

Sdoc

Frank Seitz, <http://fseitz.de/>

20. Januar 2018

Inhaltsverzeichnis

1	Einleitung	3
2	Beispiel	4
2.1	Sdoc	4
2.2	PDF	5
3	Syntax	6
3.1	Allgemeines	6
3.1.1	Kommentare	6
3.1.2	Zeilenfortsetzungen	6
3.1.3	Leerzeilen	6
3.1.4	Pfad-Expansion	6
3.1.5	Block-Syntax und Wiki-Syntax	6
3.2	Dokument (Document)	7
3.2.1	Eigenschaften	7
3.3	Inhaltsverzeichnis (TableOfContents)	8
3.3.1	Eigenschaften	8
3.4	Abschnitt (Section)	8
3.4.1	Unterabschnitte vom Inhaltsverzeichnis ausschließen	9
3.4.2	Appendix	9
3.4.3	Eigenschaften	9
3.5	Zwischenüberschrift (BridgeHead)	10
3.5.1	Eigenschaften	10
3.6	Seitenumbruch (PageBreak)	10
3.6.1	Eigenschaften	11
3.7	Paragraph (Paragraph)	11
3.7.1	Eigenschaften	11
3.8	Abbildung (Graphic)	11
3.8.1	Eigenschaften	12
3.9	Aufzählung (List)	12
3.9.1	Punktliste	12
3.9.2	Numerierungsliste	12
3.9.3	Definitionsliste	13
3.9.4	Verschachtelung von Listen	13

3.9.5	Eigenschaften	14
3.10	Code (Code)	14
3.10.1	Einrückung mit Leerraum	14
3.10.2	In Block-Syntax	14
3.10.3	Syntax-Highlighting	15
3.10.4	Zeilennummern	15
3.10.5	Text aus Datei laden	16
3.10.6	Text von Programmaufruf	16
3.10.7	Text filtern	16
3.10.8	Text extrahieren	16
3.10.9	Eigenschaften	17
3.11	Verweis (Link)	18
3.11.1	Eigenschaften	18
3.12	Kommentar (Comment)	18
3.12.1	Eigenschaften	19
3.13	Segment	19
3.13.1	A - Anker (anchor)	19
3.13.2	B - Fettschrift (bold)	19
3.13.3	C - Festbreitenschrift (code)	19
3.13.4	G - Inline-Grafik (graphic)	19
3.13.5	I - Kursivschrift (italic)	20
3.13.6	L - Verweis (link)	20
3.13.7	M - Mathematische Formel (math)	20
3.13.8	Q - Anführungszeichen (quote)	20
3.14	Beispiel in Block-Syntax	21
4	Sonstiges	22
4.1	Reservierte Zeichen	22
4.1.1	LaTeX	22
4.2	Warnungen	23
4.3	Dieses Dokument übersetzen	23
5	Programmierung	24
5.1	Einen neuen Knoten-Typ definieren	24
5.2	Einen neuen Segment-Typ definieren	24

1 Einleitung

Dieses Dokument beschreibt die Syntax und den Gebrauch der Auszeichnungssprache 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. Es kann als Beispiel herangezogen werden. Wie es übersetzt werden kann, ist in Abschnitt [Dieses Dokument übersetzen](#) auf Seite 23 beschrieben.

2 Beispiel

2.1 Sdoc

```
1 %Document:
2   title="Ein Beispieldokument"
3   author="Nico Laus"
4   date="today"
5
6 = Einleitung
7
8 Dies ist eine I{Einleitung}. Mit wenig C{Aufwand} erstellen wir in
9 Sdoc ein B{Dokument}. Als Textauszeichnungen sind auch Kombinationen
10 wie B{I{fett und kursiv}} oder C{I{monospace und kursiv}} möglich.
11
12 == Ein Beispiel
13
14 Mit dem Satz des Pythagoras  $M^2 + b^2 = c^2$  berechnen wir die Länge der
15 Hypotenuse:  $M^2 = \sqrt{a^2 + b^2}$ .
16
17 %Graphic:
18   file="+/sdoc-graphic-illusion"
19   align="center"
20   scale=0.3
21
22 Programm-Quelltexte können wir mit Syntax-Highlighting verschönern:
23
24 %Code: lang=Perl ln=1 indent=1
25   open(my $fh, 'ls -l |') or die "open failed ($!)\n";
26   while (<$fh>) {
27     s/\n/<BR>\n/;
28     print;
29   }
30   close($fh) or die "close failed ($!)\n";
31 .
32
33 Auf eine andere Stelle im Dokument zu verweisen ist einfach. Hier ist
34 ein Verweis auf die L{Einleitung}.
35
36 * Apfel
37 * Birne
38 * Pflaume
39
40 Das ist eine einfache Aufzählung. Aufzählungen können beliebig
41 geschachtelt werden.
42
43 # eof
```

2.2 PDF

Ein Beispieldokument

Nico Laus

20. Januar 2018

Inhaltsverzeichnis

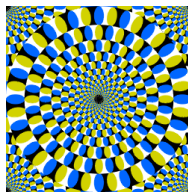
1 Einleitung	1
1.1 Ein Beispiel	1

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 18.

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 21 ist das Beispiel aus Abschnitt [Beispiel/Sdoc](#) auf Seite 4 vollständig in Block-Syntax wiedergegeben.

3.2 Dokument (Document)

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

```
%Document:  
  author=STRING  
  date=STRING  
  documentType=article|report|book  
  fontSize=10pt|11pt|12pt  
  geometry=STRING  
  language=german|english  
  sectionNumberDepth=0|1|2|3|4|5  
  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 Eigenschaften

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.

documentType (Default: article) Der Typ des Dokuments. Mögliche Werte: 'article', 'report', 'book'. Unter LaTeX wird der Wert übersetzt in die Dokumentklassen scrartcl, scrreprt, scrbook.

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

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

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.

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

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 platziert, 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 7). 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 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.

3.4.3 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 1
==* Zwischenüberschrift Stufe 2
===* Zwischenüberschrift Stufe 3
====* Zwischenüberschrift Stufe 4
```

produziert:

Zwischenüberschrift Stufe 1

Zwischenüberschrift Stufe 2

Zwischenüberschrift Stufe 3

Zwischenüberschrift Stufe 4

3.5.1 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
Paragraph.
```

```
---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 Eigenschaften

`%Paragraph:`

`TEXT`

`.`

Ein Paragraph hat keine Eigenschaften.

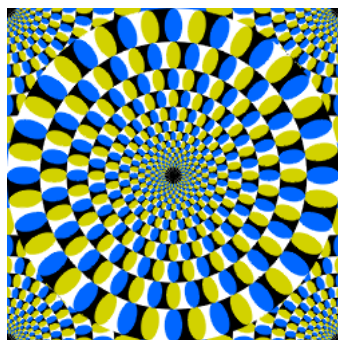
3.8 Abbildung (Graphic)

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 [19](#).

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 Dokumentverzeichnis 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 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 Code (Code)

Ein Code-Abschnitt ist ein literaler Textblock, der, abgesehen von Einrückung, Farbgebung oder Zeilennummerierung, im Zielformat exakt so wiedergegeben wird wie er in der Quelle steht, einschließlich Leerzeichen und Zeilenumbrüchen.

Es gibt zwei Möglichkeiten einen literalen Textblock anzugeben.

3.10.1 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";
```

3.10.2 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
```

3.10.3 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);
}
```

3.10.4 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;
```

3.10.5 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
```

3.10.6 Text von Programmaufruf

Der Text des Code-Abschnitts 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
```

3.10.7 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
```

3.10.8 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 "# beginnend" "# 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;
}
}
```

3.10.9 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** - sofern angegeben - ist:

1. **load** oder **exec** (beide gemeinsam anzugeben ergibt keinen Sinn)
2. **filter**
3. **extract**
4. 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.
5. 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.

6. 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`.
7. (Default: *kontextabhängig*) Rücke den Text ein. Im Falle von Zeilennummern (`1n=N`) oder Syntax-Highlighting (`1ang=LANGUAGE`) wird per Default *nicht* eingerückt. Sonst wird per Default eingerückt. Durch explizite Setzung des Attributs kann der jeweilige Default überschrieben werden.
8. 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 Dokumentverzeichnis expandiert.
9. Wende Syntax-Highlighting auf den Text an. Das Attribut `indent` wird auf den Default 0 gesetzt.
10. Setze an den Anfang der Zeilen eine Zeilennummer, wenn `N > 0`. Start-Zeilennummer ist `N`. Das Attribut `indent` wird auf den Default 0 gesetzt.

3.11 Verweis (Link)

Ein Verweis referenziert ein *internes* oder *externes* Ziel. Ein internes Ziel ist ein Knoten innerhalb desselben Dokuments. Ein externes Ziel ist eine Datei oder ein URL.

Ein Verweis wird in den Text eines Sdoc-Dokuments durch ein L-Segment `L{TEXT}` gesetzt.

3.11.1 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.

3.12 Kommentar (Comment)

In den Quelltext eines Sdoc-Dokuments können Kommentarzeilen eingebettet werden. Diese werden, sofern das Zielformat dies unterstützt, als Kommentare dorthin übertragen. Unterstützt das Zielformat keine Kommentare, entfallen sie.

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
```

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
```

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  
.
```

3.12.1 Eigenschaften

```
%Comment:  
TEXT  
.
```

Ein Kommentar hat keine Eigenschaften.

3.13 Segment

3.13.1 A - Anker (anchor)

3.13.2 B - Fettschrift (bold)

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

Beispiel

Du machst mich `B{sehr}` wütend.

Du machst mich **sehr** wütend.

3.13.3 C - Festbreitenschrift (code)

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

Beispiel

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

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

3.13.4 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 [11](#), 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.

3.13.5 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.

3.13.6 L - Verweis (link)

3.13.7 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:

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

3.13.8 Q - Anführungszeichen (quote)

Der von Q{TEXT} 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"?

3.14 Beispiel in Block-Syntax

Hier das Beispiel aus Abschnitt [Beispiel/Sdoc](#) auf Seite 4 ohne Wiki-Syntax, also ausschließlich in Block-Syntax. Der Quelltext verlängert sich dadurch von 43 auf 70 Zeilen.

```
1 %Document:
2   title="Ein Beispieldokument"
3   author="Nico Laus"
4   date="today"
5
6 %Section:
7   level=1
8   title="Einleitung"
9
10 %Paragraph:
11 Dies ist eine I{Einleitung}. Mit wenig C{Aufwand} erstellen wir in
12 Sdoc ein B{Dokument}. Als Textauszeichnungen sind auch Kombinationen
13 wie B{I{fett und kursiv}} oder C{I{monospace und kursiv}} möglich.
14 .
15
16 %Section:
17   level=2
18   title="Ein Beispiel"
19
20 %Paragraph:
21 Mit dem Satz des Pythagoras  $M^2a^2+b^2=c^2$  berechnen wir die Länge der
22 Hypotenuse:  $M^2c=\sqrt{a^2+b^2}$ .
23 .
24
25 %Graphic:
26   file="+/sdoc-graphic-illusion"
27   align="center"
28   scale=0.3
29
30 %Paragraph:
31 Programm-Quelltexte können wir mit Syntax-Highlighting verschönern:
32 .
33
34 %Code: lang=Perl ln=1 indent=1
35   open(my $fh, 'ls -l |') or die "open failed ($!)\n";
36   while (<$fh>) {
37     s/\n/<BR>\n/;
38     print;
39   }
40   close($fh) or die "close failed ($!)\n";
41 .
42
43 %Paragraph:
44 Auf eine andere Stelle im Dokument zu verweisen ist einfach. Hier ist
45 ein Verweis auf die L{Einleitung}.
46 .
47
48 %List:
49   listType="unordered"
```

```

50 %Item: key="*"
51 %Paragraph:
52 Apfel
53 .
54 %Item: key="*"
55 %Paragraph:
56 Birne
57 .
58 %Item: key="*"
59 %Paragraph:
60 Pflaume
61 .
62
63 %Paragraph:
64 Das ist eine einfache Aufzählung. Aufzählungen können beliebig
65 geschachtelt werden.
66 .
67
68 %Comment:
69 # eof
70 .

```

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

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 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.