein Verweis auf die Einleitung.

- Apfel
- Birne
- Pflaume

Das ist eine einfache Aufzählung. Aufzählungen können beliebig geschachtelt werden.

3 Syntax

3.1 Allgemeines

3.1.1 Kommentare

In den Quelltext des Dokuments können Kommentarzeilen eingebettet werden. Diese werden nicht mit gesetzt, aber als Kommentare in den Quelltext des Zielformats übertragen. Eine Kommentarzeile beginnt mit einem (#) am *Zeilenanfang*. Eine Einrückung ist nicht erlaubt.

```
# Dies ist ein Kommentar, der nicht im Zieldokument erscheint
```

Siehe auch Abschnitt Kommentar auf Seite 25.

3.1.2 Zeilenfortsetzungen

Ist das letzte Zeichen einer Zeile ein Backslash (\), wird die nächste Zeile als eine Fortsetzung dieser Zeile angesehen und mit dieser zusammengefasst.

```
Hier ist ein URL, der sich in der Quelle über zwei
Zeilen erstreckt: https://de.wikipedia.org/\
wiki/Auszeichnungssprache
```

produziert:

```
Hier ist ein URL, der sich in der Quelle über zwei
Zeilen erstreckt: https://de.wikipedia.org/wiki/Auszeichnungssprache
```

Leerraum am Anfang der Folgezeile wird entfernt. Soll Leerraum zwischen Zeile und Fortsetzungszeile erhalten bleiben, muss dieser *vor* dem Backslash stehen.

Eine Zeilenfortsetzung kann unterdrückt werden, indem der Backslash am Ende der Zeile durch einen vorangestellten Backslash maskiert wird. Der maskierende Backslash wird automatisch entfernt.

3.1.3 Leerzeilen

Leerzeilen können zur besseren optischen Gliederung zwischen aufeinanderfolgenden Elementen eingestreut werden, sind aber optional. Es macht keinen Unterschied, ob zum Beispiel zwischen einem Abschnittstitel und dem darauf folgenden Paragraphen eine Leerzeile steht oder nicht. Dasselbe gilt für alle anderen Kombinationen von Elementen. Einzige Ausnahme: Zwischen Paragraphen ist eine Leerzeile notwendig, um sie voneinander abzugrenzen.

3.1.4 Pfad-Expansion

Wird im Sdoc-Quelltext auf eine lokale Datei Bezug genommen und beginnt ihr Pfad mit `+/`, wird das Pluszeichen zum Verzeichnis der Dokumentdatei expandiert.

3.1.5 Block-Syntax und Wiki-Syntax

Der Quelltext eines Sdoc-Dokuments kann als eine Abfolge von *Blöcken* angesehen werden. Einige dieser Blöcke können *Segmente* enthalten.

Die allgemeine Schreibweise für einen Block ist

```
%TYPE: KEY=VAL ...
TEXT
.
```

Hierbei ist `TYPE` der Block-Typ, `KEY=VAL ...` eine Liste von Block-Eigenschaften und `TEXT` ist der Block-Text (der, je nach Block-Typ auch fehlen kann). Der Block-Text kann Segmente enthalten. *Jedes* der Strukturelemente eines Sdoc-Dokuments kann in Block-Syntax notiert werden.

Daneben definiert Sdoc für *einige* - nicht alle - Block-Typen eine kompaktere Wiki-Syntax. Ein Abschnitt (Section) kann beispielsweise in Wiki-Syntax als

```
== Dies ist ein Abschnitt der Ebene 2
```

oder in Block-Syntax als

```
%Section:
  level=2
  title="Dies ist ein Abschnitt der Ebene 2"
```

notiert werden. Die Wiki-Syntax ist "Syntaktischer Zucker", um das Dokument kompakter und lesbarer zu halten. In Abschnitt Beispiel in Block-Syntax auf Seite 28 ist das Beispiel aus Abschnitt Beispiel/Sdoc auf Seite 5 vollständig in Block-Syntax wiedergegeben.

3.2 Dokument (Document)

Information über das Dokument wird in einem - optionalen - Dokument-Bock angegeben:

```
%Document:
  anchor=STRING
  author=STRING
  copyComments=BOOL
  date=STRING
  language=german|english
  latexDocumentClass=scrartcl|scrreprt|scrbook|article|report|book
  latexFontSize=10pt|11pt|12pt
  latexGeometry=STRING
  latexPaperSize=PAPERSIZE
  latexParSkip=LENGTH
  sectionNumberDepth=0|1|2|3|4|5
  smallerMonospacedFont=BOOL
  tableOfContents=BOOL
  title=STRING
```

Der Dokument-Block kann irgendwo im Dokument stehen. Es ist aus Gründen der Übersichtlichkeit aber sinnvoll, ihn an den Anfang zu setzen.

Besitzt mindestens eines der Attribute `author`, `date` oder `title` einen Wert, wird ein Dokument-Vorspann mit den entsprechenden Angaben generiert.

Enthält das Dokument keinen Dokument-Block, wird kein Dokument-Vorspann generiert und alle Attribute erhalten ihre Defaultwerte (siehe Abschnitt Eigenschaften).

3.2.1 Segmente

In Titel (`title`), Autor (`author`) und Datum (`date`)des Dokuments können Segmente (siehe Abschnitt Syntax/Segment auf Seite 26) verwendet werden:

```
A{}, B{}, C{}, G{}, I{}, L{}, M{}, Q{}
```

3.2.2 Eigenschaften

anchor Anker des Dokuments.

author Der Autor des Dokuments.

date Das Datum des Dokuments. Formatelemente der Funktion strftime(3) werden expandiert. Spezielle Wert sind:

'today' Wird unter LaTeX ersetzt zu: `\today`, sonst `%d. %B %Y`

'now' Wird expandiert zu: `%Y-%m-%d %H:%M:%S`

Ist kein Wert angegeben, erscheint kein Datum im Dokumenttitel.

copyComments Kopiere Sdoc-Kommentare in den Quelltext des Zielformats. Dies ist z.B. nützlich, um eine Stelle im Quelltext des Zielformats zu finden, die aus einem bestimmten Sdoc-Konstrukt hervorgegangen ist.

language (Default: german) Die Sprache, in der das Dokument verfasst ist. In dieser Sprache werden Dokument-Bezeichnungen wie z.B. "Inhaltsverzeichnis" erzeugt. Ferner bestimmt die Sprache in LaTeX die Trennregeln. Mögliche Werte: german, english.

latexDocumentClass LaTeX-Dokumentklasse. Werte: 'scrartcl', 'scrreprt', 'scrbook', 'article', 'report', 'book', ...

latexFontSize (Default: '10pt') Die Größe des LaTeX-Font. Mögliche Werte: '10pt', '11pt', '12pt'.

latexGeometry Definition der Seitengeometrie mittels des LaTeX-Pakets `geometry`.

latexPaperSize Papiergröße für LaTeX.

sectionNumberDepth (Default: 3) Die Abschnittsebene, bis zu welcher Abschnitte numeriert werden. Mögliche Werte: 0, 1, 2, 3, 4, 5. 0 bedeutet: Die Abschnitte des Dokuments werden nicht numeriert.

smallerMonospacedFont Wähle einen kleineren Monospaced Font als standardmäßig.

tableOfContents (Default: 1) Erzeuge ein Inhaltsverzeichnis, auch wenn das Dokument keinen TableOfContents-Block enthält.

title Der Titel des Dokuments.

3.3 Inhaltsverzeichnis (TableOfContents)

Das Inhaltsverzeichnis wird an die Stelle plaziert, wo der TableOfContents-Block steht.

```
%TableOfContents:
  maxDepth=N
```

Der TableOfContents-Block ist optional. Kommt er im Dokument nicht vor, wird das Inhaltsverzeichnis automatisch nach dem Dokument-Vorspann eingefügt, wenn die Dokument-Eigenschaft `tableOfContents=1` gesetzt ist (siehe Abschnitt Dokument/Eigenschaften auf Seite 9). Ist `tableOfContents=0` gesetzt, findet dies nicht statt und das Dokument erhält kein Inhaltsverzeichnis.

3.3.1 Eigenschaften

maxDepth (Default: 3) Tiefe, bis zu welcher Abschnitte ins Inhaltsverzeichnis aufgenommen werden. Mögliche Werte: 0, 1, 2, 3, 4, 5. 0 = kein Inhaltsverzeichnis.

3.4 Abschnitt (Section)

Ein Abschnitt beginnt mit einem oder mehreren =, gefolgt von mindestens einem Leerzeichen, gefolgt von dem Abschnittstitel. Die Zahl der = gibt die Ebene des Abschnitts an. Der Titel endet mit dem Ende der Zeile. Längere Titel können per Zeilenfortsetzung geschrieben werden.

```
= Dies ist ein Abschnitt der Ebene 1

== Dies ist ein Abschnitt der Ebene 2

=== Dies ist ein Abschnitt der Ebene 3
```

Abschnitte werden optional mit 1. 1.1. 1.2. ... 2. 2.1. usw. durchnummeriert, wenn das %Document-Attribut `sectionNumberDepth` auf einen Wert > 0 gesetzt wird.

3.4.1 Unterabschnitte vom Inhaltsverzeichnis ausschließen

Sollen die *Unterabschnitte* eines Abschnitts nicht im Inhaltsverzeichnis erscheinen, wird an die Folge von = des Abschnitts ein Ausrufungszeichen (!) angehängt. Mit dem Abschnitt endet der betreffende Teilbaum im Inhaltsverzeichnis. Die unterdrückten Unterabschnitte müssen nicht gekennzeichnet werden.

```
= Abschnitt A (erscheint in Inhaltsverzeichnis)

==! Abschnitt AB (erscheint in Inhaltsverzeichnis)

=== Abschnitt ABC (erscheint nicht in Inhaltsverzeichnis)

==== Abschnitt ABCD (erscheint nicht in Inhaltsverzeichnis)
```

Im Unterschied zum %TableOfContents-Attribut `maxDepth` können auf diesem Weg Abschnitte unterschiedlicher Ebenen vom Inhaltsverzeichnis ausgeschlossen werden.

3.4.2 Übergeordnete Ebenen

Den Abschnittsebenen 1 bis 4 sind zwei weitere Ebenen *übergeordnet*, die für größere Dokumente oder zur Vereinigung mehrerer Dokumente verwendet werden können. Dies sind die Ebenen

```
=- Teil

==- Kapitel
```

In LaTeX stehen diese zur Verfügung, wenn die Dokumentklassen `scrreprt`, `scrbook`, `report`, `book` verwendet werden.

3.4.3 Appendix

Die Appendizes eines Dokuments beginnen ab dem ersten Abschnitt der Ebene 1, bei dem auf die = ein Pluszeichen (+) folgt:

```
=+ Mit diesem Abschnitt beginnen die Appendizes
```

Alle folgenden Abschnitte und Unterabschnitte werden als Appendizes angesehen (unabhängig davon, ob sie mit + ausgezeichnet oder nicht) und entsprechend mit A. A.1. A.2 ... B. B.1. usw. numeriert. Sonst gibt es keinen Unterschied zwischen Abschnitten und Appendizes.

3.4.4 Segmente

Im Titel eines Abschnitts können Segmente (siehe Abschnitt Syntax/Segment auf Seite 26) verwendet werden:

```
A{}, B{}, C{}, G{}, I{}, L{}, M{}, Q{}
```

3.4.5 Eigenschaften

```
%Section:
  anchor=STRING
  isAppendix=BOOL
  level=N
  stopToc=BOOL
  title=STRING
```

anchor Anker des Abschnitts.

isAppendix Mit diesem Abschnitt beginnen die Appendizes.

level Tiefe des Abschnitts in der Abschnittshierarchie, beginnend mit 1.

stopToc Alle Abschnitte unterhalb dieses Abschnitts werden nicht in das Inhaltsverzeichnis aufgenommen.

title Titel des Abschnitts.

3.5 Zwischenüberschrift (BridgeHead)

Ein Abschnitt wird vom Inhaltsverzeichnis ausgeschlossen und damit lediglich als Zwischenüberschrift genutzt, wenn an die Folge von = ein Stern (*) angehängt wird.

```
==* Zwischenüberschrift Stufe 2

===* Zwischenüberschrift Stufe 3

====* Zwischenüberschrift Stufe 4
```

produziert:

Zwischenüberschrift Stufe 2

Zwischenüberschrift Stufe 3

Zwischenüberschrift Stufe 4

3.5.1 Segmente

In einer Zwischenüberschrift können Segmente (siehe Abschnitt Syntax/Segment auf Seite 26) verwendet werden:

```
A{}, B{}, C{}, G{}, I{}, L{}, M{}, Q{}
```

3.5.2 Eigenschaften

```
%BridgeHead:
  anchor=STRING
```

```
level=N
title=STRING
```

anchor Anker der Zwischenüberschrift.

level Größe der Zwischenüberschrift, beginnend mit 1.

title Titel der Zwischenüberschrift.

3.6 Seitenumbruch (PageBreak)

Ein Seitenumbruch wird erzeugt durch eine Zeile mit dem Inhalt

```
---PageBreak---
```

Dies darf nur der einzige Inhalt der Zeile sein, ohne Leerraum davor oder dahinter. Ein Seitenumbruch ist nur in der Druck-Fassung von Bedeutung bzw. wenn die Online-Fassung gedruckt wird.

```
Dies ist ein
Pragraph.

---PageBreak---

Dieser Paragraph beginnt in
der gedruckten Version des Dokuments
auf einer neuen Seite.
```

3.6.1 Eigenschaften

```
%PageBreak:
```

Der Seitenumbruch hat keine Eigenschaften.

3.7 Paragraph (Paragraph)

Ein Paragraph ist ein Textblock ohne Einrückung, der mit einer Leerzeile oder mit dem Beginn eines anderen Elements (Liste, Code, etc.) endet.

```
= Dies ist ein Abschnitt

Dies ist ein Paragraph auf Abschnittsebene,
der sich in der Quelle über drei
Zeilen erstreckt.

* Dies ist ebenfalls ein Paragraph,
  allerdings auf Listenebene.
* Dies ist der zweite Punkt derselben
  Liste. Noch ein Paragraph.
```

3.7.1 Segmente

In einem Paragraph können Segmente (siehe Abschnitt Syntax/Segment auf Seite 26) verwendet werden:

```
B{}, C{}, G{}, I{}, L{}, M{}, Q{}
```

3.7.2 Eigenschaften

```
%Paragraph:
TEXT
.
```

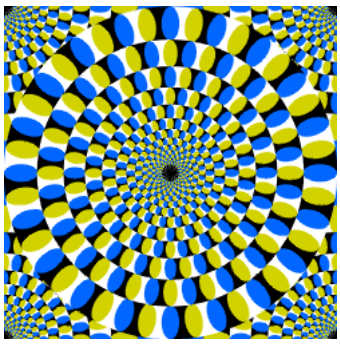
Ein Paragraph hat keine Eigenschaften.

3.8 Abbildung (Graphic)

Eine Abbildung ist ein freistehendes Bild zwischen zwei Paragraphen oder anderen Block-Elementen.

```
%Graphic:
  file="+/sdoc-graphic-illusion"
  scale=0.5
```

produziert:



Für eine Grafik, die *im Textfluss* erscheinen soll, siehe Abschnitt G - Inline-Grafik auf Seite 26.

3.8.1 Eigenschaften

```
%Graphic:
  align=left|center|right
  file=PATH
  definition=BOOL
  latexOptions=STRING
  name=STRING
  scale=X
```

align (Default: 'left') Horizontale Ausrichtung des Bildes. Mögliche Werte: 'left', 'center', 'right'.

file Pfad der Bilddatei. Beginnt der Pfad mit `+/`, wird das Pluszeichen zum Pfad des Dokumentverzeichnisses expandiert.

definition Wenn gesetzt, stellt der Grafik-Block lediglich eine Definition dar, d.h. die Grafik wird nicht an dieser Stelle angezeigt, sondern an anderer Stelle von einem G-Segment referenziert. Ist Attribut `name` definiert, ist der Default 1, andernfalls 0.

latexOptions LaTeX-spezifische Optionen, die direkt an das LaTeX-Makro `\includegraphics` übergeben werden.

name Name der Grafik. Ein Name muss angegeben sein, wenn die Grafik von einem G-Segment referenziert wird. Ist ein Name gesetzt, ist der Default für das Attribut `definition` 1, sonst 0.

scale Skalierungsfaktor. Im Falle von LaTeX wird dieser zu den `latexOptions` hinzugefügt, sofern dort kein Skalierungsfaktor angegeben ist.

3.9 Aufzählung (List)

Es gibt drei verschiedene Arten von Listen.

3.9.1 Punktliste

Eine Punktliste (oder *unordered list*) ist eine Abfolge von Punkten, deren erste Zeile jeweils mit einem Stern (`*`) beginnt.

```
* Apfel
* Birne
* Pflaume
```

produziert:

- Apfel
- Birne
- Pflaume

3.9.2 Numerierungsliste

Eine Numerierungsliste (oder *ordered list*) ist eine Abfolge von Punkten, deren erste Zeile jeweils mit einer Zahl gefolgt von einem Punkt (`.`) beginnt.

```
1. Staub wischen
2. Wäsche waschen
3. Essen kochen
```

produziert:

1. Staub wischen
2. Wäsche waschen
3. Essen kochen

3.9.3 Definitionsliste

Eine Definitionsliste (oder *definition list*) ist eine Abfolge von Punkten, deren erste Zeile jeweils aus einem Term in eckigen Klammern gefolgt von einem Doppelpunkt `[TERM]:` beginnt.

```
[255 239 213]:
  PapayaWhip
[188 143 143]:
  RosyBrown
[255 218 185]:
  PeachPuff
```

produziert:

255 239 213 PapayaWhip
188 143 143 RosyBrown
255 218 185 PeachPuff

3.9.4 Verschachtelung von Listen

Listen können beliebig verschachtelt werden:

```
* A
  1. B
  2. C
     [D]:
       E
     [F]:
       G
  3. H
* I
* J
```

produziert:

- A
  1. B
  2. C

     D E

     F G
  3. H
- I
- J

3.9.5 Segmente

Im Term einer Definitonsliste können Segmente (siehe Abschnitt Syntax/Segment auf Seite 26) verwendet werden:

```
B{}, C{}, I{}, L{}, M{}, Q{}
```

3.9.6 Eigenschaften

List

```
%List:
  listType=TYPE
```

listType Art der Liste. Mögliche Werte: 'unordered', 'ordered',, 'description'. Wenn nicht gesetzt, wird der Wert vom ersten List-Item gesetzt.

Item

```
%Item:
  key=STRING
```

key Stern (*) im Falle einer Punktliste. Nummer im Falle einer Nummerierungsliste. Zeichenkette im Falle einer Definitionsliste.

3.10 Tabelle (Table)

Eine Tabelle wird durch eine ASCII-Darstellung, eingefasst in einen Table-Block, definiert.

```
%Table:
Links Rechts Zentriert
----- ------ ---------
A          1    AB
AB        12    CD
ABC      123    EF
ABCD    1234    GH
.
```

produziert:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF
ABCD	1234	GH

3.10.1 Definition

Die Definition umfasst den Kopf und den Körper der Tabelle. Der Kopf enthält die Kolumnen-Titel, der Körper die Kolumnen-Daten. Die beiden Bereiche werden durch eine Trennzeile aus Bindestrichen (-) und Leerzeichen ( ) getrennt. Neben der Trennung von Titel und Daten hat die Trennzeile die wichtige Aufgabe, die Lage und die Breite der Kolumnen zu definieren.

Obige Tabelle besitzt aufgrund der drei Abschnitte aus Bindestichen

```
----- ------ ---------
```

drei Kolumnen:

- Die erste Kolumne ist 5 Zeichen breit und reicht von Zeichenposition 1 bis 5.
- Die zweite Kolumne ist 6 Zeichen breit und reicht von Zeichenposition 7 bis 12.
- Die dritte Kolumne ist 9 Zeichen breit und reicht von Zeichenpostion 14 bis 20.

Alle Zeichen außerhalb dieser Bereiche werden *nicht* beachtet. Dies gilt sowohl für die Kolumnentitel als auch die Kolumnendaten.

3.10.2 Ausrichtung

Aus der Anordnung der Werte in einer Kolumne - sowohl Titel als auch Daten - ergibt sich, ob die Kolumne links, rechts oder zentriert ausgerichtet ist. Bei einer links ausgerichteten Kolumne belegen *alle* (nichtleeren) Werte die erste Zeichenpositon. Bei einer rechts ausgerichteten Kolumne belegen *alle* (nichtleeren) Werte die letzte Zeichenpositon. Bei einer zentrierten Kolumne sind die Werte weder eindeutig links noch rechts ausgerichtet.

3.10.3 Titelzeile

Die Titelzeile ist optional. Sie kann auch weggelassen werden. Die Trennzeile für die Kolumnendefinition ist trotzdem erforderlich.

```
%Table:
----- ------ ---------
A           1    AB
AB         12    CD
ABC       123    EF
ABCD     1234    GH
.
```

produziert:

A	1	AB
AB	12	CD
ABC	123	EF
ABCD	1234	GH

3.10.4 Mehrzeilige Daten

Sowohl die Kolumnentitel als auch die Kolumnendaten können mehrzeilig sein. Für die Kolumentitel gilt dies generell. Kommen in der Tabelle mehrzeilige Kolumnendaten vor, müssen *alle* Datenzeilen mit einer Trennzeile voneinander getrennt werden.

```
%Table:
  Right  Left
Aligned  Aligned       Centered
-------  ------------  --------
      1  Dies ist die     A
         erste Zeile

      2  Zweite Zeile     B

      3  Die dritte       C
         Zeile
.
```

produziert:

Right Aligned	Left Aligned	Centered
1	Dies ist die erste Zeile	A
2	Zweite Zeile	B
3	Die dritte Zeile	C

3.10.5 Beschriftung

Ist das Attribut `caption` gesetzt, wird der Tabelle eine Beschriftung hinzugefügt.

```
%Table:
  caption="Eine Tabelle"
Links Rechts Zentriert
----- ------ ---------
A           1    AB
AB         12    CD
.
```

produziert:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD

Tabelle 4: Eine Tabelle

In der Beschriftung können die Segmente A, B, C, I, G, M, L, Q verwendet werden.

3.10.6 Linien

Mittels des Attributs `border` kann die Tabelle mit Linien versehen werden.

Beispiele

Keine Linien:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF

Tabelle 5: border=0

Linie unter Titelzeile:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF

Tabelle 6: border=t

Linie unter Titel und ober- und unterhalb der Tabelle:

Links	Rechts	Zentriert
A	1	AB

Fortsetzung nächste Seite

Fortsetzung

Links	Rechts	Zentriert
AB	12	CD
ABC	123	EF

Tabelle 7: border=tH

Horizontale Linien:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF

Tabelle 8: border=hH

Innere horizontale und vertikale Linien:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF

Tabelle 9: border=hv

Linie unter Titel und um die Tabelle:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF

Tabelle 10: border=tHV

Alle horizontalen Linien und vertikale Außenlinien:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF

Tabelle 11: border=hHV

Alle Linien:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD
ABC	123	EF

Tabelle 12: border=hvHV

3.10.7 Segmente

In der Beschriftung (`caption`) und allen Zellen einer Tabelle können alle Segmente (siehe Abschnitt Syntax/Segment auf Seite 26) verwendet werden:

```
A, B, C, G, I, L, M, Q
```

3.10.8 Eigenschaften

```
%Table:
  anchor=STRING
  border=STRING
  caption=STRING
TEXT
.
```

anchor Anker der Tabelle.

border Legt fest, welche Linien in und um die Tabelle gezeichnet werden.

- **t** Linie zwischen Titel und Daten.
- **h** Horizontale Linien *zwischen* den Zeilen. Impliziert t.
- **v** Vertikale Linien *zwischen* den Spalten.
- **H** Horizontale Linien ober- und unterhalb der Tabelle.
- **V** Vertikale Linien links und rechts von der Tabelle.

caption Die Beschriftung der Tabelle. Diese erscheint unter der Tabelle.

3.11 Code (Code)

Ein Code-Abschnitt ist ein literaler Textblock, der, abgesehen von Einrückung, Farbgebung oder Zeilennumerierung, im Zielformat exakt so wiedergegeben wird wie er in der Quelle steht, einschließlich Leerzeichen und Zeilenumbrüchen.
Es gibt zwei Möglichkeiten einen literalen Textblock anzugeben.

3.11.1 Einrückung mit Leerraum

Erstens entsteht ein Code-Abschnitt durch Einrückung mit ein oder mehr Leerzeichen am Zeilenanfang. Die Einrückung wird automatisch entfernt.

```
open(my $fh, 'ls -l |') or die "open failed ($!)\n";
while (<$fh>) {
    s/\n/<BR>\n/;
    print;
```

```
}
close($fh) or die "close failed ($!)\n";
```

3.11.2 In Block-Syntax

Zweitens kann ein Code-Abschnitt als Block definiert werden. Der Block beginnt mit `%Code:` und endet mit einem einzelnen Punkt (`.`) auf einer eigenen Zeile. Dazwischen ist eine Einrückung mit Leerzeichen erlaubt, aber nicht vorgeschrieben. Diese wird ebenfalls automatisch entfernt.

```
%Code:
  Dies ist ein
  Code-Block
.
```

produziert:

```
Dies ist ein
Code-Block
```

3.11.3 Syntax-Highlighting

Enthält der Code-Abschnitt den Quelltext einer Programmiersprache, lässt sich dieser mit Syntax-Highlighting aufwerten. Hierzu wird der Quelltext in einen Code-Block eingefasst und mit dem Attribut `lang=LANGUAGE` versehen. Hierbei ist `LANGUAGE` der Name der betreffenden Programmiersprache.

```
%Code: lang=Perl
  sub gcd {
      my ($class,$a,$b) = @_;
      return $b == 0? $a: $class->gcd($b,$a%$b);
  }
.
```

produziert:

```
sub gcd {
    my ($class,$a,$b) = @_;
    return $b == 0? $a: $class->gcd($b,$a%$b);
}
```

3.11.4 Zeilennummern

Wird das Attribut `ln=N` mit `N` $>$ `0` gesetzt, werden die Zeilen durchnummeriert, beginnend mit Zeilennummer `N`. Die Nummerierung findet auch statt, wenn kein Syntax-Highlighting aktiviert ist.

```
%Code: lang=Perl ln=117
  use DirHandle;

  my $dh = DirHandle->new($dir);
  while (my $entry = $dh->next) {
      say $entry;
  }
  $dh->close;
.
```

produziert:

```
117 use DirHandle;
118
119 my $dh = DirHandle->new($dir);
120 while (my $entry = $dh->next) {
121     say $entry;
122 }
123 $dh->close;
```

3.11.5 Text aus Datei laden

Der Text des Code-Abschnitts wird bei Angabe von `load=PATH` aus Datei `PATH` gelesen. Ist `PATH` ein relativer Pfad, wird diesem das Verzeichnis der Dokumentdatei vorangestellt.

```
%Code: load="/etc/timezone"
```

produziert:

```
Europe/Berlin
```

3.11.6 Text von Programmaufruf

Der Text des Code-Abchnitts wird bei Angabe von `exec=COMMAND` vom Kommando `COMMAND` geliefert und zwar von dessen Ausgabe auf stdout und stderr. Beginnt COMMAND mit einem Plus (+) wird dem Kommando der Pfad des Dokumentverzeichnisses vorangestellt.

```
%Code: exec="date --date=010101"
```

produziert:

```
Mon Jan  1 00:00:00 CET 2001
```

3.11.7 Text filtern

Der Text des Code-Abschnitts kann durch ein externes Programm gefiltert werden. Dies geschieht bei Angabe der Direktive `filter=COMMAND`. Hierbei ist `COMMAND` ein Kommando, das als Filter arbeitet, also von stdin liest und nach stdout schreibt.

```
%Code: filter="grep -v ^D"
A Dies
B ist
C ein
D Filter-
E Test
.
```

produziert:

```
A Dies
B ist
C ein
E Test
```

3.11.8 Text extrahieren

Kommt der Text aus einer Datei oder von einem Programmaufruf, möchte man u.U. nur einen Teil davon darstellen.

Mit der Direktive `extract` ist es möglich, den Text auf einen Teil zu reduzieren. Der Wert der Direktive ist ein Regulärer Ausdruck, der den gewünschten Teil "captured".

In einem längeren Codeabschnitt kann der interessante Teil zum Beispiel mit Markern wie "# beginünd "# end"gekennzeichnet sein:

```
%Code: extract="# begin\s+(.*)# end"
#!/usr/bin/env perl

use strict;
use warnings;
use 5.010;

# begin
my $max = '-inf';
for my $n (-100,-50,-10) {
    $max = $n if $n > $max;
}
# end
say $max;
__END__
-10
.
```

produziert:

```
my $max = '-inf';
for my $n (-100,-50,-10) {
    $max = $n if $n > $max;
}
```

3.11.9 Eigenschaften

```
%Code:
  exec=COMMAND
  extract=REGEX
  filter=COMMAND
  indent=BOOL
  load=FILE
  lang=LANGUAGE
  ln=N
TEXT
.
```

Im Falle der Direktiven `exec` und `load` entfällt der Text-Body `TEXT` und der abschließende Punkt.

Die Reihenfolge der Ausführung der Direktiven `exec`, `extract`, `filter` und `load` - so weit angegeben - ist:

1. `load` oder `exec` (nur alternativ möglich)
2. `filter`

3. `extract`

exec Führe das Kommando `COMMAND` aus und verwende dessen Ausgabe nach stdout und stderr als Text des Code-Blocks. Der Code Block hat in diesem Fall keinen Text-Body. Beginnt das Kommando mit `+/`, wird das Pluszeichen zum Pfad des Dokumentverzeichnisses expandiert.

extract Reduziere den Text auf einen Teil. Der Reguläre Ausdruck $regex hat Perl-Mächtigkeit und wird unter den Modifiern `s` (`.` matcht Zeilenumbrüche) und `m` (`^` und `$` matchen Zeilenanfang und -ende) interpretiert. Der Reguläre Ausdruck muss einen eingebetteten Klammerausdruck `(...)` enthalten. Dieser "captured" den gewünschten Teil.

filter Schicke den Text des Code-Blocks an Kommando COMMAND und ersetze ihn durch dessen Ausgabe. Das Kommando arbeitet als Filter, liest also von stdin und schreibt nach stdout.

indent (Default: *kontextabhängig*) Rücke den Text ein. Im Falle von Zeilennummern (`ln=N`) oder Syntax-Highlighting (`lang=LANGUAGE`) wird per Default *nicht* eingerückt. Sonst wird per Default eingerückt. Durch explizite Setzung des Attributs kann der jeweilige Default überschrieben werden.

load Lade Datei `FILE` und verwende dessen Inhalt als Text des Code-Blocks. Der Code Block hat in diesem Fall keinen Text-Body. Beginnt der Pfad der Datei mit `+/`, wird das Pluszeichen zum Pfad des Dokumentverzeichnisses expandiert.

lang Wende Syntax-Highlighting auf den Text an. Das Attribut `indent` wird auf den Default 0 gesetzt.

ln Setze an den Anfang der Zeilen eine Zeilennummer, wenn `N` $>$ `0`. Start-Zeilennummer ist `N`. Das Attribut `indent` wird auf den Default 0 gesetzt.

3.12 Verweis (Link)

Ein Verweis referenziert ein *internes* oder *externes* Ziel. Ein internes Ziel ist eine andere Stelle innerhalb desselben Dokuments. Auf diese Stelle wird beim Verfolgen des Verweises positioniert. Ein externes Ziel ist eine lokale Datei oder ein URL. Diese externe "Ressource" wird beim Folgen des Verweises aufgerufen.

Ein Verweis wird in den Text eines Sdoc-Dokuments durch ein L-Segment `L{TEXT}` gesetzt. Hierbei ist es gleichgültig, ob es sich um einen internen oder einen externen Verweis handelt. Der Unterschied besteht in der Art und Weise, wie der Verweis intern aufgelöst wird (siehe unten).

Der Link-Text `TEXT` wird im Zieldokument als Verweis kenntlich gemacht und kann in HTML oder PDF angeklickt werden. Der Text kann auch umbrochen sein, sich also über mehr als eine Zeile erstrecken. Dies hat auf die Link-Auflösung durch Sdoc keinen Einfluss, denn der Text wird zunächst in zwei Schritten *kanonisiert*:

1. Zeilenumbrüche werden durch Leerzeilen ersetzt.
2. Folgen von zwei und mehr Leerzeichen werden auf *ein* Leerzeichen reduziert.

Für die Ermittelung des Ziels aus dem L-Segment

```
... siehe Abschnitt L{Schachtelung
von  Segmenten} ...
```

wird also der kanonisierte Text

```
Schachelung von Segmenten
```

gebildet.

3.12.1 Interner Verweis

3.12.2 Eigenschaften

```
%Link:
  name=STRING
  file=PATH
  regex=REGEX
  url=URL
```

name Name der Link-Definiton.

file Pfad einer lokalen Datei.

regex Regex, der den internen Zielknoten identifiziert.

url URL eines externen Dokuments.

3.13 Datei laden (Include)

Durch einen Include-Block kann der Inhalt einer Datei zum Dokument hinzugefügt werden. Der geladene Sdoc-Quelltext wird an der Stelle zum Dokument hinzugefügt, an der der Include-Block steht.

```
%Include: load="+/sdoc-test-include.sdoc"
```

produziert:

Dies ist ein aus der Datei **sdoc-test-include.sdoc** inkludierter Sdoc-Quelltext.

3.13.1 Eigenschaften

```
%Include:
  load=PATH
```

load Lade Datei PATH und füge dessen Inhalt in das Dokument ein. Beginnt der Pfad mit `+/`, wird das Pluszeichen zum Pfad des Dokumentverzeichnisses expandiert.

3.14 Kommentar (Comment)

In den Quelltext eines Sdoc-Dokuments können Kommentarzeilen eingebettet werden. Eine Kommentarzeile beginnt mit einem Hash (#) am *Zeilenanfang*, eine Einrückung ist nicht erlaubt.

```
# Dies ist ein Kommentar, der in den Quelltext des Zielformats
# übertragen wird, aber im Zieldokument nicht auftaucht
```

Ist die Dokument-Eigenschaft `copyComments=1` gesetzt, werden die Sdoc-Kommentare in das Zielformat übertragen, sofern das Zielformat Kommentare unterstützt. Unterstützt das Zielformat keine Kommentare, entfallen sie.

In den Quelltext des LaTeX-Dokuments wird der Kommentar beispielsweise übertragen als

```
% Dies ist ein Kommentar, der in den Quelltext des Zielformats
% übertragen wird, aber im Zieldokument nicht auftaucht
```

Kommentare in das Zielformat zu übertragen kann z.B. nützlich sein, um eine Stelle im Quelltext des Zielformats zu finden, die aus einem bestimmten Sdoc-Konstrukt hervorgegangen ist, oder um Kennzeichen für eine Weiterverarbeitung zu setzen.

In Block-Syntax kann ein Kommentar auch geschrieben werden als

```
%Comment:
Dies ist ein Kommentar, der in den Quelltext des Zielformats
übertragen wird, aber im Zieldokument nicht auftaucht
.
```

3.14.1 Eigenschaften

```
%Comment:
TEXT
.
```

Ein Kommentar hat keine Eigenschaften.

3.15 Segment

3.15.1 A - Anker (anchor)

3.15.2 B - Fettschrift (bold)

Der von `B{}` eingefasste Text `TEXT` wird fett wiedergegeben.

Beispiel

```
Du machst mich B{sehr} wütend.
```

Du machst mich **sehr** wütend.

3.15.3 C - Festbreitenschrift (code)

Der von `C{}` eingefasste Text `TEXT` wird in Festbreitenschrift wiedergegeben.

Beispiel

```
Der Konstruktoraufruf C{$class->new()} instantiiert das Objekt.
```

Der Konstruktoraufruf `$class->new()` instantiiert das Objekt.

3.15.4 G - Inline-Grafik (graphic)

Eine Inline-Grafik, also eine Grafik, die im Textfluss erscheint, wird durch ein G-Segment `G{NAME}` erzeugt. Hierbei ist NAME der Name eines Grafik-Blocks (siehe Abschnitt Abbildung auf Seite 13, der die Grafik mit allen Detailangaben definiert.

Der Grafik-Block kann *irgendwo* innerhalb des Dokuments stehen: z.B. nach dem Paragraphen, der das L-Segment enthält, am Ende des Abschnitts oder am Anfang oder am Ende des Dokuments.

Beispiel

```
Dies ist eine G{Invader} Inline-Grafik.
```

```
%Graphic:
  name="Invader"
  file="+/sdoc-graphic-invader"
  scale=0.08
```

produziert:

Dies ist eine  Inline-Grafik.

3.15.5 I - Kursivschrift (italic)

Der von `I{}` eingefasste Text `TEXT` wird kursiv wiedergegeben.

Beispiel

```
Ich sagte, du sollst das I{nicht} essen.
```

Ich sagte, du sollst das *nicht* essen.

3.15.6 L - Verweis (link)

Beispiel: Interner Link

```
L{Dokument}
```

Dokument

Beispiel: Interner Link mit Seitenzahl

```
L{+Dokument}
```

Dokument auf Seite 8

Beispiel: Externer Link

```
L{http://fseitz.de/}
```

http://fseitz.de/

Beispiel: Link via Link-Definition

```
L{Google}

%Link:
  name="Google"
  url="http://google.com"
```

Google

3.15.7 M - Mathematische Formel (math)

Der von `M~TEXT~` eingefasste Text `TEXT` wird als mathematische Formel gesetzt. Da die Formel-Syntax von LaTeX verwendet wird und diese umfangreichen Gebrauch von geschweiften Klammern macht, ist das Begrenzungszeichen des Segments `M~~` die Tilde (~).

Beispiel

```
M~c=\sqrt{a^2+b^2}~
```

produziert:
$c = \sqrt{a^2 + b^2}$

3.15.8 Q - Anführungszeichen (quote)

Der von `Q{}` eingefasste Text `TEXT` wird in doppelte Anführungsstriche gesetzt. Dieses Konstrukt anstelle der doppelten Anführungsstriche auf der Tastatur (") zu verwenden ist für LaTeX wichtig, da die doppelten Anführungsstriche dort im Zusammenhang mit der Spracheinstellung `ngerman` oder `german` eine Sonderbedeutung haben und daher u.U. missinterpretiert werden.

Beispiel

```
Was heißt schon Q{richtig}?
```

Was heißt schon "richtig"?

3.15.9 Allgemeines

Ein Segment kann sich über mehrere Zeilen erstrecken.

Segment aufheben

Die Bedeutung eines Segments kann im Text (außerhalb von Segmenten) aufgehoben werden, indem dem Buchstabe ein Backslash vorangestellt wird.

```
\B{xxx}
```

B{xxx}

Geschweifte Klammern innerhalb eines Segments

Kommen innerhalb eines Segments geschweifte Klammern vor, müssen diese mit einem Backslash geschützt werden.

```
C{$x->\{'name'\}}
```

`$x->{'name'}`

Auf diese Weise kann auch die Bedeutung eines Segments innerhalb eines anderen Segments aufgehoben werden. Das Voranstellen eines Backslash vor den Segment-Buschstaben ist hier nicht der richtige Weg, da die geschweiften Klammern dann ungeschützt sind.

```
C{C\{TEXT\}}
```

`C{TEXT}`

Schachtelung von Segmenten

Segmente können geschachtelt werden. Es hängt allerdings vom Zielformat ab, ob solche Konstuktionen ins Zielformat umgesetzt werden können.

```
B{So geht es I{auch}!}
```

So geht es *auch*!

```
I{Näheres in Abschnitt L{Dokument}}
```

Näheres in Abschnitt Dokument

3.16 Beispiel in Block-Syntax

Hier das Beispiel aus Abschnitt Beispiel/Sdoc auf Seite 5 ohne Wiki-Syntax, also ausschließlich in Block-Syntax. Der Quelltext verlängert sich dadurch von 43 auf 70 Zeilen.

```
%Document:
  title="Ein Beispieldokument"
  author="Nico Laus"
  date="today"

%Section:
  level=1
  title="Einleitung"

%Paragraph:
Dies ist eine I{Einleitung}. Mit wenig C{Aufwand} erstellen wir in
Sdoc ein B{Dokument}. Als Textauszeichnungen sind auch Kombinationen
wie B{I{fett und kursiv}} oder C{I{monospace und kursiv}} möglich.
.

%Section:
  level=2
  title="Ein Beispiel"

%Paragraph:
Mit dem Satz des Pythagoras M~a^2+b^2=c^2~ berechnen wir die Länge der
Hypotenuse: M~c=\sqrt{a^2+b^2}~.
.

%Graphic:
  file="+/sdoc-graphic-illusion"
  align="center"
  scale=0.3

%Paragraph:
Programm-Quelltexte können wir mit Syntax-Highlighting verschönern:
.

%Code: lang=Perl ln=1 indent=1
  open(my $fh, 'ls -l |') or die "open failed ($!)\n";
  while (<$fh>) {
      s/\n/<BR>\n/;
      print;
  }
  close($fh) or die "close failed ($!)\n";
.

%Paragraph:
Auf eine andere Stelle im Dokument zu verweisen ist einfach. Hier ist
ein Verweis auf die L{Einleitung}.
.

%List:
  listType="unordered"
```

```
%Item: key="*"
  %Paragraph:
  Apfel
  .
%Item: key="*"
  %Paragraph:
  Birne
  .
%Item: key="*"
  %Paragraph:
  Pflaume
  .

%Paragraph:
Das ist eine einfache Aufzählung. Aufzählungen können beliebig
geschachtelt werden.
.

%Comment:
# eof
.
```

4 Sonstiges

4.1 Reservierte Zeichen

Damit Zeichen mit einer Sonderbedeutung im Text richtig wiedergegeben werden, müssen diese - je nach Zielformat - gewandelt werden.

4.1.1 LaTeX

In LaTeX gibt es besonders viele reservierte Zeichen. Die folgende Liste umfasst alle Zeichen, die bei LaTeX/PDF als Zielformat im Fließtext besonders behandelt werden. Sie sollten in einem PDF korrekt dargestellt werden, einmal ohne und einmal mit einem Leerzeichen vor dem betreffenden Zeichen.

Backslash: \xxx, \ xxx

Größer: >0, > 0

Kleiner: <0, < 0

Tilde: ~x, ~ x

Dollar: $x, $ x

Unterstrich: _x, _ x

Prozent: %x, % x

Geschweifte Klammern: {x, { x, }x, } x

Caret: ^x, ^ x

Hash: #x, # x

Kaufmanns-Und: &x, & x

Pipe: |x, | x
Latex-Symbol: LaTeX, LaTeX x
Tex-Symbol: TeX, TeX x

4.2 Warnungen

Beim Parsen und Übersetzen eines Sdoc-Dokuments können folgende Warnungen auftreten. Diese werden nach stderr geschrieben.

Appendix flag allowed on toplevel sections only Ein Unterabschnitt ist mit '+' als Appendix gekennzeichnet worden. Nur Hauptabschnitte können als Appendix gekennzeichnet werden.

Attribute "KEY"does not exist Es wurde ein Block-Attribut zu setzen versucht, das nicht existiert.

Can't resolve link uniquely: L{TEXT} Ein interner Link konnte nicht eindeutig aufgelöst werden. Zwei oder mehr gleichwertige Link-Ziele wurden ermittelt.

Can't resolve link: L{TEXT} Ein interner Link konnte nicht aufgelöst werden. Es wurde kein Link-Ziel gefunden.

Graphic not found: G{NAME} Der Grafik-Knoten mit dem Namen `NAME`, der von einem G-Segment referenziert wird, konnte nicht gefunden werden.

Graphic node not used: name="NAME" Der Grafik-Knoten mit dem Namen `NAME`, der als Definition deklariert ist, wird von keinem G-Segment genutzt.

Link node not used: name="ungenutzt" Es wurde per %Link: ein Link definiert, der nirgendwo innerhalb des Dokuments genutzt wird.

More than one anchor: A{TEXT} In einem Block wurde mehr als ein Anker gesetzt. Nur der erste Anker innerhalb eines Blocks wird verwendet.

Node does not allow anchors: A{TEXT} In einem Block, der keinen Anker zulässt, wurde ein Anker zu setzen versucht.

Node does not allow formulas: M~TEXT~ In einem Block, der keine mathematische Formel zulässt, wurde eine mathematische Formel zu setzen versucht.

Node does not allow inline graphics: G{NAME} In einem Block, der keine Inline-Grafik (G-Segment) zulässt, wurde eine Inline-Grafik zu setzen versucht.

Node does not allow links: L{TEXT} In einem Block, der keine Links zulässt, wurde ein Link zu setzen versucht.

4.3 Dieses Dokument übersetzen

Übersetzung nach PDF:

```
$ sdoc pdf sdoc-manual.sdoc --output=sdoc-manual.pdf
```

Vorschau auf das PDF-Dokument:

```
$ sdoc pdf sdoc-manual.sdoc --preview
```

5 Programmierung

5.1 Einen neuen Knoten-Typ definieren

Um einen neuen Knoten-Typ `TYPE` zur Sdoc-Sprache hinzuzufügen, gehen wir in folgenden Schritten vor:

1. Wir erweitern die Methode `LineProcessor::nextType()` um die Erkennung des neuen Knoten-Typs, also die Erkennung seines Markup im Sdoc-Quelltext. Besitzt der Knoten-Typ kein eigenes Markup, sondern wird er allein in Block-Syntax notiert (wie z.B. der Dokument-Knoten), entfällt dieser Schritt.
2. Wir tragen eine Instanz des neuen Knoten-Typs in ein Sdoc-Dokument ein und versuchen dieses zu parsen mit

   ```
   $ sdoc validate DOCUMENT.sdoc
   ```

   Der neue Knoten-Typ `TYPE` sollte zwar erkannt werden, aber die Übersetzung sollte gleichwohl fehlschlagen mit der Fehlermeldung:

   ```
   Exception:
       SDOC-00002: Unknown node type
   Type:
       <TYPE>
   ...
   ```

   Schlägt die Übersetzung *nicht* fehl, wurde der neue Knoten-Typ nicht erkannt. Grund für den Fehler ist, dass noch keine Knoten-Klasse für den neuen Typ implementiert wurde. Dies machen wir im nächsten Schritt.
3. Wir suchen uns eine ähnliche Knoten-Klasse aus den `Sdoc::Node`-Klassen aus, kopieren sie zur einer neuen Klasse `Sdoc::Node::TYPE` und schreiben diese geeignet um. Indem wir die Regressionstestfälle mit anpassen, können wir sicherstellen, dass das Parsen und Instantiieren des neuen Knoten-Typs funktioniert.
4. Wir fügen die neue Knoten-Klasse `Sdoc::Node::TYPE` zur Import-Liste der Klasse `Sdoc::Document` hinzu. Vergessen wir dies, erhalten wir im nächsten Schritt weiterhin die Fehlermeldung aus Schritt 2.
5. Wir überprüfen, dass bei der Übersetzung in die Zielformate das richtige Ergebnis produziert wird. Hierzu übersetzen wir das Dokument nacheinander in die einzelnen Zielformate und sehen uns das jeweilige Ergebnis an:

   ```
   $ sdoc latex --preview DOCUMENT.sdoc
   $ sdoc pdf --preview DOCUMENT.sdoc
   ```

   Die Option `--preview` ist hierbei praktisch, da das Dokument dann in dem jeweiligen Format unmittelbar angezeigt wird.

5.2 Einen neuen Segment-Typ definieren

Um einen neuen Segment-Typ `T` zur Sdoc-Sprache hinzuzufügen, gehen wir in folgenden Schritten vor:

1. Wir erweitern die Methode `<%Sdoc::LineProcessor>%::parseSegments()` um die Vorbehandlung des neuen Segment-Typs, also dessen Erkennung und Wandlung in die interne Segment-Repräsentation. Hierzu suchen wir den Code eines ähnlichen, bereits existierenden Segment-Typs, kopieren diesen und passen ihn an.

2. Wir erweitern die Methode(n) <%Sdoc::Node>%::FORMATText() (z.B. latexText()), so dass sie den FORMAT-spezifischen Code neuen Segment-Typ erzeugen.

5.3 Knoten-Eigenschaft hinzufügen

Eine Eigenschaft ATTRIBUTE wird zur Knoten-Klasse Sdoc::Node::TYPE in zwei Schritten hinzugefügt:

1. Wir beschreiben die Eigenschaft in der Klassen-Dokumentation.
2. Wir fügen die Eigenschaft und ihren Defaultwert zum Konstruktor hinzu (bei Aufruf des Basisklassen- Konstruktors).

Anschließend kann die Eigenschaft im Sdoc-Quelltext gesetzt werden mit:

```
%TYPE:
  ATTRIBUTE="VALUE"
```

In den Methoden der Klasse kann die Eigenschaft abgefragt werden mit:

```
$val = $node->ATTRIBUTE;
```

Beispiel: Eigenschaft `language` der Klasse Sdoc::Node::Document.