

# Sdoc Handbuch

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

2. Dezember 2021

## Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>5</b>
<b>2. Beispiel</b>	<b>6</b>
2.1. Sdoc Quelltext . . . . .	6
2.2. PDF . . . . .	7
2.3. HTML . . . . .	8
<b>3. Dynamische Dokumente</b>	<b>9</b>
3.1. Beispiel . . . . .	9
<b>4. Syntax</b>	<b>10</b>
4.1. Allgemeines . . . . .	10
4.1.1. Kommentare . . . . .	10
4.1.2. Zeilenfortsetzungen . . . . .	10
4.1.3. Leerzeilen . . . . .	10
4.1.4. Pfad-Expansion . . . . .	10
4.1.5. Block-Syntax und Wiki-Syntax . . . . .	11
4.2. Dokument (Document) . . . . .	12
4.2.1. Segmente . . . . .	12
4.2.2. Eigenschaften . . . . .	12
4.3. Inhaltsverzeichnis (TableOfContents) . . . . .	15
4.3.1. Eigenschaften . . . . .	15
4.4. Abschnitt (Section) . . . . .	16
4.4.1. Übergeordnete Ebenen . . . . .	16
4.4.2. Appendix . . . . .	16
4.4.3. Unterabschnitte vom Inhaltsverzeichnis ausschließen . . . . .	16
4.4.4. Segmente . . . . .	17
4.4.5. Eigenschaften . . . . .	17
4.5. Zwischenüberschrift (BridgeHead) . . . . .	18
4.5.1. Segmente . . . . .	18
4.5.2. Eigenschaften . . . . .	18
4.6. Seitenumbruch (PageBreak) . . . . .	19
4.6.1. Eigenschaften . . . . .	19
4.7. Paragraph (Paragraph) . . . . .	20
4.7.1. Segmente . . . . .	20
4.7.2. Eigenschaften . . . . .	20
4.8. Abbildung (Graphic) . . . . .	21
4.8.1. Segmente . . . . .	22

4.8.2. Eigenschaften . . . . .	22
4.9. Aufzählung (List) . . . . .	24
4.9.1. Punktliste . . . . .	24
4.9.2. Nummerierungsliste . . . . .	24
4.9.3. Definitionsliste . . . . .	24
4.9.4. Verschachtelung von Listen . . . . .	26
4.9.5. Segmente . . . . .	26
4.9.6. Eigenschaften . . . . .	26
4.10. Tabelle (Table) . . . . .	28
4.10.1. Definition . . . . .	28
4.10.2. Ausrichtung . . . . .	28
4.10.3. Ohne Titelzeile . . . . .	28
4.10.4. Mehrzeilige Daten . . . . .	29
4.10.5. Beschriftung . . . . .	30
4.10.6. Titelfarbe . . . . .	30
4.10.7. Linien . . . . .	31
4.10.8. Segmente . . . . .	32
4.10.9. Eigenschaften . . . . .	33
4.10.10 L <sup>A</sup> T <sub>E</sub> X Setup . . . . .	34
4.11. Code (Code) . . . . .	35
4.11.1. Einrückung mit Leerraum . . . . .	35
4.11.2. In Block-Syntax . . . . .	35
4.11.3. Syntax-Highlighting . . . . .	35
4.11.4. Zeilennummern . . . . .	36
4.11.5. Text aus Datei laden . . . . .	36
4.11.6. Text von Programmaufruf . . . . .	37
4.11.7. Text filtern . . . . .	37
4.11.8. Text extrahieren . . . . .	37
4.11.9. Eigenschaften . . . . .	38
4.12. Verweis (Link) . . . . .	40
4.12.1. Interner Verweis . . . . .	40
4.12.2. Eigenschaften . . . . .	40
4.13. Datei laden (Include) . . . . .	41
4.13.1. Eigenschaften . . . . .	41
4.14. Zitat (Quote) . . . . .	42
4.14.1. Eigenschaften . . . . .	42
4.15. Kommentar (Comment) . . . . .	43
4.15.1. #-Kommentare . . . . .	43
4.15.2. %-Kommentare . . . . .	43
4.15.3. Eigenschaften eines %-Kommentars . . . . .	43
4.16. Zielformatblock (Format) . . . . .	44
4.16.1. Eigenschaften . . . . .	44
4.17. Stilvorlage (Style) . . . . .	45
4.17.1. Eigenschaften . . . . .	45
4.18. Selbstdefinierte Textauszeichnung (Segment) . . . . .	46
4.18.1. Beispiel . . . . .	46
4.18.2. Eigenschaften . . . . .	46
4.19. Nachverarbeitung (PostProcessor) . . . . .	47
4.19.1. Eigenschaften . . . . .	47
4.20. Segment . . . . .	48
4.20.1. A - Anker (anchor) . . . . .	48
4.20.2. B - Fettschrift (bold) . . . . .	48
4.20.3. C - Festbreitenschrift (code) . . . . .	48

4.20.4. G - Inline-Grafik (graphic) . . . . .	48
4.20.5. I - Kursivschrift (italic) . . . . .	48
4.20.6. L - Verweis (link) . . . . .	49
4.20.7. M - Mathematische Formel (math) . . . . .	49
4.20.8. N - Zeilenumbruch (newline) . . . . .	49
4.20.9. Q - Anführungszeichen (quote) . . . . .	50
4.20.10 S - Selbstdefinierte Textauszeichnung . . . . .	50
4.20.11 Allgemeines . . . . .	50
4.21. Beispiel in Block-Syntax . . . . .	52
<b>5. Einrückung von Blöcken</b>	<b>54</b>
5.1. Ohne Einrückung . . . . .	54
5.2. Mit Einrückung . . . . .	55
5.3. Sonderfall Listen . . . . .	56
<b>6. Verweise</b>	<b>57</b>
6.1. Einen Verweis setzen . . . . .	57
6.2. Referenzierbare Objekte . . . . .	57
6.3. Darstellung eines Verweises . . . . .	57
6.4. Verweis-Attribute . . . . .	58
6.4.1. + . . . . .	58
<b>7. Formate</b>	<b>59</b>
7.1. HTML/CSS . . . . .	59
7.1.1. BridgeHead . . . . .	59
7.1.2. Code (FIXME: Selektoren und Beispiele ergänzen) . . . . .	59
7.1.3. Comment . . . . .	60
7.1.4. Document . . . . .	61
7.1.5. (Graphic) . . . . .	61
7.1.6. Include . . . . .	62
7.1.7. Link . . . . .	62
7.1.8. List/Item . . . . .	62
7.1.9. PageBreak . . . . .	63
7.1.10. Paragraph . . . . .	63
7.1.11. Section . . . . .	63
7.1.12. Quote . . . . .	64
7.1.13. Style . . . . .	64
7.1.14. (Table) . . . . .	64
7.1.15. TableOfContents . . . . .	64
7.2. L <sup>A</sup> T <sub>E</sub> X . . . . .	66
7.2.1. Reservierte Zeichen . . . . .	66
7.2.2. Paket <code>array</code> . . . . .	66
7.2.3. Paket <code>babel</code> . . . . .	66
7.2.4. Paket <code>caption</code> . . . . .	66
7.2.5. Paket <code>float</code> . . . . .	66
7.2.6. Paket <code>fontenc</code> . . . . .	67
7.2.7. Paket <code>geometry</code> . . . . .	67
7.2.8. Paket <code>enumitem</code> . . . . .	67
7.2.9. Paket <code>graphicx</code> . . . . .	67
7.2.10. Paket <code>hyperref</code> . . . . .	67
7.2.11. Paket <code>inputenc</code> . . . . .	67
7.2.12. Paket <code>lmodern</code> . . . . .	67
7.2.13. Paket <code>longtable</code> . . . . .	67
7.2.14. Paket <code>makecell</code> . . . . .	67

7.2.15. Paket <code>microtype</code> . . . . .	67
7.2.16. Paket <code>minted</code> . . . . .	67
7.2.17. Paket <code>quoting</code> . . . . .	68
7.2.18. Paket <code>scrlayer-scrpage</code> . . . . .	68
7.2.19. Paket <code>showframe</code> . . . . .	68
7.2.20. Paket <code>varioref</code> . . . . .	68
7.2.21. Paket <code>xcolor</code> . . . . .	68
7.3. MediaWiki . . . . .	69
7.3.1. CSS . . . . .	69
7.3.2. Listen . . . . .	69
7.3.3. Einschränkungen . . . . .	69
<b>8. Sonstiges</b>	<b>70</b>
8.1. Warnungen . . . . .	70
8.2. Dieses Dokument übersetzen . . . . .	70
<b>9. Programmierung</b>	<b>71</b>
9.1. Einen neuen Knoten-Typ definieren . . . . .	71
9.2. Einen neuen Segment-Typ definieren . . . . .	71
9.3. Knoten-Eigenschaft hinzufügen . . . . .	72
9.4. Nutzer/Konfiguration/Knoten-Eigenschaft hinzufügen . . . . .	72
<b>A. Code Styles</b>	<b>73</b>
A.1. Übersicht . . . . .	73

## 1. Einleitung

Dieses Dokument beschreibt die Syntax und den Gebrauch der Sprache Sdoc. Sdoc ist eine leichtgewichtige Auszeichnungssprache zum Verfassen von Dokumenten. Ein Dokument, das in Sdoc geschrieben wurde, 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 gedacht. Ihr Aussehen kann via CSS gestaltet werden.

Die Druck-Fassung wird von L<sup>A</sup>T<sub>E</sub>X 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 selbst in Sdoc geschrieben. Wie es übersetzt, angezeigt und verarbeitet wird, ist in Abschnitt [8.2 - Dieses Dokument übersetzen](#) auf Seite [70](#) beschrieben.

## 2. Beispiel

### 2.1. Sdoc Quelltext

```
1 %Document:
2   title="Ein Beispieldokument"
3   author="Nico Laus"
4   date="today"
5   indentMode=1
6
7 = Einleitung
8
9 Dies ist eine I{Einleitung}. Mit wenig C{Aufwand} erstellen wir in
10 Sdoc ein B{Dokument}. Als Textauszeichnungen sind auch Kombinationen
11 wie B{I{fett und kursiv}} möglich.
12
13 == Ein Beispiel
14
15 Mit dem Satz des Pythagoras  $M^2 = a^2 + b^2$  berechnen wir die Länge
16 der Hypotenuse:  $M = \sqrt{a^2 + b^2}$ .
17
18 %Graphic:
19   file="+/sdoc-graphic-illusion"
20   caption="Illusion"
21   latexAlign="center"
22   width=100
23   height=100
24   border=1
25
26 Programm-Quelltexte können wir mit Q{Syntax-Highlighting} darstellen:
27
28 %Code: lang=Perl ln=1
29   open(my $fh, 'ls -l |') or die "open failed ($!)\n";
30   while (<$fh>) {
31     s/\n/<BR>\n/;
32     print;
33   }
34   close($fh) or die "close failed ($!)\n";
35 .
36
37 Auf eine andere Stelle im Dokument zu verweisen ist einfach. Hier ist
38 ein Verweis auf die L{Einleitung}.
39
40 * Apfel
41 * Birne
42 * Pflaume
43
44 Das ist eine einfache Aufzählung. Aufzählungen können beliebig
45 geschachtelt werden.
46
47 # eof
```

## 2.2. PDF

```
$ sdoc pdf sdoc-example.sdoc
```

# Ein Beispieldokument

Nico Laus

13. Juni 2020

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
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*** möglich.

### 1.1 Ein Beispiel

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

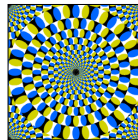


Abbildung 1: Illusion

Programm-Quelltexte können wir mit “Syntax-Highlighting” darstellen:

```
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](#) auf dieser Seite.

- Apfel
- Birne
- Pflaume

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

## 2.3. HTML

```
$ sdoc html sdoc-example.sdoc
```

# Ein Beispieldokument

Nico Laus  
13. Juni 2020

## Inhaltsverzeichnis

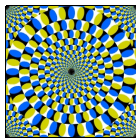
- 1 [Einleitung](#)
- 1.1 [Ein Beispiel](#)

## 1 Einleitung

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

### 1.1 Ein Beispiel

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



**Abbildung 1:** Illusion

Programm-Quelltexte können wir mit "Syntax-Highlighting" darstellen:

```
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. Dynamische Dokumente

Ein Designziel von Sdoc ist es, das dynamische Erzeugen von Dokumenten oder Dokumentteilen zu ermöglichen, z.B. das Einbinden von sich (laufend ändernden) Quelltexten. Das Konzept hierfür ist einfach:

- variable Dokument-Anteile werden per `%Include: file=FILE` geladen
- die Dokument-Anteile werden vorab durch Aufruf eines Programms erzeugt

Das Programm (Präprozessor) wird einfach neben der Dokumentdatei abgelegt. Das Programm hat den gleichen Grundnamen wie die Dokumentdatei, aber mit der Endung `.srun` (wenn die Dokumentdatei die Endung `.sdoc` hat) oder `.srun3` (wenn die Dokumentdatei die Endung `.sdoc3` hat). Beim Übersetzen wird das Programm automatisch vorab aufgerufen.

### 3.1. Beispiel

```
$ ls
mein-dokument.sdoc
mein-dokument.srun
```

Der Aufruf `sdoc mein-dokument.sdoc` führt als erstes das Programm `mein-dokument.srun` aus, bevor die Dokumentdatei `mein-dokument.sdoc` prozessiert wird. Das Programm erhält das Verzeichnis des Dokuments als aktuelles Arbeitsverzeichnis.

Dasselbe mit alternativer Dateiendung:

```
mein-dokument.sdoc3
mein-dokument.srun3
```

# 4. Syntax

## 4.1. Allgemeines

### 4.1.1. Kommentare

In den Quelltext des Dokuments können Kommentarzeilen eingebettet werden. Diese werden nicht mit gesetzt. Eine Kommentarzeile beginnt mit einem (#) am *Zeilenanfang*. Eine Einrückung ist nicht erlaubt.

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

Daneben gibt es eine zweite Art von Kommentarzeile, die *als Kommentar* in das Zieldokument übertragen wird (sofern das Zielformat Kommentare unterstützt). Diese Art von Kommentarzeile beginnt mit Prozent + Leerzeichen (%) am Zeilenanfang.

```
% Dies ist ein Kommentar, der im Quelltext des Zieldokuments erscheint
```

Weitere Information zu Kommentaren siehe Abschnitt [4.15 - Kommentar \(Comment\)](#) auf Seite [43](#).

### 4.1.2. 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.

### 4.1.3. 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.

### 4.1.4. 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.

#### 4.1.5. 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 [4.21 - Beispiel in Block-Syntax](#) auf Seite [52](#) ist das Beispiel aus Abschnitt [2.1 - Sdoc Quelltext](#) auf Seite [6](#) vollständig in Block-Syntax wiedergegeben.

## 4.2. Dokument (Document)

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

```
%Document:
  anchor=STRING
  author=STRING
  codeStyle=STYLE
  copyComments=BOOL
  cssPrefix=NAME
  date=STRING
  htmlDocumentStyle=NAME
  htmlIndentation=INTEGER
  indentMode=BOOL
  language=german|english
  latexDocumentClass=scrartcl|scrreprt|scrbook|article|report|book
  latexDocumentOptions=STRING
  latexFontSize=10pt|11pt|12pt
  latexGeometry=STRING
  latexIndentation=FLOAT
  latexPageStyle=TITLEPAGE,OTHERPAGES
  latexPaperSize=PAPERSIZE
  latexParSkip=LENGTH
  latexShowFrames=BOOL
  quiet=BOOL
  sectionNumberLevel=-2|-1|0|1|2|3|4
  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 4.2.2 - [Eigenschaften](#) auf dieser Seite).

### 4.2.1. Segmente

In Titel (**title**), Autor (**author**) und Datum (**date**) des Dokuments können Segmente (siehe Abschnitt 4.20 - [Segment](#) auf Seite 48) verwendet werden:

A, B, C, G, I, L, M, N, Q, S

### 4.2.2. Eigenschaften

**anchor=STRING**

Anker des Dokuments.

**author=STRING**

Der Autor des Dokuments. Wenn gesetzt, wird eine Titelseite bzw. ein Titelabschnitt erzeugt.

**codeStyle=STYLE**

(Default: 'default') Der Style, in dem das Syntax-Highlighting von Code-Blöcken erfolgt, wenn

das Attribut `lang` gesetzt ist. Die Liste der verfügbaren Code Styles findet sich in Abschnitt [Code Styles](#) auf Seite 73.

**copyComments=BOOL**

(Default: 1) 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.

**cssPrefix=NAME**

(Default: 'sdoc3') Präfix für die CSS-Klassen der Elemente des Dokuments.

**date=STRING**

Das Datum des Dokuments. Wenn gesetzt, wird eine Titelseite bzw. ein Titelabschnitt erzeugt. Formatelemente von strftime werden expandiert. Spezielle Werte:

**'today'**

Wird unter L<sup>A</sup>T<sub>E</sub>X 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.

**htmlDocumentStyle=NAME**

(Default: default) Der Name des CSS-Stylesheet, das für das Design des HTML-Dokuments verwendet wird.

**htmlIndentation=INTEGER**

(Default: 20) In HTML die Tiefe der Einrückung für List-, Graphic-, Code-, Table-Blöcke vom linken Rand, wenn diese eingerückt werden sollen. Das Maß wird einheitenlos in Pixel angegeben.

**indentMode=BOOL**

(Default: 0) Wenn gesetzt, werden alle Code-, Graphic-, List- und Table-Blöcke, für die nichts anderes angegeben ist, eingerückt dargestellt.

**language=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 L<sup>A</sup>T<sub>E</sub>X die Trennregeln. Mögliche Werte: german, english.

**latexDocumentClass=NAME**

L<sup>A</sup>T<sub>E</sub>X-Dokumentklasse. Werte: 'scrartcl', 'scrreprt', 'scrbook', 'article', 'report', 'book', ...

**latexDocumentOptions=STRING**

Kommaseparierte Liste von L<sup>A</sup>T<sub>E</sub>X Dokumentklassen-Optionen. Z.B. 'twoside'.

**latexFontSize=SIZE**

(Default: '10pt') Die Größe des L<sup>A</sup>T<sub>E</sub>X-Font. Mögliche Werte: '10pt', '11pt', '12pt'.

**latexGeometry=STRING**

(Default: 'height=22.5cm,bottom=3.8cm' bei a4paper) Definition der Seitengeometrie mittels des L<sup>A</sup>T<sub>E</sub>X-Pakets `geometry`. Damit können gegenüber der Grundeinstellung abweichende Seitenmaße definiert werden.

**latexIndentation=FLOAT**

(Default: 11.5) In L<sup>A</sup>T<sub>E</sub>X die Tiefe der Einrückung für Code-, Graphic-, List- und Table-Blöcke vom linken Rand, wenn diese eingerückt werden sollen. Einheit ist pt, die aber nicht angegeben wird.

**latexPageStyle=NAME**

Definiere den Seitenstil der Titelseite und der sonstigen Seiten. Der Seitenstil legt fest, was im Kopf und im Fuß der Seite erscheint. Zur Verfügung stehen die Seitenstil-Bezeichnungen

'empty' (Kopf- und Fußzeile leer), 'plain' (nur Fuß mit Seitennummer), 'headings' (Kopf mit Abschnittstiteln, Fuß mit Seitennummer). Ist lediglich ein Seitenstil angegeben, gilt dieser für alle Seiten. Beispiel:

```
latexPageStyle="empty"
```

ist identisch zu

```
latexPageStyle="empty,empty"
```

**latexPaperSize=NAME**

(Default: 'a4paper') Papiergröße für L<sup>A</sup>T<sub>E</sub>X.

**latexParSkip=LENGTH**

(Default: '0.5ex') Vertikaler Abstand zwischen Absätzen.

**latexShowFrames=BOOL**

(Default: 0) Zeichne den Text-, Kopfzeilen-, Fußzeilen- und Kommentarbereich in die Seiten ein. Dies ist eine Debugging-Option.

**quiet=BOOL**

(Default: 0) Gib keine Warnungen aus.

**sectionNumberLevel=N**

(Default: 3) Die Abschnittsebene, bis zu welcher Abschnitte nummeriert werden. Mögliche Werte: -2, -1, 0, 1, 2, 3, 4, 5, 6, wobei das Maximum vom Zielformat abhängt (L<sup>A</sup>T<sub>E</sub>X: 4, HTML: 6). -2 bedeutet: Die Abschnitte des Dokuments werden nicht nummeriert.

**smallerMonospacedFont=BOOL**

Wähle einen kleineren Monospaced Font als standardmäßig.

**tableOfContents=BOOL**

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

**title=STRING**

Der Titel des Dokuments. Wenn gesetzt, wird eine Titelseite bzw. ein Titelabschnitt erzeugt.

### 4.3. Inhaltsverzeichnis (TableOfContents)

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

```
%TableOfContents:  
  htmlTitle=BOOL  
  htmlIndent=BOOL  
  maxLevel=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 4.2.2 - [Eigenschaften](#) auf Seite 12). Ist `tableOfContents=0` gesetzt, findet dies nicht statt und das Dokument erhält kein Inhaltsverzeichnis.

#### 4.3.1. Eigenschaften

##### **htmlIndent=BOOL**

(Default: undef) Forciere die Einrückung (wenn auf 1 gesetzt) oder Nicht-Einrückung (wenn auf 0 gesetzt).

##### **htmlTitle=BOOL**

(Default: 1) Versieh das Inhaltsverzeichnis in HTML mit einer Überschrift (Inhaltsverzeichnisöder Contents").

##### **maxLevel=N**

(Default: 3) Tiefe, bis zu welcher Abschnitte ins Inhaltsverzeichnis aufgenommen werden. Mögliche Werte: -2, -1, 0, 1, 2, 3, 4, 5, 6, wobei das Maximum vom Zielformat abhängt (L<sup>A</sup>T<sub>E</sub>X: 4, HTML: 6). -2 = kein Inhaltsverzeichnis.

#### 4.4. 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
```

Um einen Abschnittsbeginn im Quelltext hervorzuheben kann die Zeile mit einer Folge von Gleichheitszeichen (`=`) beliebiger Länge beendet werden. Die Folge von Gleichheitszeichen ist nicht Teil des Titels, sie wird automatisch entfernt.

```
= Dies ist ein Abschnitt der Ebene 1 =====
```

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

##### 4.4.1. 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  $\text{\LaTeX}$  stehen diese zur Verfügung, wenn die Dokumentklassen `scrreprt`, `scrbook`, `report`, `book` verwendet werden.

##### 4.4.2. Appendix

Die Appendizes eines Dokuments beginnen ab dem ersten Abschnitt der Ebene 1, bei dem auf das `=` 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. nummeriert. Sonst gibt es keinen Unterschied zwischen Abschnitten und Appendizes.

##### 4.4.3. 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 `maxLevel` können auf diesem Weg Abschnitte unterschiedlicher Ebenen vom Inhaltsverzeichnis ausgeschlossen werden.

#### 4.4.4. Segmente

Im Titel eines Abschnitts können Segmente (siehe Abschnitt 4.20 - [Segment](#) auf Seite 48) verwendet werden:

A, B, C, G, I, L, M, N, Q, S

#### 4.4.5. Eigenschaften

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

**anchor=STRING**

Anker des Abschnitts.

**isAppendix=BOOL**

Mit diesem Abschnitt beginnen die Appendizes.

**level=N**

Tiefe des Abschnitts in der Abschnittshierarchie, beginnend mit 1.

**title=STRING**

Titel des Abschnitts.

**tocStop=BOOL**

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

## 4.5. 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

Wie ein Abschnitt kann eine Zwischenüberschrift im Quelltext hervorgehoben werden, indem sie mit einer Folge von Gleichheitszeichen (=) beliebiger Länge beendet wird. Die Folge von Gleichheitszeichen ist nicht Teil des Titels, sie wird automatisch entfernt.

```
==* Zwischenüberschrift Stufe 2 =====
```

#### 4.5.1. Segmente

In einer Zwischenüberschrift können Segmente (siehe Abschnitt [4.20 - Segment](#) auf Seite [48](#)) verwendet werden:

A, B, C, G, I, L, M, N, Q, S

#### 4.5.2. Eigenschaften

```
%BridgeHead:
  anchor=STRING
  level=N
  title=STRING
```

##### **anchor=STRING**

Anker der Zwischenüberschrift.

##### **level=N**

Größe der Zwischenüberschrift, beginnend mit 1.

##### **title=STRING**

Titel der Zwischenüberschrift.

## 4.6. 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.
```

### 4.6.1. Eigenschaften

```
%PageBreak:
```

Der Seitenumbruch hat keine Eigenschaften.

### 4.7. 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.
```

#### 4.7.1. Segmente

In einem Paragraph können Segmente (siehe Abschnitt [4.20 - Segment](#) auf Seite [48](#)) verwendet werden:

```
B, C, G, I, L, M, N, Q, S
```

#### 4.7.2. Eigenschaften

```
%Paragraph:  
TEXT  
.
```

Ein Paragraph hat keine Eigenschaften.

## 4.8. Abbildung (Graphic)

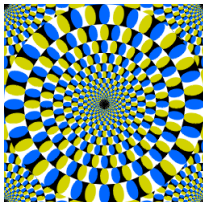
Eine Abbildung ist ein freistehendes Bild zwischen zwei Paragraphen oder anderen Block-Elementen.

Die Abbildung kann aus einer Datei (Attribut `file`) stammen oder eine Ressource im Netz sein (Attribut: `url`). Für  $\text{\LaTeX}$  wird aktuell eine lokale Datei benötigt. Ist ein URL definiert, wird dieser in HTML eingesetzt.

Die Größe der Darstellung wird durch die Angabe der Breite (`width`) und der Höhe (`height`) festgelegt. Die Werte werden ganzzahlig ohne Einheit angegeben und von Sdoc in der Einheit *Pixel* (1/96in) interpretiert. Für HTML ist dies die natürliche Einheit. Für  $\text{\LaTeX}$  werden die Werte in die Einheit *Point* (1/72in) umgerechnet (1px = 0.75pt), so dass die Abbildung in  $\text{\LaTeX}$  in der gleichen Größe dargestellt wird.

```
%Graphic:
  file="+/sdoc-graphic-illusion"
  width=100
  height=100
```

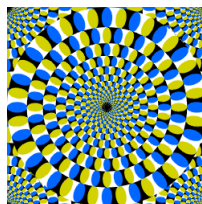
produziert:



Per Default wird die Abbildung linksseitig mit der Einrückung `latexIndentation` bzw. `htmlIndentation` (siehe Abschnitt 4.2 - [Dokument \(Document\)](#) auf Seite 12) gesetzt. Die Einrückung wird weggelassen wenn das Attribut `indent=0` gesetzt ist. Mit `align=center` kann die Abbildung auch mittig gesetzt werden.

```
%Graphic:
  file="+/sdoc-graphic-illusion"
  width=100
  height=100
  align=center
```

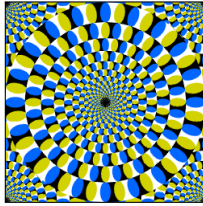
produziert:



Eine Abbildung kann durch Setzen des Attributs `caption` mit einer Beschriftung versehen werden. Das Setzen des Attributs `border` auf 1, fügt einen Rahmen hinzu.

```
%Graphic:
  file="+/sdoc-graphic-illusion"
  caption=Illusion
  width=100
  height=100
  border=1
```

produziert:



**Abbildung 1:** Illusion

Für eine Grafik, die *im Textfluss* erscheinen soll, siehe Abschnitt [4.20.4 - G - Inline-Grafik \(graphic\)](#) auf Seite [48](#).

#### 4.8.1. Segmente

In der Beschriftung (`caption`) einer Abbildung können alle Segmente (siehe Abschnitt [4.20 - Segment](#) auf Seite [48](#)) verwendet werden:

A, B, C, G, I, L, M, N, Q, S

#### 4.8.2. Eigenschaften

```
%Graphic:
  align=left|center
  anchor=STRING
  border=BOOL
  caption=STRING
  file=PATH
  height=INTEGER
  latexAlign=left|center
  latexOptions=STRING
  link=URL
  name=STRING
  indent=BOOL
  padding=LENGTH
  scale=FLOAT
  show=BOOL
  url=URL
  width=INTEGER
```

**align=left|center**

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

**anchor=STRING**

Anker der Grafik.

**border=BOOL**

(Default: 0) Zeichne einen Rahmen um die Abbildung.

**caption=STRING**

Beschriftung der Grafik. Diese erscheint unter der Grafik.

**file=PATH**

Pfad der Bilddatei, idealerweise ohne *ohne* Extension. Beginnt der Pfad mit +/, wird das Pluszeichen zum Pfad des Dokumentverzeichnis expandiert.

**height=INTEGER**

Höhe in Pixeln (ohne Angabe der Einheit), auf die das Bild skaliert wird.

**latexAlign=left|center**

Horizontale Ausrichtung des Bildes in L<sup>A</sup>T<sub>E</sub>X. Hat Priorität gegenüber dem Attribut align.

**latexOptions=STRING**

L<sup>A</sup>T<sub>E</sub>X-spezifische Optionen, die direkt an das L<sup>A</sup>T<sub>E</sub>X-Makro `\includegraphics` übergeben werden.

**link=LINK**

Versehe das Bild mit einem Verweis. Dies kann ein Verweis auf ein internes oder externes Ziel sein wie bei einem L-Segment (siehe 4.20.6 - L - Verweis (link) auf Seite 49).

**name=STRING**

Name der Grafik. Ein Name muss angegeben sein, wenn die Grafik von einem G-Segment referenziert wird.

**indent=BOOL**

Rücke die Grafik ein. Das Attribut ist nur bei `align=left` und bei Inline-Grafiken von Bedeutung.

**padding=LENGTH**

(Default: 0) Füge Leerraum der Breite LENGTH um die Grafik hinzu. Der Leerraum befindet sich zwischen Grafik und Rahmen (Attribut `border`). Der Abstand LENGTH muss, sofern nicht 0, mit einer Einheit wie px oder mm versehen werden. Ein Pixelwert wird für L<sup>A</sup>T<sub>E</sub>X nach pt gewandelt.

**scale=FLOAT**

Skalierungsfaktor. Der Skalierungsfaktor hat bei L<sup>A</sup>T<sub>E</sub>X Priorität gegenüber der Angabe von `width` und `height`.

**show=BOOL**

Wenn auf 1 gesetzt, wird die Grafik an Ort und Stelle angezeigt, wenn 0, nicht. Sie nicht anzuzeigen ist erforderlich, wenn sie lediglich von G-Segmenten (Inline Grafik) genutzt werden soll. Der Default für das Attribut ist 0, wenn der interne Zähler `useCount` > 0 (d.h. die Grafik wird im Dokument als Inline-Grafik genutzt), andernfalls 1.

**url=URL**

Wenn angegeben und URL nicht leer, wird in HTML `` gesetzt.

**width=INTEGER**

Breite in Pixeln (ohne Angabe der Einheit), auf die das Bild skaliert wird.

## 4.9. Aufzählung (List)

Es gibt drei verschiedene Arten von Listen.

### 4.9.1. 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

### 4.9.2. Nummerierungsliste

Eine Nummerierungsliste (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

### 4.9.3. Definitionsliste

Eine Definitionsliste (oder *definition list*) ist eine Abfolge von Punkten, deren erste Zeile jeweils mit 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
```



Alternativ kann der Doppelpunkt auch innerhalb der eckigen Klammern liegen [TERM:], der dann zum Bestandteil des Terms wird.

```
[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
```

Im Term der Liste sind Segmente zulässig. Die folgende Definitonsliste ist nicht schön, zeigt aber alle Möglichkeiten.

```
[C{monospace}]:
  Der Term ist C{monospace}.
[B{fett}]:
  Der Term ist in B{fetter} Schrift.
[I{kursiv}]:
  Der Term ist in I{kursiver} Schrift. (Kursive Schrift wird unter
  KOMA im Term nicht dargestellt, unter den klassischen
  LaTeX-Dokumentklassen schon)
[L{http://google.com}]:
  Der Term ist ein Verweis (auf eine Website).
[M~c=sqrt{a^2+b^2}~]:
  Der Term ist eine mathematische Formel.
[Q{Zitat}]:
  Der Term ist in Anführungsstriche gesetzt.
[G{InvaderItem}]:
  Der Term ist eine Grafik.
```

produziert:

```
monospace
  Der Term ist monospace.
fett
  Der Term ist in fetter Schrift.
kursiv
  Der Term ist in kursiver Schrift. (Kursive Schrift wird unter KOMA im Term nicht dargestellt,
  unter den klassischen LATEX-Dokumentklassen schon)
http://google.com
  Der Term ist ein Verweis (auf eine Website).
 $c = \sqrt{a^2 + b^2}$ 
  Der Term ist eine mathematische Formel.
“Zitat”
  Der Term ist in Anführungsstriche gesetzt.
```



Der Term ist eine Grafik.



#### 4.9.4. 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
```

#### 4.9.5. Segmente

Im Term einer Definitionsliste können Segmente (siehe Abschnitt [4.20 - Segment](#) auf Seite [48](#)) verwendet werden:

B, C, G, I, L, M, N, Q, S

#### 4.9.6. Eigenschaften

##### List

```
%List:
  indent=BOOL
  listType=TYPE
```

##### indent=BOOL

Rücke die Liste ein. Das Attribut ist nur für Aufzählungs- und Markierungslisten von Bedeutung.

##### listType=TYPE

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

**Item**

`%Item:`

`key=STRING`

**key=STRING**

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

### 4.10. 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

#### 4.10.1. Definition

Die Definition umfasst den Kopf und den Körper der Tabelle. Der Kopf enthält die Spalten-Titel, der Körper die Spalten-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 Spalten zu definieren.

Obige Tabelle besitzt aufgrund der drei Abschnitte aus Bindestrichen

```
-----
```

drei Spalten:

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

Alle Zeichen außerhalb dieser Bereiche werden *nicht* beachtet. Dies gilt sowohl für die Spalten-Titel als auch die Spaltendaten.

#### 4.10.2. Ausrichtung

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

#### 4.10.3. Ohne Titelzeile

Die Titelzeile ist optional. Sie kann auch weggelassen werden. Die Trennzeile für die Spaltendefinition 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

#### 4.10.4. Mehrzeilige Daten

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

```
%Table:
```

```

Nach  Nach
Rechts Links      Zentriert
-----
      1 Dies ist die      A
        erste Zeile
      2 Zweite Zeile      B
      3 Die dritte       C
        Zeile
.
```

produziert:

Nach Rechts	Nach Links	Zentriert
1	Dies ist die erste Zeile	A
2	Zweite Zeile	B
3	Die dritte Zeile	C

Bei Tabellen mit mehrzeiligen Kolumnendaten werden per Default, d.h. wenn für **border** nichts anderes definiert ist, zusätzlich die Spalten mit senkrechten Linien getrennt.

Damit die ASCII-Darstellung besser lesbar ist, können die Trennzeilen zwischen den Datenzeilen auch mit Bindestrichen geschrieben werden:

```
%Table:
```

```

Nach  Nach
Rechts Links      Zentriert
-----
```

```

      1 Dies ist die      A
        erste Zeile
-----
      2 Zweite Zeile      B
-----
      3 Die dritte       C
        Zeile
.

```

#### 4.10.5. 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.

#### 4.10.6. Titelfarbe

Die Titelfarbe kann mittels des Attributs `titleColor` gesetzt werden:

```

%Table:
  titleColor="#ffd700"
  caption="titleColor=#ffd700"

```

```

Links Rechts Zentriert
-----
A          1    AB
AB         12   CD
.

```

produziert:

Links	Rechts	Zentriert
A	1	AB
AB	12	CD

**Tabelle 5:** titleColor=#ffd700

**4.10.7. 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 6:** border=0

Linie unter Titelzeile:

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

**Tabelle 7:** border=t, titleColor=0

Linie unter Titel und ober- und unterhalb der Tabelle:

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

**Tabelle 8:** border=tH

Horizontale Linien:

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

**Tabelle 9:** border=hH

Innere horizontale und vertikale Linien:

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

**Tabelle 10:** border=hv, titleColor=0

Linie unter Titel und um die Tabelle:

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

**Tabelle 11:** border=tHV

Alle horizontalen Linien und vertikale Außenlinien:

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

**Tabelle 12:** border=hHV

Alle Linien:

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

**Tabelle 13:** border=hvHV

#### 4.10.8. Segmente

In den Zellen der Tabelle - sowohl im Titel als auch in den Daten -, sowie in der Bildunterschrift (**caption**) können alle Segmente (siehe Abschnitt 4.20 - [Segment](#) auf Seite 48) verwendet werden:

A, B, C, G, I, L, M, N, Q, S

Das folgende Beispiel zeigt sämtliche Möglichkeiten.

```
%Table:
caption="Segmente: A{TabelleSegmente} B{fett}, C{mono}, G{InvaderList}, \
I{kursiv}, L{http://google.com}, M~c=\sqrt{a^2+b^2}~, Q{Zitat}"
```

```
L{Segmente}
```

```
-----
```

```
B{Dies ist fett}
C{Dies ist monospace}
G{InvaderList}
I{Dies ist kursiv}
L{http://google.com}
M~c=\sqrt{a^2+b^2}~
Q{Dies ist ein Zitat}
```

```
.
```


```
%Graphic:
name="InvaderList"
file="+/sdoc-graphic-invader"
scale=0.03
```



```
show=0
```

```
%Link:
name="Segmente"
regex="Syntax/Segment"
```

produziert:

Segmente auf Seite 48
Dies ist fett
Dies ist monospace

<i>Dies ist kursiv</i>
<a href="http://google.com">http://google.com</a>
$c = \sqrt{a^2 + b^2}$
“Dies ist ein Zitat”

**Tabelle 14:** Segmente: **fett**, **mono**, , *kursiv*, <http://google.com>,  $c = \sqrt{a^2 + b^2}$ , “Zitat”



#### 4.10.9. Eigenschaften

```
%Table:
anchor=STRING
border=STRING
caption=STRING
indent=BOOL
titleColor=COLOR
TEXT
.
```

##### **anchor=STRING**

Anker der Tabelle.

##### **border=STRING**

(Default: 'hHV') 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=STRING**

Die Beschriftung der Tabelle. Diese erscheint unter der Tabelle.

##### **indent=BOOL**

Rücke die Tabelle ein. Das Attribut ist nur bei **align=left** von Bedeutung.

**titleColor=COLOR**

(Default: `'#e8e8e8'`) Die Farbe der Titelzeile. Der Wert wird als RGB-Wert angegeben.

**4.10.10.  $\LaTeX$  Setup**

In  $\LaTeX$  ist die Tabelle als `longtable` realisiert. Diese setzen wir, abweichend vom Default-Setup, linksbündig mit einer Einrückung, so dass sie mit anderen Konstrukten wie [List](#) auf Seite 24, [Code](#) auf der nächsten Seite, [Graphic](#) auf Seite 21 gleichtief eingerückt ist.

### 4.11. 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.

#### 4.11.1. 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";
```

#### 4.11.2. 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.

**Achtung:** Der Quelltext innerhalb des `%Code`-Blocks sollte generell eingerückt werden, damit Zeilen, die mit `#` beginnen, nicht als Sdoc-Kommentarzeilen interpretiert und ausgefiltert werden!

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

produziert:

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

#### 4.11.3. 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);
}

```

#### 4.11.4. Zeilennummern

Wird das Attribut `ln=N` mit `N > 0` gesetzt, werden die Zeilen durchnummeriert, beginnend mit Zeilennummer `N`.

```

%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;

```

Die Nummerierung findet auch statt, wenn kein Syntax-Highlighting aktiviert ist.

```

%Code: 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;

```

#### 4.11.5. 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/issue"
```

produziert:

```
Debian GNU/Linux 10 \n \l
```

#### 4.11.6. 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 Dokumentverzeichnis vorangestellt.

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

produziert:

```
Mon 01 Jan 2001 12:00:00 AM CET
```

#### 4.11.7. 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
```

#### 4.11.8. 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 "# begin" und "# end" gekennzeichnet sein ("# begin":

```
%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;
}

```

#### 4.11.9. 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=COMMAND**

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=REGEX**

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=COMMAND**

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=BOOL**

(Default: *kontextabhängig*) Rücke den Text ein. Im Falle von Zeilennummern (**ln=N**) wird *nicht* eingerückt. Sonst wird eingerückt. Durch explizite Setzung des Attributs kann der jeweilige

Default überschrieben werden.

**load=FILE**

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=LANGUAGE**

Der Code ist Quelltext der Sprache LANGUAGE. Mit dieser Option wird das Syntax-Highlighting aktiviert. Dies verfügbaren Sprachen (“Lexer”) liefert das Kommando

```
$ pygmentize -L lexers
```

**ln=N**

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

## 4.12. 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 Ermittlung des Ziels aus dem L-Segment

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

wird also der kanonisierte Text

```
Schachelung von Segmenten
```

gebildet.

### 4.12.1. Interner Verweis

### 4.12.2. Eigenschaften

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

#### **name=STRING**

Name der Link-Definiton. Mehrere Namen können mit | getrennt definiert werden. Beispiel:  
name=SSdoc|Sdoc Homepage"

#### **file=PATH**

Pfad einer lokalen Datei.

#### **regex=REGEX**

Regex, der den internen Zielknoten identifiziert.

#### **url=URL**

URL eines externen Dokuments.



### 4.13. 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.

```
# eof
```

#### 4.13.1. Eigenschaften

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

##### **load=PATH**

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

### 4.14. Zitat (Quote)

Ein Abschnitt mit Größerzeichen plus Leerzeichen am Zeilenanfang wird als Zitat-Block gesetzt

```
> Es ist verdammt schwer, einen Menschen zu nehmen,  
> wie er ist, wenn er sich anders gibt, als er ist.  
> (Ernst Ferstl)
```

produziert:

*Es ist verdammt schwer, einen Menschen zu nehmen, wie er ist, wenn er sich anders gibt,  
als er ist. (Ernst Ferstl)*

#### 4.14.1. Eigenschaften

```
%Quote:  
TEXT  
.
```

Ein Zitat-Block hat keine Eigenschaften.

## 4.15. Kommentar (Comment)

Sdoc unterscheidet zwei Arten von Kommentaren:

1. Kommentare mit einem Hash (“#”) am Zeilenanfang
2. Kommentare mit einem Prozentzeichen plus Leerzeichen (“% ”) am Zeilenanfang

### 4.15.1. #-Kommentare

Zeilen, die mit einem Hash beginnen, werden ignoriert. Sie können in jeder Zeile im Sdoc-Quelltext vorkommen. Sie sind für Quelltextnotizen nützlich oder zum Auskommentieren von Sdoc-Code. Beispiel:

```
# Dies ist ein Kommentar
```

### 4.15.2. %-Kommentare

Zeilen, die mit Prozentzeichen (plus Leerzeichen) beginnen, werden als Kommentar-Blöcke in den Syntaxbaum übernommen. Sie sind, da sie eine eigene syntaktische Einheit bilden, nur *zwischen* anderen Blöcken zulässig. Der Kommentar

```
% Dies ist ein Kommentar, der in den
% Quelltext des Zieldokuments übertragen wird
```

kann äquivalent in Block-Schreibweise geschrieben werden als

```
%Comment:
Dies ist ein Kommentar, der in den
Quelltext des Zieldokuments übertragen wird
.
```

%-Kommentare zeichnen sich dadurch aus, dass sie in den Quelltext *des Zieldokuments* übertragen werden, sofern das Zielformat Kommentare unterstützt. Sie sind für Notizen, die im Zieldokument-Quelltext erscheinen sollen, oder für Markierungen, die das Debugging oder eine Weiterverarbeitung des Zieldokuments vereinfachen, nützlich.

Das Übertragen von %-Kommentaren in den Ziel-Quelltext kann durch die Dokument-Eigenschaft `copyComments=BOOL` gesteuert werden. Per Default ist `copyComments=1` definiert, d.h. alle %-Kommentare werden übertragen. Wird `copyComments=0` gesetzt, findet keine Übertragung statt.

### 4.15.3. Eigenschaften eines %-Kommentars

```
%Comment:
TEXT
.
```

Ein %-Kommentar hat keine Eigenschaften.

## 4.16. Zielformatblock (Format)

Mit diesem Konstrukt kann der Quelltext eines oder mehrerer Zielformate direkt in das Dokument eingebunden werden. Der Quelltext wird ohne jede Änderung, auch was Einrückung und Leerraum angeht, bei Erzeugung des Dokuments in dem jeweiligen Zielformat in den Gesamtquelltext einzu-eins eingesetzt. Es ist ein Fehler, wenn ein Zielformatblock keinen Quelltext für das konkret erzeugte Zielformat definiert. Der Quelltext eines Zielformats kann leer sein.

### 4.16.1. Eigenschaften

```
%Format:  
@@FORMAT@@  
CODE  
...  
.
```

#### **FORMAT**

Name des Formats (Groß/Klein-Schreibung beliebig). Z.B. `HTML`, `Latex`, `MediaWiki`.

#### **CODE**

Code des jeweiligen Formats.

Es können mehrere `FORMAT/CODE`-Abschnitte aufeinanderfolgen.

### 4.17. Stilvorlage (Style)

In HTML kann das Aussehen des Dokuments durch CSS-Regeln verändert werden. Die CSS-Regeln können *innerhalb* des Dokuments oder *außerhalb* in einer Datei definiert werden. Beide Möglichkeiten - die lokale Definition und das Laden eines externen StyleSheets - bietet der Style-Block.

Im Rumpf des Style-Blocks **CODE** werden lokale CSS-Regeln definiert. Ein externes StyleSheet wird durch Setzen des Attributs **source=PATH** geladen. In einem Style-Block kann auch beides zusammen angewendet werden. Die lokal definierten Regeln eines Style-Blocks werden *nach* dem externen StyleSheet des Blocks eingebunden, haben gegenüber dessen Setzungen somit Priorität. Mit lokalen CSS-Regeln können die Setzungen des externen StyleSheet also modifiziert oder überschrieben werden.

Für mehrere externe StyleSheets müssen mehrere Style-Blöcke definiert werden.

Innerhalb des Stylesheet-Code wird die Zeichenfolge **\$PREFIX** durch den cssPrefix des Dokuments ersetzt. Z.B.

```
%Style:
  . $PREFIX-code {
    background-color: yellow;
  }
.
```

#### 4.17.1. Eigenschaften

```
%Style:
  source=PATH
CODE
.
```

#### **CODE**

Lokale CSS-Regeln.

#### **source=PATH**

Pfad einer externen Stylesheet-Datei. Beginnt der Pfad mit +/, wird das Pluszeichen zum Pfad des Dokumentverzeichnis expandiert.

### 4.18. Selbstdefinierte Textauszeichnung (Segment)

Soll ein Textteil in einer Weise dargestellt werden, die durch die vordefinierten Segmente B, C, I nicht erreicht wird, kann eine eigene Textauszeichnung definiert werden.

#### 4.18.1. Beispiel

Ein Teil des Textes soll rot dargestellt werden. Zunächst die Segmenttyp-Definiton:

```
%Segment:
  name=red
  html='<span style="color: red">%s</span>'
  latex='{\\color{red}%s}'
  mediawiki='<span style="color: red">%s</span>'
```

Dann die Anwendung im Text:

```
... S{red,Dies ist wichtig} ...
```

Ergebnis:

... Dies ist wichtig ...

#### 4.18.2. Eigenschaften

```
%Link:
  name=STRING
  html=CODE
  latex=CODE
  mediawiki=CODE
```

**name=STRING**

Name des Segment-Typs.

**html=CODE**

HTML-Code des Segment-Typs.

**latex=CODE**

L<sup>A</sup>T<sub>E</sub>X-Code des Segment-Typs.

**mediawiki=CODE**

MediaWiki-Code des Segment-Typs.

### 4.19. Nachverarbeitung (PostProcessor)

Durch ein oder mehrere Post-Prozessoren kann das resultierende Dokument (in einem konkreten Format) nachträglich manipuliert werden. Die Sprache, in der dies geschieht, ist Perl.

```
%PostProcessor:
my ($doc,$format,$code) = @_;

CODE
.
```

Der Post-Prozessor erhält drei Parameter:

#### **\$doc**

Eine Referenz auf den Wurzelknoten des Dokumentbaums.

#### **\$format**

Den Bezeichner für das Format. Mögliche Werte: 'latex', 'html', 'mediawiki' usw.

#### **\$code**

Der Quelltext in dem angegebenen Format.

Beispiel:

```
%PostProcessor:
my ($doc,$format,$code) = @_;

if ($format eq 'latex') {
    # Für die LaTeX-Titelseite ist das Autor-Feld zu lang, daher
    # ersetzen wir das erste Komma durch einen Zeilenumbruch
    $code =~ s/(\author\{.*?\},\s*/$1\\\\\/;
}
.
```

#### 4.19.1. Eigenschaften

```
%PostProcessor:
my ($doc,$format,$code) = @_;

CODE
.
```

#### **CODE**

Perl-Code.

## 4.20. Segment

### 4.20.1. A - Anker (anchor)

### 4.20.2. B - Fettschrift (bold)

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

#### Beispiel

Du machst mich B{sehr} wütend.

Du machst mich **sehr** wütend.

### 4.20.3. 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.

### 4.20.4. 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 4.8 - [Abbildung \(Graphic\)](#) auf Seite 21, 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.

### 4.20.5. 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.



**4.20.6. L - Verweis (link)****Beispiel: Interner Link**

`L{Dokument}`

[Dokument](#) auf Seite 12

**Beispiel: Interner Link mit Abschnitts-, Bild-, Tabellennnummer**

`L{+Dokument}`

4.2 - [Dokument \(Document\)](#) auf Seite 12

**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](#)

**4.20.7. M - Mathematische Formel (math)**

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

**Beispiel**

Satz des Pythagoras: `M~c=\sqrt{a^2+b^2}~`

produziert:

Satz des Pythagoras:  $c = \sqrt{a^2 + b^2}$

**4.20.8. N - Zeilenumbruch (newline)**

Die Zeichenkette `~N~` forciert einen Zeilenumbruch.

Im Falle von  $\text{\LaTeX}$  ist Vorsicht geboten. An manchen Stellen sind keine Zeilenumbrüche erlaubt oder, wie z.B. im Dokumenten-Titel, können nicht mehrere Zeilenumbrüche aufeinander folgen. Fehlermeldung in diesen Situationen: “ $\text{\LaTeX}$  Error: There’s no line here to end”).

`Dies ist~N~ein Test.`

produziert:

Dies ist  
ein Test.

**4.20.9. 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  $\text{\LaTeX}$  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"?

**4.20.10. S - Selbstdefinierte Textauszeichnung**

Der von `S{name,...}` eingefasste Text wird mit der selbst definierten Textauszeichnung *name* gesetzt.

**Beispiel**

`S{red,Dies ist wichtig}`

produziert:

*Dies ist wichtig*

**4.20.11. 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-Buchstaben ist hier nicht der richtige Weg, da die geschweiften Klammern geschützt werden müssen.

`C{C\{TEXT\}}`

`C{TEXT}`

**Schachtelung von Segmenten**

Segmente können geschachtelt werden.

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

**So geht es *auch*!**

I{Näheres in Abschnitt L{Dokument}}

*Näheres in Abschnitt [Dokument](#) auf Seite [12](#)*

## 4.21. Beispiel in Block-Syntax

Hier das Beispiel aus Abschnitt 2.1 - [Sdoc Quelltext](#) auf Seite 6 ohne Wiki-Syntax, also ausschließlich in Block-Syntax. Der Quelltext wird dadurch deutlich länger und ist weniger gut lesbar.

```

1 %Document:
2   title="Ein Beispieldokument"
3   author="Nico Laus"
4   date="today"
5   latexPageStyle="empty"
6
7 %Section:
8   level=1
9   title="Einleitung"
10
11 %Paragraph:
12 Dies ist eine I{Einleitung}. Mit wenig C{Aufwand} erstellen wir in
13 Sdoc ein B{Dokument}. Als Textauszeichnungen sind auch Kombinationen
14 wie B{I{fett und kursiv}} möglich.
15 .
16
17 %Section:
18   level=2
19   title="Ein Beispiel"
20
21 %Paragraph:
22 Mit dem Satz des Pythagoras  $M^2=a^2+b^2$  berechnen wir die Länge der
23 Hypotenuse:  $M=\sqrt{a^2+b^2}$ .
24 .
25
26 %Graphic:
27   file="+/sdoc-graphic-illusion"
28   align="center"
29   scale=0.3
30   width=100
31   height=100
32   border=1
33
34 %Paragraph:
35 Programm-Quelltexte können wir mit Q{Syntax-Highlighting} darstellen:
36 .
37
38 %Code: lang=Perl ln=1 indent=1
39   open(my $fh, 'ls -l |') or die "open failed ($!)\n";
40   while (<$fh>) {
41     s/\n/<BR>\n/;
42     print;
43   }
44   close($fh) or die "close failed ($!)\n";
45 .
46
47 %Paragraph:
48 Auf eine andere Stelle im Dokument zu verweisen ist einfach. Hier ist

```

```
49 ein Verweis auf die L{Einleitung}.
50 .
51
52 %List:
53   listType="unordered"
54 %Item: key="*"
55   %Paragraph:
56     Apfel
57   .
58 %Item: key="*"
59   %Paragraph:
60     Birne
61   .
62 %Item: key="*"
63   %Paragraph:
64     Pflaume
65   .
66
67 %Paragraph:
68 Dies ist eine einfache Aufzählung. Aufzählungen können beliebig
69 geschachtelt werden.
70 .
71
72 %Comment:
73 # eof
74 .
```

## 5. Einrückung von Blöcken

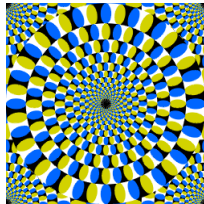
Die Einrückung von Code-, Graphic-, List- und Table-Blöcken vom linken Dokument-Rand kann individuell oder global eingestellt werden. Die individuelle Einstellung hat Vorrang gegenüber der globalen Einstellung.

Die globale Einstellung erfolgt durch das Attribut `Document.indentMode=BOOL` (Default: 0), die individuelle durch das Attribut `Block.indent=BOOL` (Default: undef).

### 5.1. Ohne Einrückung

Keine Einrückung findet statt, wenn `Document.indentMode=0` und `Block.indent=undef` oder `Block.indent=0` eingestellt ist (Ausnahme siehe Abschnitt [Sonderfall Listen](#) auf Seite 56).

- Apfel
- Birne
- Pflaume



1. Einkaufen
2. Kochen
3. Abwaschen

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

```

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

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

```

**255 239 213**

PapayaWhip

**188 143 143**

RosyBrown

**255 218 185**

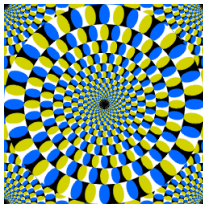
PeachPuff

## 5.2. Mit Einrückung

Eine Einrückung findet statt, wenn `Document.indentMode=1` und `Block.indent=undef` oder `Block.indent=1` eingestellt ist:

Die Tiefe der Einrückung wird für die Zielformate getrennt festgelegt. Dies geschieht durch die globalen Dokument-Attribute `FormatIndentation=FLOAT`. Hierbei ist *Format* der Bezeichner für das Zielformat: `latex` für L<sup>A</sup>T<sub>E</sub>X, `html` für HTML. Der Wert des jeweiligen Attributs gibt den Abstand vom linken Rand des Dokuments an. Im Falle von L<sup>A</sup>T<sub>E</sub>X wird der Abstand in *Punkt* (pt) gemessen, im Falle von HTML in *Pixel* (px). Es wird der numerische Wert angegeben, die Einheit ist *nicht* Bestandteil des Attributwerts.

- Apfel
- Birne
- Pflaume



1. Einkaufen
2. Kochen
3. Abwaschen

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

```

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

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

```

**255 239 213**  
PapayaWhip

**188 143 143**  
RosyBrown

**255 218 185**  
PeachPuff

### 5.3. Sonderfall Listen

Die drei Listenarten - `unordered`, `ordered` und `description` - stellen einen Sonderfall hinsichtlich der Einrückung dar. Sie werden, sofern ihr Attribut `Block.indent` nicht explizit auf den Wert 0 gesetzt wird, immer eingerückt, also auch, wenn `Document.indentMode=0` gesetzt ist.

#### Unordered

- Apfel
- Birne
- Pflaume

#### Ordered

1. Einkaufen
2. Kochen
3. Abwaschen

#### Description

**255 239 213**  
PapayaWhip

**188 143 143**  
RosyBrown

**255 218 185**  
PeachPuff



## 6. Verweise

Ein Verweis (Link) referenziert eine andere Stelle im selben Dokument (*interner* Verweis) oder eine Ressource, die sich außerhalb des Dokuments befindet (*externer* Verweis).

### 6.1. Einen Verweis setzen

In Sdoc wird ein Verweis, gleichgültig, ob intern oder extern, durch ein L-Segment gesetzt:

```
... L{TEXT} ...
```

Die Punkte stellen hierbei irgendeinen umgebenden Kontext dar. Der Verweis-Text **TEXT** ist der Text, der im Zieldokument erscheint und für den Leser als Verweis kenntlich gemacht wird. Der Text kann auch umbrochen sein, sich also über mehr als eine Zeile erstrecken:

```
... L{Dies ist ein  
längerer Verweis-Text} ...
```

Der Text zwischen den geschweiften Klammern eines L-Segments kann beliebige Zeichen enthalten. Lediglich die Zeichen `{}` (öffnende geschweifte Klammer) und `}` (schließende geschweifte Klammer) müssen - wie bei allen Segmenten - durch einen Backslash (`\`) geschützt werden, sofern sie nicht zu einem eingebetteten Segment gehören.

Der Verweis-Text wird intern vor der Ermittlung des Verweis-Ziels kanonisiert:

1. Einleitende Verweis-Attribute werden entfernt (siehe Abschnitt [6.4 - Verweis-Attribute](#) auf der nächsten Seite).
2. Zeilenumbrüche werden durch Leerzeichen ersetzt.
3. Leerzeichen am Anfang und am Ende werden entfernt.
4. Folgen von zwei und mehr Leerzeichen werden auf *ein* Leerzeichen reduziert.

Beispiel: Der Verweis-Text des L-Segments

```
... siehe Abschnitt L{+Schachtelung  
von Segmenten} für weitere ...
```

wird intern kanonisiert zum Verweis-Text

```
Schachelung von Segmenten
```

Das Segment-Attribut `+` am Anfang wurde entfernt sowie der Zeilenumbruch und die *zwei* Leerzeichen zwischen `von` und `Segmenten` wurden durch jeweils ein Leerzeichen ersetzt.

### 6.2. Referenzierbare Objekte

- BridgeHead
- Graphic
- Section
- Table

### 6.3. Darstellung eines Verweises

```
... <Objekt-Nr> <Verweis-Text> <Seitenangabe> ...  
      (optional)      (immer)      (nur Druckfassung)
```

## 6.4. Verweis-Attribute

### 6.4.1. +

Wird ein interner Verweis mit einem Pluszeichen (+) eingeleitet, bedeutet dies, dass der Verweis-Text vom referenzierten Objekt genommen und ihm die Nummer des Objekts (Abschnitt, Bild, Tabelle) voran gestellt wird, sofern vorhanden. Der Verweis

```
... siehe Abschnitt L{Einen Verweis setzen} ...
```

erscheint im Zieldokument als

... siehe Abschnitt [Einen Verweis setzen](#) auf der vorherigen Seite ...

wohingegen der Verweis

```
... siehe Abschnitt L{+Einen Verweis setzen} ...
```

erscheint als

... siehe Abschnitt [6.1 - Einen Verweis setzen](#) auf der vorherigen Seite ...

wobei der Verweis-Text auch abweichen kann, je nachdem, ob der Text mit dem Text des referenzierten Objekts übereinstimmt (was hier nicht der Fall ist).

In der Druck-Fassung eines Dokuments folgt auf den Verweis-Text grundsätzlich die Angabe der Seite, auf der das Verweis-Ziel zu finden ist. In der Online-Fassung, die nicht seitenbasiert ist, gibt es diese Seitenangabe natürlich nicht.

## 7. Formate

### 7.1. HTML/CSS

Im Folgenden ist für jeden Block-Typ der Aufbau seines HTML-Codes und eine Liste von CCS-Selektoren wiedergegeben, mit denen das Aussehen des Blocks beeinflusst werden kann. Die Liste der Selektoren zeigt exemplarisch, wie der Block und seine Bestandteile adressiert werden können.

#### 7.1.1. BridgeHead

##### 7.1.1.1. HTML

```
<hN class="sdoc-bridgehead">TITLE</hN>
```

N ist eine ganze Zahl von 1 bis 6.

##### 7.1.1.2. CSS

Selektor	Ziel
.sdoc-bridgehead	Alle Zwischenüberschriften
hN.sdoc-section	Zwischenüberschriften der Ebene N

##### 7.1.1.3. Beispiele

###### Alle Zwischenüberschriften in dunkelgrauer Schrift

```
.sdoc-bridgehead {
    color: #404040;
}
```

### 7.1.2. Code (FIXME: Selektoren und Beispiele ergänzen)

#### 7.1.2.1. HTML

##### Ohne Zeilennummern

```
<div class="sdoc-code [indent]">
  <pre>TEXT</pre>
</div>
```

##### Mit Zeilennummern

```
<div class="sdoc-code [indent]">
  <table>
  <tr>
    <td class="ln">
      <pre>LINENUMBERS</pre>
    </td>
    <td class="margin"></td>
    <td class="text">
      <pre>TEXT</pre>
    </td>
  </tr>
</table>
</div>
```

### 7.1.2.2. CSS

#### Allgemein

Selektor	Ziel
<code>.sdoc-code</code>	Alle Code-Blöcke
<code>.sdoc-code.indent</code>	Eingerückte Code-Blöcke

#### Ohne Zeilennummern

Selektor	Ziel
<code>.sdoc-code &gt; pre</code>	Quelltext ohne Zeilennummern

#### Mit Zeilennummern

Selektor	Ziel
<code>.sdoc-code table</code>	Tabelle als Ganzes
<code>.sdoc-code td.ln</code>	Zeilennummern-Kolumne
<code>.sdoc-code td.margin</code>	Margen-Kolumne
<code>.sdoc-code td.text</code>	Text-Kolumne

### 7.1.2.3. Beispiele

#### Alle Code-Blöcke farbig hinterlegen

```
.sdoc-code {
  background-color: yellow;
}
```

#### Abstand zu anderen Blöcken

```
.sdoc-code {
  margin-top: 16px;
  margin-bottom: 16px;
}
```

### 7.1.3. Comment

#### 7.1.3.1. HTML

Einzeiliger Kommentar:

```
<!-- TEXT -->
```

Mehrzeiliger Kommentar:

```
<!--
  TEXT
-->
```

#### 7.1.3.2. CSS

Ein Comment-Block ist visuell nicht repräsentiert und daher auch nicht Gegenstand von CSS-Regeln.

### 7.1.4. Document

#### 7.1.4.1. HTML

```
<div class="sdoc-document">
  <h1>TITLE</h1>
  <p>
    <span class="author">AUTHOR</span>, <span class="date">DATE</date>
  </p>
</div>
```

Alle drei Bestandteile des Dokument-Kopfs - TITLE, AUTHOR, DATE - sind optional. Sind alle drei Bestandteile nicht gegeben, entfällt der Dokument-Kopf.

#### 7.1.4.2. CSS

Selektor	Ziel
.sdoc-document	Kopf des Dokuments insgesamt
.sdoc-document h1	Titel des Dokuments
.sdoc-document p	Bereich aus Autor und Datum
.sdoc-document .author	Autor
.sdoc-document .date	Datum

#### 7.1.4.3. Beispiele

##### Dokument-Kopf zentrieren (ähnli. $\LaTeX$ )

```
.sdoc-document {
  text-align: center;
}
```

##### Titel vergrößern, Abstand zu Author, Date verkleinern

```
.sdoc-document h1 {
  font-size: 230%;
  margin-bottom: 10px;
}
.sdoc-document p {
  margin-top: 10px;
}
```

##### Author, Date in größerer Schrift

```
.sdoc-document p {
  font-size: 115%;
}
```

### 7.1.5. (Graphic)

#### 7.1.5.1. HTML

```
<div class="sdoc-graphic [indent]" id="ID">
  
  <p>
    <span class="prefix">PREFIX:</span> <span class="caption">CAPTION</span>
```

```

    </p>]
  </div>

```

### 7.1.5.2. CSS

Selektor	Ziel
.sdoc-graphic	Das Konstrukt insgesamt
.sdoc-graphic.noindent	Das Konstrukt ohne Einrückung

### 7.1.6. Include

Ein Include-Block bindet einen externen Sdoc-Quelltext ein, hat aber selbst keine Darstellung in HTML und ist daher auch nicht Gegenstand von CSS-Regeln.

### 7.1.7. Link

Ein Link-Block definiert einen Verweis, hat aber selbst keine Darstellung in HTML und ist daher auch nicht Gegenstand von CSS-Regeln.

### 7.1.8. List/Item

#### 7.1.8.1. HTML

##### unordered

```

<div class="sdoc-list [noindent]" id="ID">
  <ul>
    <li>
      ...
    </li>
    ...
  </ul>
</div>

```

##### ordered

```

<div class="sdoc-list [noindent]" id="ID">
  <ol>
    <li>
      ...
    </li>
    ...
  </ol>
</div>

```

##### description

```

<div class="sdoc-list [noindent]" id="ID">
  <dl>
    <dt>...</dt>
    <dd>
      ...
    </dd>
    ...
  </dl>

```

```

    </dl>
  </div>

```

Die drei Listenarten können beliebig verschachtelt auftreten. Das `<div>` wird nur um die oberste Liste herum gesetzt.

### 7.1.8.2. CSS

Im Folgenden ist LTAG `ul`, `ol` oder `dl` und ITAG `li` oder `dd`.

Selektor	Ziel
<code>.sdoc-list</code>	Das Konstrukt insgesamt
<code>.sdoc-list.noindent</code>	Das Konstrukt ohne Einrückung
<code>.sdoc-list LTAG</code>	Alle Listen vom Typ TAG
<code>.sdoc-list &gt; LTAG</code>	Die oberste Liste (vom Typ TAG)
<code>.sdoc-list ITAG &gt; *:first-child</code>	Erstes Sub-Element eines Item
<code>.sdoc-list ITAG &gt; *:last-child</code>	Letztes Sub-Element eines Item

### 7.1.8.3. Beispiele

Terme einer Definitionsliste fett darstellen

```

.sdoc-list dt {
    font-weight: bold;
}

```

## 7.1.9. PageBreak

### 7.1.9.1. HTML

```

<div style="page-break-before: always"></div>

```

### 7.1.9.2. CSS

Ein Seitenumbruch ist als HTML-Konstrukt "atomar" und daher nicht Gegenstand von CSS-Regeln.

## 7.1.10. Paragraph

### 7.1.10.1. HTML

```

<p class="sdoc-paragraph">
  TEXT
</p>

```

### 7.1.10.2. CSS

Selektor	Ziel
<code>.sdoc-paragraph</code>	Alle Paragraphen

## 7.1.11. Section

### 7.1.11.1. HTML

```

<hN class="sdoc-section">TITLE</hN>

```

N ist eine ganze Zahl von 1 bis 6.

#### 7.1.11.2. CSS

Selektor	Ziel
.sdoc-section	Alle Abschnittsüberschriften
hN.sdoc-section	Abschnittsüberschriften der Ebene N

#### 7.1.11.3. Beispiele

**Alle Überschriften des Dokuments grau hinterlegen**

```
.sdoc-document h1, .sdoc-section {
    background-color: #f0f0f0;
}
```

#### 7.1.12. Quote

```
<blockquote class="sdoc-quote">
    TEXT
</blockquote>
```

##### 7.1.12.1. CSS

Selektor	Ziel
.sdoc-quote	Alle Zitat-Blöcke

#### 7.1.12.2. Beispiele

**Alle Zitate in kursiver Schrift (wie L<sup>A</sup>T<sub>E</sub>X)**

```
.sdoc-quote {
    font-style: italic;
}
```

#### 7.1.13. Style

Ein Style-Block fügt CSS-Code zum Dokument hinzu, ist selbst jedoch nicht visuell repräsentiert und daher auch nicht Ziel von CSS-Regeln.

#### 7.1.14. (Table)

#### 7.1.15. TableOfContents

##### 7.1.15.1. HTML

```
<div class="sdoc-tableofcontents">
    [<h3>TITLE</h3>]
    <ul [class="indent"]>
        <li>
            [<span class="number">NUMBER</span>] <a href="#ID">SECTION</a>
            ...
        </li>
        ...
    </ul>
```



```

    </ul>
  </div>

```

Die Überschrift TITLE ist optional. Die Abschnittsnummer NUMBER existiert nur bei nummerierten Abschnittsebenen.

#### 7.1.15.2. CSS

Selektor	Ziel
<code>.sdoc-tableofcontents</code>	Inhaltsverzeichnis insgesamt
<code>.sdoc-tableofcontents h3</code>	Überschrift Inhaltsverzeichnis
<code>.sdoc-tableofcontents ul</code>	Alle Abschnittsebenen
<code>.sdoc-tableofcontents &gt; ul</code>	Erste Abschnittsebene
<code>.sdoc-tableofcontents &gt; ul.indent</code>	Erste Abschnittsebene bei Einrückung
<code>.sdoc-tableofcontents ul ul</code>	Untergeordnete Abschnittsebenen
<code>.sdoc-tableofcontents li</code>	Eintrag
<code>.sdoc-tableofcontents span.number</code>	Abschnittsnummer

#### 7.1.15.3. Beispiele

##### Abschnittsummern in fetter Schrift

```

.sdoc-tableofcontents span.number {
    font-weight: bold;
}

```

##### Inhaltsverzeichnis als hervorgehobener Block

```

.sdoc-tableofcontents {
    display: inline-block;
    border: 1px solid black;
    background-color: #f0f0f0;
    padding: 4px 8px 4px 4px;
}

```

## 7.2. L<sup>A</sup>T<sub>E</sub>X

### 7.2.1. Reservierte Zeichen

In L<sup>A</sup>T<sub>E</sub>X gibt es besonders viele reservierte Zeichen. Die folgende Liste umfasst alle Eingabezeichen, die im Falle von L<sup>A</sup>T<sub>E</sub>X oder PDF als Zielformat von Sdoc gewandelt werden. Sie sollten in einem PDF in den Spalten “ohne WS” und “mit WS” korrekt dargestellt werden, einmal ohne und einmal mit Whitespace nach dem betreffenden Zeichen.

Zeichen	L <sup>A</sup> T <sub>E</sub> X	ohne WS	mit WS
Backslash	<code>\textbackslash{}</code>	<code>\x</code>	<code>\ x</code>
Dach	<code>\textasciicircum{}</code>	<code>^x</code>	<code>^ x</code>
Dollar	<code>\\$</code>	<code>\$x</code>	<code>\$ x</code>
Geschweifte Klammer auf	<code>\{</code>	<code>{x</code>	<code>{ x</code>
Geschweifte Klammer zu	<code>\}</code>	<code>}x</code>	<code>} x</code>
Größer	<code>\textgreater{}</code>	<code>&gt;0</code>	<code>&gt; 0</code>
Kaufmanns-Und	<code>\&amp;</code>	<code>&amp;x</code>	<code>&amp; x</code>
Kleiner	<code>\textless{}</code>	<code>&lt;0</code>	<code>&lt; 0</code>
Lattenkreuz	<code>\#</code>	<code>#x</code>	<code># x</code>
Pipe	<code>\textbar{}</code>	<code> x</code>	<code>  x</code>
Prozent	<code>\%</code>	<code>%x</code>	<code>% x</code>
Tilde	<code>\textasciitilde{}</code>	<code>~x</code>	<code>~ x</code>
Unterstrich	<code>\_</code>	<code>_x</code>	<code>_ x</code>

Tabelle 28: L<sup>A</sup>T<sub>E</sub>X Sonderzeichen

Es werden außerdem folgende Zeichenketten übersetzt:

Zeichenkette	L <sup>A</sup> T <sub>E</sub> X	ohne WS	mit WS
L <sup>A</sup> T <sub>E</sub> X	<code>\L<sup>A</sup>T<sub>E</sub>X{}</code>	<code>L<sup>A</sup>T<sub>E</sub>Xx</code>	<code>L<sup>A</sup>T<sub>E</sub>X x</code>
T <sub>E</sub> X	<code>\T<sub>E</sub>X{}</code>	<code>T<sub>E</sub>Xx</code>	<code>T<sub>E</sub>X x</code>

Tabelle 29: Speziell behandelte Zeichenketten

### 7.2.2. Paket array

CTAN: <https://ctan.org/pkg/array>

### 7.2.3. Paket babel

CTAN: <https://ctan.org/pkg/babel>

### 7.2.4. Paket caption

CTAN: <https://ctan.org/pkg/caption>

### 7.2.5. Paket float

CTAN: <https://ctan.org/pkg/float>

**7.2.6. Paket fontenc**

CTAN: <https://ctan.org/pkg/fontenc>

**7.2.7. Paket geometry**

CTAN: <https://ctan.org/pkg/geometry>

**7.2.8. Paket enumitem**

CTAN: <https://ctan.org/pkg/enumitem>

**7.2.9. Paket graphicx**

CTAN: <https://ctan.org/pkg/graphicx>

**7.2.10. Paket hyperref**

CTAN: <https://ctan.org/pkg/hyperref>

**7.2.11. Paket inputenc**

CTAN: <https://ctan.org/pkg/inputenc>

**7.2.12. Paket lmodern**

Wir verwenden das Paket `lmodern`, weil es statt der klassischen `Computer Modern` Schriften die qualitativ besseren `Latin Modern` Schriften auswählt. Dies sind die Schriften der drei Schriftfamilien “roman”, “sans-serif” und `typewriter`.

**7.2.13. Paket longtable**

CTAN: <https://ctan.org/pkg/longtable>

**7.2.14. Paket makecell**

CTAN: <https://ctan.org/pkg/makecell>

**7.2.15. Paket microtype**

CTAN: <https://ctan.org/pkg/microtype>

Wir verwenden das Paket `microtype`, um den Zeilenumbruch durch Anwendung von mikrotypografischen Methoden zu verbessern. Hierzu zählt ein optischer Randausgleich, sowie Wort- und Zeichendehnung.

**7.2.16. Paket minted**

CTAN: <https://ctan.org/pkg/minted>

**7.2.17. Paket quoting**

CTAN: <https://ctan.org/pkg/quoting>

Wir verwenden das Paket `quoting` für Zitate, da es eine einfache Kontrolle über den Font, die Einrückung und den vertikalen Leerraum über und unter Zitatabschnitten bietet.

**7.2.18. Paket scrlayer-scrpage**

CTAN: <https://ctan.org/pkg/scrlayer-scrpage>

**7.2.19. Paket showframe**

CTAN: <https://ctan.org/pkg/showframe>

**7.2.20. Paket varioref**

CTAN: <https://ctan.org/pkg/varioref>

Wir verwenden das Paket `varioref`, um “schlaue” Seitenverweise zu erzeugen. In Standard-L<sup>A</sup>T<sub>E</sub>X wird ein Seitenverweis mit `\pageref` erzeugt. Dieser Seitenverweis beschränkt sich auf die Seitennummer. Dies wirkt ein wenig absurd, wenn das Verweisziel auf der gleichen Seite liegt. Wir verwenden als Ersatz das Makro `\vpageref` aus dem Paket `varioref`. Dieses liefert keine Seitennummer, sondern eine komplette Formulierung wie z.B. “auf Seite 45”, abhängig von der Spracheinstellung, die über das Paket `babel` für das Dokument vorgenommen wurde. Das Makro erkennt, ob das Verweisziel auf der gleichen, der vorherigen oder der nächsten Seite liegt und erzeugt eine entsprechend schlaue Formulierung wie “auf dieser Seite” (siehe Abschnitt 2.2 - PDF auf Seite 7 und den zugehörigen Sdoc-Quelltext in Abschnitt 2.1 - Sdoc Quelltext auf Seite 6).

**7.2.21. Paket xcolor**

CTAN: <https://ctan.org/pkg/xcolor>

### 7.3. MediaWiki

MediaWiki ist ein eher umständlicher Wiki-Dialekt, der die meisten Sdoc-Konstrukte umsetzen kann, wenn auch mit Einschränkungen.

Da das Zielformat von MediaWiki ausschließlich HTML ist, ist die Wiki-Sprache durchsetzt mit HTML- und Stylesheet-Code.

#### 7.3.1. CSS

Globaler Stylesheet-Code ist nur möglich, wenn der Administrator dies zulässt. Auf der Seite selbst kann Stylesheet-Code portabel nur durch Setzung des style-Attributs definiert werden.

#### 7.3.2. Listen

Listen haben eine Reihe von Einschränkungen:

- Der Inhalt eines Listenelements kann nur ein einzelner Text (Paragraph) sein.
- Der Text muss einzellig sein.
- Listen können geschachtelt werden, was oft aber nicht besonders gut aussieht, da die Einrückung der verschiedenen Listenarten unterschiedlich ist. Insbesondere die Defintionsliste ist, wenn sie eingebettet ist, nicht tief genug.

#### 7.3.3. Einschränkungen

Z.Zt. noch nicht dokumentiert.

## 8. Sonstiges

### 8.1. 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.

### 8.2. 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
```

## 9. Programmierung

### 9.1. 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](#) auf Seite 12), 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 Instanzieren 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.

### 9.2. 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. Am Ende der Methode kommen zweimal die Charakter-Klassen vor, in die der Segment-Buchstabe eingetragen wird: `[ABCGILQS]`.

2. Wir erweitern die Character-Klasse in der Methode `Sdoc::Node::expandText()` um den Segment-Buchstaben. Dort wird die Expansion angestoßen.
3. Wir fügen die Behandlung des Segments zu den formatspezifischen Methoden `expandSegementsToHtml()`, `expandSegementsToTeX()`, `expandSegementsToMediaWiki()` usw. hinzu.

### 9.3. 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`.

### 9.4. Nutzer/Konfiguration/Knoten-Eigenschaft hinzufügen

Eine Nutzer/Konfiguration/Knoten-Eigenschaft zeichnet sich dadurch aus, dass ihr Wert in folgender Reihenfolge bestimmt wird:

1. Setzung per Option durch den Nutzer bei Aufruf von `sdoc`
2. Setzung im Dokument durch Knoten-Attribut
3. Setzung durch Eintrag in Konfigurationsdatei (`~/.sdoc.conf`)

Eine Nutzer/Konfiguration/Knoten-Eigenschaft wird hinzugefügt durch die Schritte:

1. Knoten-Eigenschaft hinzufügen (siehe [Knoten-Eigenschaft hinzufügen](#) auf dieser Seite)
2. Aufruf-Option zu `sdoc` hinzufügen
3. Konfigurationsvariable hinzufügen

Beispiel: Attribute `codeStyle`, `cssPrefix`



## A. Code Styles

Die Liste der Namen der verfügbaren Code-Styles liefert das Kommando

```
$ sdoc code-style-names
```

Eine HTML-Seite mit einer Darstellung aller Code-Styles für den Programmcode der Programmiersprache LANG in Datei FILE liefert das Kommando:

```
$ sdoc code-style-page LANG FILE
```

### A.1. Übersicht

**abap**

No description.

**algol**

No description.

**algol\_nu**

No description.

**arduino**

The Arduino® language style. This style is designed to highlight the Arduino source code, so expect the best results with it.

**autumn**

A colorful style, inspired by the terminal highlighting style.

**borland**

Style similar to the style used in the borland IDEs.

**bw**

No description.

**colorful**

A colorful style, inspired by CodeRay.

**default**

The default style (inspired by Emacs 22).

**emacs**

The default style (inspired by Emacs 22).

**friendly**

A modern style based on the VIM pyte theme.

**fruity**

Pygments version of the “native” vim theme.

**igor**

Pygments version of the official colors for Igor Pro procedures.

**lovelace**

The style used in Lovelace interactive learning environment. Tries to avoid the “angry fruit salad” effect with desaturated and dim colours.

**manni**

A colorful style, inspired by the terminal highlighting style.

**monokai**

This style mimics the Monokai color scheme.

**murphy**

Murphy's style from CodeRay.

**native**

Pygments version of the "native" vim theme.

**paraiso-dark**

No description.

**paraiso-light**

No description.

**pastie**

Style similar to the pastie default style.

**perldoc**

Style similar to the style used in the perldoc code blocks.

**rainbow\_dash**

A bright and colorful syntax highlighting theme.

**rrt**

Minimalistic "rrt" theme, based on Zap and Emacs defaults.

**tango**

The Crunchy default Style inspired from the color palette from the Tango Icon Theme Guidelines.

**trac**

Port of the default trac highlighter design.

**vim**

Styles somewhat like vim 7.0

**vs**

No description.

**xcode**

Style similar to the Xcode default colouring theme.