PeiTEX TEXnical Solutions
Marei Peischl
Prinz-Rupprecht-Straße 43
kontakt@peitex.de

Einführung in LaTeX 2ε

mit Berücksichtigung von KOMA-Script

20. April 2019

Über dieses Skript

Das Ihnen vorliegende Dokument basiert auf verschiedenen Kursen und wird seit 2011 kontinuierlich weiterentwickelt. Aufgrund dieser laufenden Aktualisierungen kann jedoch nicht garantieren, dass es sich um ein fehlerfreies Dokument handelt. Ich bin bei der Fehlersuche auf die Mithilfe der Leser und Workshopteilnehmer angewiesen und möchte Sie daher um Ihre Unterstützung bitten. Sobald Ihnen etwas unschlüssig ist oder Sie Ideen für Verbesserungen haben, wäre es sehr hilfreich, wenn Sie mir dies über kontakt@peitex.de mitteilen. Herzlichen Dank.

An einigen Stellen werden aufgrund der besseren Anschaulichkeit in Beispielen Makros verwendet, die im vorherigen Verlauf noch nicht besprochen wurden. Diese sind, um das Beispiel nicht noch komplizierter zu gestalten nicht besonders gekennzeichnet. In Schulungen wird explizit darauf hingewiesen und die Funktion kurz erläutert. Für Leser, die an keiner Schulung teilnehmen oder später nachschlagen, verfügt dieses Dokument über einen Index für Makros, der an diejenige Stelle des Skripts verweist, an der das Makro erläutert wird.

Regensburg, im April 2019 Marei Peischl

Inhaltsverzeichnis

0 Was ist LaTeX?
0.1 Word vs. LaTeX – Die Philosophie
0.2 Historischer Überblick zu TeX, LaTeX & Friends

I TeXnische Grundlagen 1

1 Die grundlegende Funktionsweise 3
1.1 Komponenten eines LaTeX-Systems . . . 3
1.2 Vom Quellcode zum Dokument . . . 4
1.3 Befehlsstruktur . . . 6
1.4 Die Eigenheiten von Satzprogrammen . . . 7
1.5 Die Struktur des Quellcodes . . . 9

2 Das erste LaTeX-Dokument 11

3 Der Aufbau von LaTeX-Dokumenten 13
3.1 Die Dokumentenklasse . . . 13
3.2 Ergänzungspakete . . . 20
3.3 Notwendige Anpassungen an System & Sprache . . . 22
3.3.1 Unicode Anpassungen für pdfLaTeX . . . 22
3.3.2 Sprachanpassung . . . 23
3.3.3 Anführungszeichen – Zitate . . . 25
3.3.4 Binde- & Gedankenstrich . . . 27
3.3.5 Akzente und Sonderzeichen . . . 27
3.3.6 Auslassungszeichen . . . 27
3.4 Titelei . . . 28
3.4.1 Automatische Titeleierzeugung mit `\maketitle` . . . 28
3.4.2 Frei gestaltbare Titelseite . . . 29
3.5 Vorspann, Hauptteil & Nachspann . . . 30
3.6 Gliederungsebenen . . . 30
3.6.1 Erweiterung der Gliederungsbefehle durch KOMA-Script . . . 32
3.6.2 Zusammenfassung . . . 32
3.6.3 Anhang . . . 33

3.7 Das Inhaltsverzeichnis . . . 33
3.8 Dokumente aufteilen . . . 33

4 Textformatierungen 37
4.1 Schriftarten und Textauszeichnung . . . 37
4.1.1 Unterschiede zwischen den Compilern . . . 37
4.1.2 Schriftattribute . . . 37
4.1.3 Schriftgrößen . . . 39
4.1.4 Textauszeichnung . . . 39
4.1.5 Globale Formatierungsänderungen für Elemente . . . 40
4.2 Umbrüche . . . 42
4.2.1 Absatzumbruch . . . 42
4.2.2 Zeilenumbruch . . . 43
4.2.3 Zeilenumbruch verhindern . . . 44
4.2.4 Zeilenabstand ändern . . . 44
4.2.5 Seitenumbruch . . . 44
4.2.6 Seitenumbruch verhindern . . . 45
4.2.7 Unsaubere Seitenumbrüche . . . 45
4.3 Trennung und Bindestrich . . . 45
4.3.1 Worttrennung . . . 45
4.3.2 Geschützte Leerzeichen . . . 46
4.3.3 Bindestrich . . . 47
4.4 Textausrichtung . . . 47
4.5 Farben - Das color-Paket . . . 48
4.6 Code „wörtlich“ ausdrucken . . . 48

5 Strukturautomatisierung 51
5.1 Zähler . . . 51
5.2 Längen und Zwischenräume . . . 53
5.2.1 Längen . . . 53
5.2.2 Zwischenräume . . . 55
5.2.3 Vertikale Abstände . . . 56
5.3 Eigene Befehle . . . 56
5.3.1 Eigene Makros mit Argumenten . . . 57
5.3.2 Eigene Umgebungen . . . 57
5.4 Querverweise . . . 58
5.4.1 Einfache Querverweise . . . 58
5.4.2 Fußnoten und Randnotizen . . . 58
5.4.3 Hyperlinks - Das hyperref-Paket . . . 60

II Das Seitenlayout **65**

6 Der Satzspiegel **67**
6.1 Satzspiegelkonstruktion mit typearea . . . 67
6.2 Das Seitenlayout manuell einstellen mit geometry . . . 70
6.3 Mehrspaltiger Textsatz . . . 70

7 Der Seitenstil **73**
7.1 Das Konzept der Seitenstile . . . 73
7.2 Kopf- und Fußzeilen mit scrlayer-scrpage . . . 73
7.2.1 Höhe von Kopf und Fuß . . . 74
7.2.2 Seitenstile modifizieren . . . 74
7.2.3 Formatierung der Kopf- und Fußzeilen . . . 76

III Weitere wichtige Elemente **79**

8 Aufzählungen **81**
8.1 Aufzählungslabels ändern . . . 81
8.2 KOMA-Auflistung . . . 81
8.3 Eigene Auflistungen und Listenlayouts . . . 83

9 Das Prinzip der Boxen **85**
9.1 Die drei Bearbeitungsmodi . . . 85
9.2 Rahmenboxen . . . 86
9.3 Absatzboxen . . . 87
9.4 Balkenboxen . . . 88
9.5 Boxen speichern . . . 88

10 Grafiken **91**

11 Tabellen **93**
11.1 Weitere Spaltenformatierungen – Das array-Paket . . . 94
11.2 Variable Spaltenbreite – Das tabularx-Paket . . . 95
11.3 Variable Spaltenbreite mit Ausrichtung – Das tabulary-Paket . . . 96
11.4 Verbesserte Tabellenlayouts – Das booktabs-Paket . . . 96
11.5 Mehrseitige Tabellen – Das longtable-Paket . . . 97

12 Gleitobjekte – Positionierung von Bildern und Tabellen **99**
12.1 Floating einschränken . . . 101
12.2 KOMA-spezielle caption-Einstellungen . . . 102
12.2.1 Position der caption . . . 102
12.2.2 Formatierung der caption . . . 103
12.3 Objekte von Text umfließen lassen – Das wrapfig-Paket . . . 104

13 Verzeichnisse **107**
13.1 Abbildungs- und Tabellenverzeichnis . . . 107
13.2 Verzeichnisse manipulieren . . . 107
13.3 Literaturverzeichnis . . . 108
13.3.1 Manuelle Erstellung des Literaturverzeichnisses . . . 108
13.3.2 Automatisches Literaturverzeichnis mit Biber und dem biblatex-Paket . . . 108
13.4 Index . . . 114

14 Formeln und Einheiten **117**
14.1 Typografische Regeln . . . 117
14.2 Schriftattribute . . . 118
14.3 Schriftstile . . . 119
14.4 Zeilenmodus vs. abgesetzter Modus . . . 119
14.4.1 Der Zeilenmodus . . . 120
14.4.2 Der abgesetzte Modus . . . 120
14.5 Referenz und Bezug . . . 122
14.6 Mathematische Akzente . . . 123
14.7 Exponenten und Indizes . . . 123
14.8 Brüche und Binomialkoeffizienten . . . 124
14.9 Wurzeln . . . 125
14.10 Operatoren . . . 125
14.11 Spezielle Buchstaben und Zeichen . . . 127
14.12 Klammern . . . 128
14.13 Matrizen . . . 129
14.14 Overset und Underset . . . 130
14.15 Text im Mathemodus . . . 130
14.16 Theoreme . . . 131
14.17 Werte und Einheiten mit siunitx . . . 131
14.17.1 Werte & Einheiten . . . 132
14.17.2 Wertetabellen . . . 133
14.18 Das mhchem-Paket . . . 134
14.19 Kommutative Diagramme . . . 135
14.19.1 Die Struktur der Diagramme . . . 135
14.19.2 Pfeile . . . 136

0 Was ist LaTeX?

LaTeX ist eine Sammlung von Makros für das Textsatzprogramm TeX, welches für den Satz qualitativ hochwertiger Dokumente verwendet wird. Am meisten Verbreitung findet es im naturwissenschaftlichen und technischen Umfeld. Dies ist historisch begründet, da LaTeX und auch reines TeX von Anfang an mathematische Notationen unterstützte. Allerdings ist es für sämtliche Dokumententypen geeignet.

0.1 Word vs. LaTeX – Die Philosophie

Textverarbeitungsprogramme gliedern sich in verschiedene Sparten:

- Wortprozessoren[1] (z. B. Microsoft Word): Eine Formatierungseigenschaft oder Formatvorlage wird für einen Textbereich ausgewählt,und der formatierte Text erscheint genau so auf dem Bildschirm.

- Desktop-Publishing-Systeme (z. B. Adobe InDesign): Mehr Möglichkeiten im Vergleich zu Wortprozessoren mit dem Ziel, professionelle Druckvorlagen für Veröffentlichungen zu erstellen. Diese Programme nutzen ebenfalls hauptsächlich grafische Benutzeroberflächen.

- Textsatzsysteme (z. B. TeX): Diese Programme verfügen über *keine*[2] eigene grafische Oberfläche. Hier bearbeitet der Nutzer einen Quelltext (Code), welcher neben dem Inhalt auch die Dokumentenstruktur enthält. Dieser wird vom Programm interpretiert, welches entsprechend der Layouteinstellungen daraus eine Ausgabedatei (üblicherweise eine `.pdf`-Datei) erzeugt.

Die Konzepte haben ihre eigenen Vorteile, und die Wahl der Software verbleibt letzten Endes immer beim Anwender. Den wesentlichsten Unterschied stellt die Beziehung zwischen Dokumenteninhalt und Formatierung dar. Bei TeX-basierten Systemen erhält der Inhalt ein sogenanntes Markup, eine semantische Auszeichnung. Im Code wird nicht direkt angegeben, wie ein Element formatiert werden soll, sondern der Textbereich erhält eine Kennzeichnung. Zum Beispiel kann eine Textzeile entweder Fließtext oder eine Überschrift, eine Beschriftung, o. Ä. sein. Das Layout wird beim übersetzen in eine `.pdf`-Datei den Einstellungen entnommen.

Diese Vorgehensweise ermöglicht dem Nutzer ohne viele Anpassungen ein einheitliches Layout zu erstellen, in dem Elemente des gleichen Typs für das gesamte Dokument gleich behandelt werden. Somit wirken Dokumente ohne großen Aufwand professionell, verfügen über eine klare Struktur und sind für den Leser leichter nachvollziehbar.

[1] Oft auch Textverarbeitungssysteme, allerdings ist diese Bezeichnung im Vergleich den anderen irreführend.
[2] Es gibt Systeme, die über eine Verfügen, allerdings ist diese nicht Teil des Konzeptes.

Das System hat außerdem den Vorteil, dass es ohne großen Aufwand möglich ist, ein Layout auf verschiedene Dokumente zu übertragen. Nachträgliche Formatierungsänderungen und Wiederverwendbarkeit einzelner Elemente stellen keinerlei Hürde dar. Somit kann man aus demselben Code (durch andere Einstellungen) eine Präsentation und ein gedrucktes Skript oder eine Aufgabenstellung in verschiedenen Varianten (z. B. mit oder ohne Lösungsvorschlag) erzeugen.

LaTeX liefert dem Benutzer einige Basislayouts für die geläufigsten Dokumententypen, welche bereits sämtliche Ansprüche für einfache Text-Dokumente erfüllen. Diese Basis lässt sich durch andere Layouts und Ergänzungspakete erweitern. Somit lassen sich auch komplexere Dokumente (z. B. mit mathematischen Formalismen oder komplexen Tabellen) mit deutlich weniger Aufwand zu realisieren, als dies für Wortprozessoren der Fall ist.

Die Code-Schnittstelle ermöglicht zusätzlich eine einfache Versionierung. Darüber wird einerseits das kollaborative Arbeiten an gemeinsamen Projekten vereinfacht, andererseits existieren Möglichkeiten um auf vorherige Versionen zurückzuspringen und Vergleiche zwischen unterschiedlichen Versionen durchzuführen.

Bei LaTeX wird wie in der Softwareentwicklung meistens das Versionsverwaltungssystem git[3] eingesetzt. Hierfür existieren auch Web-Services wie GitHub[4] oder GitLab[5], die für OpenSource-Projekte kostenlos nutzbar sind.

0.2 Historischer Überblick zu TeX, LaTeX & Friends

TeX (Großschreibung von τεχ; Tau Epsilon Chi[6] – gesprochen „tech“) und METAFONT wurden ab 1977 von Donald E. Knuth (mittlerweile emeritierter Professor der Stanford University) „zum Satz schöner Bücher und insbesondere Bücher, die viel Mathematik enthalten“ [7] und der Erzeugung besonderer Zeichensätze entwickelt. Der direkte Umgang mit dem sogenannten plain TeX ist jedoch relativ kompliziert und erfordert einige Erfahrung. Leslie Lamport hat aus diesem Grund in den 1980er Jahren das Programmpaket LaTeX (LaTeX steht für LamportTeX) entwickelt. Es greift mithilfe von Makros auf die ursprünglichen Deklarationen in TeX zurück und stellt somit eine nutzerfreundlichere Möglichkeit dar, TeX zu benutzen.

Die Entwicklung der nächsten Jahrzehnte lief in viele unterschiedliche Richtungen, in deren Rahmen Programme wie NFSS, SLiTeX, $\mathcal{AMS}$-LaTeX entstanden. Ab 1993 entwickelte das LaTeX3 Project Team die jetzt „aktuelle“ Version LaTeX 2_ε[7] und schuf somit einen neuen Standard.

Heute gibt es drei häufig verwendete Varianten: pdfLaTeX, LuaLaTeX und XeLaTeX. Alle drei erzeugen mit Standardeinstellungen eine `.pdf`-Datei. Tabelle 0.1 zeigt die Unterschiede und Besonderheiten. Derzeit ist der am weitesten verbreite Compiler nach wie vor pdfLaTeX. Da aufgrund der einfacheren Entwicklung viele Neuerungen zunächst in LuaLaTeX implementiert werden, ist bei wichtigeren und größeren Objekten LuaLaTeX zu empfehlen. Damit ist zusätzlich eine größere Zahl an Schriftarten nutzbar, was gerade Anpassungen nach Vorgaben erheblich vereinfacht.

[3] `https://git-scm.com/`
[4] `https://github.com/`
[5] `https://about.gitlab.com/`
[6] vom altgriechischen τέχνη was in etwa Fähigkeit, Kunst, Handwerk bedeutet
[7] Das LaTeX3 Project (https://www.latex-project.org/) arbeitet nach wie vor kontinuierlich an Erweiterungen.

Tabelle 0.1: Unterschiedliche LaTeX-Compiler im Vergleich. (Stand 03/2019). Am weitesten ist nach wie vor pdfLaTeX verbreitet, allerdings steigt der Anteil der LuaLaTeX-Nutzer stetig. Hauptgrund ist die Vereinfachung bei der Schriftverarbeitung und die native UTF-8-Kodierung.

Feature	LaTeX/pdfLaTeX	LuaLaTeX	XeLaTeX
Ausgabe	DVI/PDF	PDF	PDF
UTF-8, Abschnitt 3.3.1	–/unvollständig	✓	✓
Schriften	komplizierte Installation einzelner Schriftarten	Systemschriften verwendbar, insbesondere auch OTF	wie LuaLaTeX zuzüglich spezieller Font Formate
Grafikformate	EPS/JPG,PNG,PDF	JPG,PNG,PDF	wie LuaLaTeX zzgl. EPS, JPEG2000, Adobe Illustrator
microtype-Unterstützung, Abschnitt 3.3.2	✓	✓	teilweise
Besonderheiten		eingebundener Lua-Interpreter	spezielle Schriftformate und -systeme

Zusätzlich zu den unterschiedlichen Varianten des Kompilierungsprogrammes gibt es noch eine Reihe von Zusatzprogrammen, die oft als „Friends“ zusammengefasst werden. Hierzu gehören Programme zur Literaturverzeichniserstellung (z. B. biber), Generatorprogrammen für Stichwortverzeichnisse oder Glossare (z. B. xindy), sowie Programme, die den Kompilierungsprozess durch Automatisierung vereinfachen (z. B. latexmk). Diese Programme werden üblicherweise zusammen mit dem Basissystem als sogenannte „TeX-Distribution“ installiert.

Wichtige Hinweise, bevor es los geht

Verwendung von Vorlagen

Es gibt jede Menge Vorlagen für LaTeX. Leider ist die Vielfalt so groß, dass sich darunter auch viel Veraltetes oder gar Unbrauchbares befindet. Sollten Sie unbedingt mit einer Vorlage arbeiten wollen, sehen Sie sich unbedingt die geladenen Pakete durch, wofür diese verwendet werden und ob Sie diese wirklich benötigen.

Außerdem sollten Sie darauf achten, dass bei den System- und Sprachanpassungen keine anderen Optionen verwendet werden als die in diesem Kurs behandelten, siehe Abschnitt 3.3. Vorlagen, die diesen Ansprüchen nicht genügen, können zumeist als veraltet angesehen werden. Sollten Sie unschlüssig sein, ob Ihre Vorlage brauchbar ist, hilft peiTeX Ihnen gerne bei dieser Einschätzung und unterstützt auch bei der Aktualisierung oder sonstigen Anpassungen.

Umgang mit Fehlermeldungen

Die Standardeinstellung der meisten TeX-Editoren[8] führt dazu, dass beim Kompiliervorgang trotz auftretender Fehler eine auf den ersten Blick zufriedenstellende Ausgabe erzeugt wird. Das verleitet leider dazu, Fehlermeldungen zu ignorieren. *Davon sollte in jedem Fall abgesehen werden!*

Fehlermeldungen geben Auskunft darüber, dass der Programmdurchlauf nicht wie erwartet verlaufen ist und irgendetwas, sei es auch noch so eine Kleinigkeit, nicht sauber funktioniert hat. Dies kann zu „komischem“ Verhalten an einer ganz anderen Stelle des Dokuments führen. Im schlimmsten Fall kann sogar Dokumenteninhalt „verschluckt“ werden.

Umgang mit Warnungen

Warnungen geben Aufschluss darüber, dass etwas anders ist als von LaTeX erwartet oder dass das Programm, weil es sich nicht zu helfen wusste, etwas „Unschönes“ getan hat. In jedem Fall ist eine Überprüfung notwendig. Erst danach kann man zweifelsfrei sagen, ob die Warnung ignoriert werden kann oder ob manuelles Eingreifen notwendig ist.

Warnungen gibt es zum Beispiel, wenn Markierungen fehlen und trotzdem auf sie referenziert wird oder LaTeX, weil es die Trennstellen eines Wortes nicht oder nur ungenügend kennt, über den Satzspiegel in den Rand hinaus schreibt.

[8] Meistens ist der „nonstopmode“ aktiviert. Das bedeutet, dass das Programm bei einem Fehler trotzdem weiterläuft. In den Programmeinstellungen kann man diese Einstellung auch ändern, indem man beim Aufruf von `pdflatex` die Option `-interaction=nonstopmode` entfernt.

Teil I

TEXnische Grundlagen

1 Die grundlegende Funktionsweise

1.1 Komponenten eines LaTeX-Systems

Um LaTeX sinnvoll benutzen zu können, braucht man (prinzipiell) nicht nur ein Programm, sondern *drei*:

Editor Hier wird der Text mit den zugehörigen Formatierungsbefehlen eingegeben. Es genügt ein einfacher Texteditor, z. B. Notepad++, gedit, …

Allerdings ist es gerade für Anfänger oft einfacher, einen speziellen LaTeX-Editor zu benutzen, da er unter Anderem auch bei der Fehlersuche hilft.
Beispiele für kostenfreie Editoren, die für alle gängigen Betriebssysteme zur Verfügung stehen: TeXstudio, TeXmaker, TeXworks Erfahrenere Programmierer und Entwickler haben häufig bereits ein Lieblings-Werkzeug (vim, Emacs, Eclipse, …). Hierfür existieren entsprechende Plugins um die LaTeX-Unterstützung zu aktivieren, in diesen Fällen ist kein gesonderter Editor notwendig. Diese Werkzeuge benötigen jedoch allgemein Einarbeitungszeit. Den Umgang gleichzeitig mit den ersten LaTeX-Erfahrungen zu erlernen kann daher frustrierend werden.

Distribution Die Distribution ist der Kern eines TeX-Systems. Sie enthält das eigentlichen Übersetzungsprogramme, sowie sogenannte Formate. Der Übersetzer wandelt den Quellcode unter Berücksichtigung der Formatierungsbefehle der Formate in eine druckbares Ausgabedokument (heute meistens eine `.pdf`-Datei) um. Diesen Prozess nennt man Kompilierung. Die durchführenden Programme heißen somit Kompilierungsprogramme oder Compiler.

Damit man dieses Programm effizient nutzen kann, enthält die Distribution noch deutlich mehr als nur das Übersetzungsprogramm. Da LaTeX für sehr viele und auch sehr unterschiedliche Bereiche verwendet wird, kann ein Programm alleine nicht genau auf die Bedürfnisse des Nutzers abgestimmt sein. Dafür existieren in LaTeX unterschiedlichste Zusatzpakete und auch kleinere Zusatzprogramme, welche die Funktionalität erheblich erweitern und somit eine nutzerfreundliche Schnittstelle für nahezu jedes Anwendungsgebiet zur Verfügung stellen.
Die beiden üblichsten Distributionen sind heute: TeX Live (für Mac OS heißt das Paket MacTeX), sowie MiKTeX. Bis vor kurzem, gab es MiKTeX nur für Windows. Mittlerweile ist es ebenfalls für alle Betriebssysteme verfügbar. Die Basisfunktionität ist für beide Distributionen gleich, allerdings unterscheiden sich die Lizenzen und dadurch die enthaltenen Pakete geringfügig.

Viele Linux-Distributionen verfügen über eigene Pakete für TeX Live. Diese sind häufig jedoch nicht auf dem neuesten Stand. Es ist daher abzuwägen, ob diese Pakete genügen oder man die neuesten Pakete nutzen möchte. In letzterem Fall besteht immer die Möglichkeit eine Installation aus den CTAN-Ressourcen durchzuführen.

Betrachter Wird benötigt, um das erzeugte Dokument anzusehen bzw. um es auszudrucken. Bei `.pdf`-Dateien wäre das z. B. der Adobe Reader. Bei manchen Editoren, wie TEXstudio oder TEXmaker ist ein solches Programm bereits integriert. Jedoch eignen sich diese eingebetteten Versionen meistens nur für eine grobe Kontrolle (Aufgrund fehlender Details in der Anzeige.). Für die abschließende Endkontrolle sollten sie nicht verwendet werden.
Allgemein verfügt das `.pdf`-Format über deutlich mehr Funktionalität als die bloße Dokumentendarstellung[1], jedoch unterstützen nicht alle Anzeigeprogamme sämtliche Features.

1.2 Vom Quellcode zum Dokument

Bei der Umwandlung in das Ausgabeformat wird in der Regel mehr als nur die Ausgabedatei erstellt. Diese zusätzlich erzeugten Dateien sind Hilfsdateien, die entweder vom Compiler für die Erstellung von Querverweisen oder Verzeichnissen oder von zusätzlichen Programmen benötigt werden. Je nach Komplexität des Dokuments kann die Zahl dieser Hilfsdateien rasch ansteigen.

Um dennoch den Überblick zu behalten, ist es empfehlenswert für ein neues Dokument einen eigenen Ordner (ggf. mit Unterordnern für die bessere Übersicht) anzulegen.

Die Quelldateien für LATEX haben überlicherweise die Dateiendung `.tex` und enthalten den sogenanngen Quellcode. Der einfachste Quellcode für ein solches Dokument lautet in etwa:

```
\documentclass{scrartcl}
\begin{document}
    Mein erstes Dokument!
\end{document}
```

Ist der Code gespeichert, übergibt man ihn an das Satzprogramm, welches den Quellcode interpretiert. Es ruft gegebenenfalls auch weitere externe Programme auf, die zum Beispiel zur Erstellung einer Bibliografie notwendig sind. Abschließend erstellt es die Ausgabe:

```
Mein erstes Dokument!
```

Falls man mit einem speziellen LATEX-Editor arbeitet, wird der Übersetzungsprozess meistens über einen einzigen Tastendruck gestartet. Allerdings ist es nach wie vor möglich, direkt über eine Kommandozeile zu kompilieren.Die notwendigen Schritte lauten:

1. Code im Editor schreiben und abspeichern. Für dieses Beispiel heißt die Datei `Code.tex` und ist in einem Ordner names `LaTeX` auf dem Desktop gespeichert.

2. Dokument übersetzen mit `pdflatex`: In der Kommandozeile[2] an den Speicherort der Datei navigieren und dann `pdflatex` ausführen:

[1] Es ist sogar möglich Videos oder Animationen einzubinden oder interaktive Formulare in diesem Format zu erzeugen.
[2] Je nach Betriebssystem wird das Programm auch *Eingabeaufforderung* (Windows) oder *Terminal* (Mac OS oder Linux) genannt.

Tabelle 1.1: Bedeutung der Dateiendungen der wichtigsten Hilfsdateien

Endung	*Erläuterung*
`.tex`	LaTeX oder TeX Input-Format. Kann kompiliert werden.
`.sty`	LaTeX-Style: Ergänzungspaket. Erweitert die Funktionalität, kann mit `\usepackage` geladen werden, siehe auch Abschnitt 3.2.
`.cls`	LaTeX-Class:. Dokumentenklasse. Sie legt den Dokumententyp fest, siehe auch Abschnitt 3.1.
`.dvi`	Device independent file format: Das Ausgabeformat der ursprünglichen Programmvariante `latex`.
`.log`	Protokoll: Detaillierte Informationen über den letzten LaTeX-Durchlauf. Enthält Warnungen und Fehlermeldungen.
`.toc`	Speichert alle Abschnittsüberschriften für den Eintrag in das Inhaltsverzeichnis.
`.lof`	Analog `.toc` für Abbildungsverzeichnis.
`.lot`	Analog `.toc` für Tabellenverzeichnis.
`.aux`	Hilfsdatei, welche die zugehörigen Informationen zu Querverweisen enthält.

Kommandozeile

```
cd ~/Desktop/LaTeX
        pdflatex dokument.tex
```

Dabei werden das `.pdf`-Dokument und einige zusätzliche Hilfsdateien erzeugt. Siehe auch Tabelle 1.1 für genauere Erläuterungen zu den Dateitypen.

3. Erneutes Übersetzen des Dokuments wie in Schritt 2. Dies ist nötig, damit Verzeichnisse wie das Inhaltsverzeichnis erstellt und Querverweise gesetzt werden (siehe auch Abschnitt 3.7).

4. *Optional:*
   Je nachdem, welche Zusatzelemente verwendet werden (Bibliografie, Stichwortverzeichnis, …) ist es notwendig, mehrere Kompilierungsvorgänge durchzuführen und ggf. auch weitere Programme zur starten (z. B. um die Literaturangaben zu sortieren). Wenn diese Elemente besprochen werden, wird auch auf diesen Punkt genauer eingegangen.

5. `.pdf`-Dokument ansehen.

In der Praxis benutzt man den Weg über die Kommandozeile nur noch selten. Stattdessen arbeitet man mit einem speziellen Editor. Ein Grund dafür ist, dass bei diesem der Quellcode dank farbiger Hervorhebungen (Syntaxhighlighting) wesentlich übersichtlicher wird. Zudem wird das Speichern des Quellcodes, die Übergabe an das Satzprogramm und das anschließende Öffnen der `.pdf`-Datei meist mit einem einzigen Tastendruck erledigt. Allerdings sollte man sich des klassischen Weges gerade bei der Fehlersuche bewusst bleiben. Der Vorteil besteht darin, dass `pdflatex` sofort stoppt, wenn es über ein Problem stolpert. Dies kann manchmal dabei helfen, Fehler genauer zu lokalisieren.

Es ist jedoch auch möglich, die Editor-Schnittstelle so zu konfigurieren, dass auch dort bei jedem Fehler sofort abgebrochen wird.

1.3 Befehlsstruktur

Einzelne Elemente werden durch Befehle als solche gekennzeichnet. Die Befehle hierfür heißen bei LaTeX Makros. Es gibt eine Vielzahl unterschiedlicher Makros, die sich jedoch auf drei grundlegende Typen zurückführen lassen:

- Sonderzeichen; Die Zeichen \, {, }, #, $,&, _, ^, ~, und % haben besondere Bedeutungen und fungieren direkt als Befehle (vgl. Tabelle 1.2).
- Einzelne Sonderzeichen hinter einem Backslash (vgl. ebenfalls Tabelle 1.2). Diese Makros werden meistens zum Ausdruck der Sonderzeichen benutzt, zum Beispiel \$ ⟶ $.
- Die klassischen Makros bestehen aus Backslash und einer Buchstabenfolge. Das erste folgende andere Zeichen wird als Befehlsende interpretiert. Die genaue Struktur und Verwendung wird im Folgenden genauer erläutert.

`\Befehl`

Dies ist die einfachste Form von Befehlen, sie werden mit \ eingeleitet und benötigen keine weiteren Argumente. Ein Beispiel hierfür ist das LaTeX-Logo: `\LaTeX`⟶LaTeX

Anmerkung: Sämtliche LaTeX Makros sind „case sensitive“. Das bedeutet, dass die Groß- und Kleinschreibung beachtet werden muss z. B. `\LaTeX` ≠ `\latex`

`\Befehl{`*Argument*`}`

Befehle mit Argumenten ermöglichen es, Makros an die genaue Verwendung anzupassen oder die Wirkung des Befehls auf einen kurzen Textabschnitt einzugrenzen. Das Argument wird durch geschweifte Klammern begrenzt und darf nicht weggelassen werden[3]. Bei den meisten Makros mit Argumenten ist das auch nachvollziehbar, da diese Makros ohne das Argument keinen Sinn ergeben. (Das Beispiel benötigt das Paket color oder xcolor, siehe auch Abschnitt 4.5.)

```
\color{gray}
Dieser Text ist grau.
```

Dieser Text ist grau.

Ohne eine Angabe der Farbe wäre der Befehl `\color` nicht sinnvoll. Außerdem erkennt man, dass der Befehl ein sogenannter Schalter ist. Er bewirkt, dass alles, was danach folgt, grau geschrieben wird.

Um die Wirkung eines solchen Schalters einzugrenzen, kann man in den meisten Fällen Gruppierungen mithilfe geschweifter Klammern benutzen:

[3] Ggf. ist es aber möglich, die Klammern leer zu lassen.

```
{\color{gray}
    Dieser Text ist grau.}\\
Dieser Text ist schwarz.
```

Dieser Text ist grau.
Dieser Text ist schwarz.

Die meisten Befehle[4] wirken nur innerhalb des Bereiches, indem sie ausgeführt werden. Ihre Einstellungen gelten nur bis zum Abschluss der aktuellen Gruppe und werden dann wieder auf den Zustand vor der Gruppe zurückgesetzt.

`\Befehl[`*optionales Argument*`]{`*Argument*`}`

Außerdem gibt es Befehle mit optionalen Argumenten. Diese funktionieren wie einfache Befehle mit Argumenten, allerdings ist hier ein Standardwert hinterlegt für den Fall, dass kein Argument angegeben wird. Ein Beispiel hierfür sind Wurzeln im Mathemodus[5]:

```
$\sqrt{2}\\
\sqrt[3]{2}$
```

$\sqrt{2}$
$\sqrt[3]{2}$

`$\sqrt{2}$` liefert hierbei das gleiche Ergebnis wie `$\sqrt[]{2}$`. Die meisten Makros sind so konzipiert, dass leere eckige Klammern komplett entfallen können.

Umgebungen

`\begin{`*Umgebungsname*`}`...`\end{`*Umgebungsname*`}`

Umgebungen funktionieren wie Befehle. Sie können ebenfalls notwendige, sowie optionale Argumente besitzen. Umgebungen werden meistens zum Umschalten von Textmodi benutzt (Abschnitt 9.1) oder um Befehle einzugrenzen, wie es bei manchen Schaltern mit Hilfe von Gruppierungen möglich ist. So erzeugen die folgenden beiden Codebeispiele das gleiche Ergebnis:

```
{\color{gray}
    Dieser Text ist grau.}
Dieser Text ist schwarz.
```

```
\begin{color}{gray}
    Dieser Text ist grau.
\end{color}
Dieser Text ist schwarz.
```

Dieser Text ist grau. Dieser Text ist schwarz.

1.4 Die Eigenheiten von Satzprogrammen

Manche Eingaben werden in Satzprogrammen, also auch bei LaTeX, grundlegend anders interpretiert, als man es von herkömmlichen Wortprozessoren kennt. Das liegt daran, dass bei LaTeX bestimmte Sonderzeichen als Steuerzeichen benutzt werden:

[4] Die Ausnahmen nennt man globale Makros. Sie werden meistens für Deklarationen benutzt. Diese Deklarationen bleiben bestehen, bis sie von einer Deklaration des gleichen Typs überschrieben werden. Bei der später folgenden Erläuterungen globaler Makros wird auch direkt auf diese Eigenheit hingewiesen.

[5] Die Dollarzeichen vor und nach dem Ausdruck dienen dazu, in den Mathemodus zu wechseln, da das Makro `\sqrt` nur innerhalb dieses Modus verwendet werden kann (vgl. Kapitel 14).

Leerzeichen TEX und somit auch LATEX interpretiert Leerzeichen im Quellcode als Wort- oder Befehlsende. Treten mehrere Leerzeichen auf, werden sie wie ein einzelnes behandelt. Die Wortabstände setzt das Programm automatisch entsprechend der Satzmethode. Im Standardmodus (Blocksatz) sind die Abstände somit in einem gewissen Maß variabel.

Leer- oder Tabulatorzeichen zu Beginn einer Codezeile werden ignoriert. Einrückungen, die zur besseren Übersichtlichkeit des Codes, führen sind somit möglich und durchaus sinnvoll.

Leerzeichen, die ein Makro beenden, werden nicht gedruckt, dies kann bei Nichtbeachtung zu fehlenden Leerzeichen führen, z. B.:

`Dieser Satz wurde mit \LaTeX gesetzt.`	Dieser Satz wurde mit LATEXgesetzt.

Abhilfe kann man dadurch schaffen, dass man entweder das Makro durch eine leere Gruppe (`{}`) beendet oder das Leerzeichen explizit eingibt (`\␣`).

```
Dieser Satz wurde mit \LaTeX{} gesetzt.
Dieser Satz wurde mit LaTeX\ gesetzt.
```

Dieser Satz wurde mit LATEX gesetzt. Dieser Satz wurde mit LaTeX gesetzt.

Zeilenumbrüche Einfache Zeilenumbrüche werden von LATEX äquivalent zu Leerzeichen interpretiert. Wichtig ist es jedoch zwischen Zeilenumbrüchen und Leerzeilen zu unterscheiden.

Leerzeilen LATEX interpretiert Leerzeilen im Quellcode nicht nur als Wort- bzw. Befehls-, sondern auch als Absatzende. Treten mehrere Leerzeilen auf, so werden sie analog zu den Leerzeichen als eine Einzige angesehen. Entsprechend werden die Absatzabstände von LATEX intern verwaltet und sind (bei Benutzung der Standardeinstellungen) variabel.

Sonderzeichen Sie stellen spezielle Befehle dar und müssen, wenn sie als Text ausgegeben werden sollen, über andere Befehle angesprochen werden. Tabelle 1.2 zeigt die Sonderzeichen, welche als Befehl interpretiert werden, samt ihrer Bedeutung und den zugehörigen Befehlen für die Textausgabe.

Kommentare Da das Konzept von LATEX sich damit befasst, dem Text eine logische Struktur zu geben, kann es häufig notwendig sein, sich Notizen zu eigenen Makros oder den eingebundenen Paketen zu machen. Auf diese Weise kann man vermeiden, dass zu Beginn des Dokumentes erst einmal hundert unnötige Definitionen gemacht werden. das Prozentzeichen (%) ist das Kommentarzeichen bei TEX und LATEX. Das Programm ignoriert alles was in einer Zeile hinter einem Prozentzeichen steht. Somit kann man Beschriftungen vornehmen, die nicht in der Ausgabe erscheinen sollen.

Durch geschickte Nutzung des Kommentarzeichens ist es auch möglich den Code übersichtlicher zu gestalten, indem man z. B. „eine Leerzeile auskommentiert“. Dadurch erhält man eine fast leere Zeile, die für bessere Übersicht sorgt und keinen Absatzumbruch erzeugt.

Tabelle 1.2: Bedeutung der Sonderzeichen in LaTeX und ihre Eingabebefehle.

Zeichen	*Bedeutung*	*Eingabe*
\	Makro-/Befehlsbeginn	`\textbackslash`
{	Beginn einer Gruppe	`\{`
}	Ende einer Gruppe	`\}`
#	Parameter	`\#`
$	Mathe-Zeilenmodus	`\$`
&	Trennzeichen bei Matrizen und Tabellen	`\&`
_	Subscript im Mathemodus (Index)	`\_`
^	Superscript im Mathemodus (Exponent)	`\textasciicircum`
~	Geschütztes Leerzeichen	`\textasciitilde`
%	Leitet Kommentare ein	`\%`

1.5 Die Struktur des Quellcodes

Der formale Aufbau eines LaTeX-Dokuments gliedert sich in zwei Teile:

```
\documentclass[Optionen]{Name}
...
```
Präambel

```
\begin{document}
...
\end{document}
```
Dokumenten-körper

Die Präambel – oft auch Header genannt – enthält alles, was für das gesamte Dokument gelten soll, also globale Definitionen und Einstellungen. Dabei können unter anderem auch Definitionen und Befehle überschrieben, beziehungsweise neu definiert werden, die bereits in der Dokumentenklasse (siehe Abschnitt 3.1) festgelegt wurden.

Der Teil der von der `document`-Umgebung eingeschlossen wird heißt Text-, oder Dokumentenkörper, wird oft auch nur als Body bezeichnet. Er enthält den eigentlichen Text mit lokalen Formatierungsbefehlen. `\begin{document}` startet die Erzeugung der Ausgabe. Das Ende dieser Umgebung (`\end{document}`) beendet das Compiler-Programm. Dies bedeutet, dass alles, was hinter dem Ende der `document`-Umgebung steht, von LaTeX nicht mehr beachtet wird und eine Fehlermeldung ausgegeben wird, falls das Programm keinen Befehl zum Abschluss des Dokuments findet.

Es gibt Befehle, die nur innerhalb der Präambel bzw. nur im Textkörper stehen sollten. Entweder, weil sie im jeweils anderen Bereich keinen Sinn ergeben (in diesem Fall führt falsche Positionierung zu Fehlermeldungen) oder weil die Reihenfolge bei manchen Makros eine Rolle spielt. Innerhalb dieses Dokumentes wird speziell auf Befehle eingegangen, bei denen die Position von Bedeutung ist. Wird nichts angemerkt, gehört das Makro in den Textkörper.

2 Das erste LaTeX-Dokument

Nun geht es zum praktischen Teil: Dem ersten eigenen Dokument. Beispiel 1 zeigt, wie ein einfaches Dokument aussieht. Um nun genau zu verstehen welcher Befehl wofür wichtig ist, zeigt es außerdem eine Untergliederung des Quellcodes nach seinen verschiedenen Funktionen. Im Folgenden werden nun die Funktionen und Eigenschaften der einzelnen Makros betrachtet.

Beispiel 1: Ein einfaches Beispieldokument

[1] Dieses Paket hat lediglich praktische Gründe, da es an der Stelle des Befehls `\blindtext` einen Text zum Testen von Dokumenten ausgibt, sonst jedoch keinerlei Bedeutung.

3 Der Aufbau von LaTeX-Dokumenten

3.1 Die Dokumentenklasse

Die Dokumentenklasse bestimmt den Dokumententyp und nimmt die grundlegenden Einstellungen vor. Die Auswahl der Klasse ist üblicherweise der erste Schritt für die Erstellung eines neuen Dokuments:

`\documentclass[`*Option1, Option2,...*`]{`*Klasse*`}`

Da alle weiteren Einstellungen von dieser Auswahl abhängen, sollte die Dokumentenklasse immer als allererstes gewählt werden. Üblicherweise sogar in der ersten aktiven Codezeile. Das Kompilierungsprogramm liest dann die Konfiguration aus der entsprechenden `.cls`-Datei und stellt damit die Grundstruktur für das Dokument zur Verfügung.

Standardklassen vs. KOMA-Script
KOMA-Script ist eine Sammlung von Klassen und Paketen für LaTeX, die sei mehr als 20 Jahren kontinuierlich weiterentwickelt und erweitert wird. Sie verbessert nicht nur die Funktionalität und Nutzerfreundlichkeit, vielmehr ermöglicht Sie es auch mit geringem Aufwand die örtlichen typografischen Gepflogenheiten zu berücksichtigen.

Die KOMA-Klassen (Dokumentenklassen aus dem KOMA-Script-Bundle) ermöglichen viele sonst sehr komplexe Anpassungen durch vorgefertigte Elemente. Wenn es eine Möglichkeit gibt, die gewünschte Anpassung mit den Elementen von KOMA-Script durchzuführen, so ist dies in jedem Fall anderen Möglichkeiten vorzuziehen, da die Möglichkeiten, die KOMA-Script bietet alle aufeinander abgestimmt sind.

Die Haupt-KOMA-Klassen sind scrbook, scrartcl, scrreprt, scrlettr2 (korrespondierend zu den Standardklassen book, article, report und letter). Die sogenannten Standardklassen sind deutlich einfacher strukturiert und sind ein Basisbaustein von LaTeX. Sie richten sich jedoch mit ihren Einstellungen nach US-amerikanischen Konventionen für Typografie und Papierformat. Zudem verfügen sie über deutlich weniger vorgefertigte Anpassungsmöglichkeiten als die KOMA-Varianten.

Die Benutzung der KOMA-Klassen ist außerdem aufgrund der möglichen typografischen Feinanpassungen und deutlichen Erweiterung der Auszeichnungssprache LaTeX sinnvoll. Sie sind bei weitem flexibler und bieten deutlich mehr Bedienkomfort als die Standardklassen. Die KOMA-Script-Klassen sind sehr gut dokumentiert (sogar zweisprachig: [9] & [10] und zusätzlich in Buchform [8]), sodass ein Blick in die Dokumentation oft hilfreich ist!

Um für ein Dokument die richtige Dokumentenklasse wählen zu können, empfiehlt es sich einen Blick auf die grundlegenden Eigenschaften zu werfen. Die korrespondierenden Standardklassen sind jeweils in Klammern angegeben. Die hier gezeigten grundlagenden Eigenschaften stimmen im Wesentlichen überein.

scrbook (book) Für große Schriftwerke im Stil von Vorlesungsmitschriften, Büchern, sowie Dissertationen.

- Titelei auf eigener Seite (oder mehreren)
- doppelseitiges Layout (Unterscheidung innerer/äußerer Rand anstatt links/rechts)
- Besitzt alle Ebenen[1]
- Kapitel beginnen immer auf neuer rechter Seite
- Seitenzählung mit römischen Ziffern (Vorspann und arabischen Ziffern (Hauptteil & Nachspann), siehe auch Abschnitt 3.5 `\part`, `\chapter`, `\section`, …
- Nummerierung von Abbildungen, Tabellen und Gleichungen mit vorangestellter Kapitelnummer (z. B. 1.1)
- Nummerierung der Fußnoten wird jedes Kapitel neu begonnen

scrartcl (article) Für kleinere Werke im Stil von Artikeln, Kurzberichten oder Referaten.

- Titel auf keiner eigenen Seite
- einseitiges Layout
- Abschnitte werden direkt untereinander gesetzt (ohne zusätzliche Seitenumbrüche)
- Seitennummerierung durchgehend in arabischen Ziffern
- Verfügt nicht über die Ebene[1] `\chapter`
- Fortlaufende Nummerierung von Elementen (z. B. Abbildungen, Tabellen, Fußnoten und Gleichungen)

scrreprt (report) Diese Klasse stellt eine Zwischenform dar. Sie wird meistens für Abschlussarbeiten gewählt, da diese zwar ein Kapitel als Gliederungsebene benötigen, jedoch oft nicht doppelseitig gedruckt werden.

- Titelei auf einer Seite (wie scrbook)
- einseitiges Layout (wie scrartcl)
- Besitzt alle Ebenen (wie scrbook)
- Kapitel beginnen immer auf neuer (jedoch nicht gezwungenermaßen auf einer rechten) Seite
- Keine Grobgliederung möglich, daher durchlaufende Seitennummerierung (wie scrartcl)
- Nummerierung von Elementen mit vorangestellter Kapitelnummer (wie scrbook)

scrlettr2 (letter) Für Briefe. Diese Dokumentenklasse hat keinerlei Ebenen, allerdings spezielle Briefelemente, wie zum Beispiel: Absender, Anschrift, Betreff, Anlagen, …

[1] Die einzelnen Ebenen und die dazugehörigen Befehle `\section`, `\part` usw. werden in Abschnitt 3.6 erläutert.

Tabelle 3.1: Die wichtigsten Dokumentenklassenoptionen für KOMA-Script. Sie werden als Option zum Makro `\documentclass` gesetzt.

Bedeutung	**`Option`** = *Wert* (mögliche Werte)	**Beispiel & Erklärung**
Entwurfsmodus	`draft` = `true` oder `false`	`draft=false`[2] Bei überlangen Zeilen werden am Zeilenende kleine, schwarze Kästchen ausgegeben. Dies erleichtert dem ungeübten Auge zu sehen, wo eine manuelle Nachbearbeitung nötig ist. Diese Option beeinflusst auch viele Ergänzungspakete, beispielsweise deaktiviert sie sämtliche Funktionen von hyperref.
Papierformat	`paper` = `letter`, `legal`, `executive`, `aX`, `bX`, `cX`, `dX`, `landscape`, `seascape`, `portrait`	`paper=a4`[2] `X` ist durch 0, 1, …,8 zu ersetzen. Es werden sämtliche ISO-Formate unterstützt. Zusätzlich kann die Ausrichtung angegeben werden: `landscape` entspricht Querformat, `seascape` dem um 180° gedrehtem Querformat und `portrait` dem Hochformat.
Titelseite (Abschnitt 3.4.1)	`titlepage` = `true`, `false` oder `firstiscover`	`titlepage=false` Schaltet bei Verwendung von `\maketitle` zwischen Titelkopf und Titelseiten um.
Nummerierung (Abschnitt 3.6)	`numbers` = `auto`, `enddot` oder `noenddot`	`numbers=noenddot` Objektnummerierung mit (4. oder 4.1.) oder ohne (4 oder 4.1) Punkt am Ende. Diese Option bezieht sich auf alle nummerierten Objekte (Überschriften, Tabellen, Bilder, …)
Größe der Überschriften (Abschnitt 3.6)	`headings` = `big`, `normal` oder `small`	`headings=normal`[2]

[2]Wert entspricht dem Standardwert

Fortsetzung auf der nächsten Seite

Tabelle 3.1: (Fortsetzung)

Bedeutung	**Option** = *Wert* (mögliche Werte)	**Beispiel & Erklärung**
Größe der Überschriften (Abschnitt 3.6)	`headings` = `big`, `normal` oder `small`	`headings=normal`[2]
Formatierung der Überschriften (Abschnitt 3.6)	`headings` = `onelineappendix`, `onelinechapter`, `twolineappendix`, `twolinechapter`	`headings=onelineappendix` Ein-/Zweizeilige Kapitel-/Anhangsüberschriften
Option bei Abschnittsmakros (Abschnitt 3.6)	`headings` = `optiontohead`, `optiontoheadandtoc`, `optiontotoc`	`headings=optiontoheadandtoc`[2] Bestimmt wofür das optionale Argument von Gliederungsbefehlen verwendet werden soll.
Kapitelanfang (Abschnitt 3.6)	`open` = `any`, `right` oder `left`	`open=right` Nur bei `twoside=true` sinnvoll. Der Kapitelanfang wird ggf. durch eine Vakatseite (Seite die absichtlich leer bleibt) verzögert, damit Kapitel z. B. immer auf ungeraden (rechten) Seiten beginnen.
Zusammenfassung (Abschnitt 3.6.2)	`abstract` = `true` oder `false`	`abstract=true` Zusammenfassung mit oder ohne die Überschrift „Zusammenfassung“.
Inhaltsverzeichnis (Abschnitt 3.7)	`toc` = `listof`, `index`, `bib`, `flat`, …	toc=graduated[2] Einstellung von Form und Inhalt des Inhaltsverzeichnisses. Die Möglichkeiten für Werte werden später erläutert.

[2]Wert entspricht dem Standardwert

Fortsetzung auf der nächsten Seite

Tabelle 3.1: (Fortsetzung)

Bedeutung	**Option** = *Wert* (mögliche Werte)	**Beispiel & Erklärung**
Schriftgröße (Abschnitt 4.1.3)	`fontsize` = `10pt`, `11pt`, `12pt`	`fontsize=11pt`[2] Es sind auch andere Angaben der Schriftgröße möglich, allerdings muss der Anwender in diesem Fall eine Datei `.clo`-Datei zur Verfügung stellen, in der er LaTeX erklärt wie die angegebene Größe zu verwenden ist (Zusatzpakete).
Absatzkennzeichnung (Abschnitt 4.2.1)	`parskip` = `full`, `half*`, `false`, `never`, ...	`parskip=full` Wählt die Methode mit der ein Absatzumbruch gekennzeichnet werden soll. Alle möglichen Werte werden später in Tabelle 4.4 (Seite 43) genauer erklärt.
doppelseitiger Druck (Kapitel 6)	`twoside` = `true` oder `false`	`twoside=true` Buchdrucklayout. Die Ränder unterscheiden sich hierbei nach äußerem und innerem Rand. Ungerade Seitenzahlen werden den rechten Seiten zugeordnet.
Bindekorrektur (Abschnitt 6.1)	`BCOR` = *Länge*	`BCOR=0mm`[2] Wert muss absolut und mit Einheit angegeben werden.
Teilungsverhältnis (Abschnitt 6.1)	`DIV` = 4, 5, ..., `calc`, `classic`, `areaset`, `last` oder `default`	`DIV=11` Gibt an in wie viele Streifen der Satzspiegel eingeteilt werden soll.
Kopfzeile zum Satzspiegel rechnen (Abschnitt 6.1)	`headinclude` = `true` oder `false`	`headinclude=false`[2] Die Kopfzeile wird als Teil des Satzspiegels betrachtet. Wird nur bei Satzspiegeleinstellungen durch KOMA-Script (typearea) beachtet.

[2]Wert entspricht dem Standardwert

Fortsetzung auf der nächsten Seite

Tabelle 3.1: (Fortsetzung)

Bedeutung	**Option** = *Wert* (mögliche Werte)	**Beispiel & Erklärung**
Fußzeile zum Satzspiegel rechnen (Abschnitt 6.1)	`footinclude` = `true` oder `false`	`footinclude=false`[2] Die Fußzeile wird als Teil des Satzspiegels betrachtet. Wird nur bei Satzspiegeleinstellungen durch KOMA-Script (typearea) beachtet.
Fußnoten (Abschnitt 5.4.2)	`footnotes` = `multiple` oder `nomultiple`	`footnotes=nomultiple`[2] Aktiviert bzw. deaktiviert die Ausgabe eines Trennzeichens (Komma) zwischen direkt hintereinanderliegenden Fußnoten.
zweispaltiger Textsatz (Abschnitt 6.3)	`twocolumn` = `true` oder `false`	`twocolumn=false`[2]
Seitenvorschub (Kapitel 7)	`cleardoublepage` = *Seitenstil*	`cleardoublepage=empty`[2] Bestimmt den Seitenstil von Vakatseiten, also den Seiten, die beim Satz absichtlich leer bleiben.
Kopfzeilen (Kapitel 7)	`headlines` = *Anzahl der Zeilen* `headheight` = *Höhe*	`headlines=1.25`[2] Anzahl der Zeilen, die für die Kopfzeile frei gehalten werden sollen. Der Standard von 1,25 Zeilen bietet in der Regel genug Platz für einzeilige Kopfzeilen und Trennlinien. Alternativ kann die Höhe auch direkt als Länge angegeben werden.

[2]Wert entspricht dem Standardwert

Fortsetzung auf der nächsten Seite

Tabelle 3.1: (Fortsetzung)

Bedeutung	**Option** = *Wert* (mögliche Werte)	**Beispiel & Erklärung**
Fußzeilen (Kapitel 7)	`footlines` = *Anzahl der Zeilen* `footheight` = *Höhe*	`footlines=1.25`[2] Anzahl der Zeilen, die für die Fußzeile frei gehalten werden sollen. Der Standard von 1,25 Zeilen bietet in der Regel genug Platz für einzeilige Fußzeilen und Trennlinien. Alternativ kann die Höhe auch direkt als Länge angegeben werden.
Kopflinie (Kapitel 7)	`headsepline` = `true` oder `false`	`headsepline=false` Trennt, falls gewünscht, die Kopfzeile vom Textkörper durch eine Linie ab.
Fußlinie (Kapitel 7)	`footsepline` = `true` oder `false`	`footsepline=true` Trennt, falls gewünscht, die Fußzeile vom Textkörper durch eine Linie ab.
Linienausrichtung (Kapitel 7)	`ilines`, `clines`, `olines`	`ilines` Die Trennlinie zwischen Text und Kopf, bzw. zwischen Text und Fuß wird innen bündig (`ilines`), zentriert (`clines`) oder außen bündig (`olines`) gesetzt.
Objektbeschriftung (Kapitel 12)	`captions` = `bottombeside`, `above`, ...	`captions=centeredbeside` Positionierung und Formatierung der `captions`. Die Werte werden später in Tabelle 12.2 genauer erläutert.

[2] Wert entspricht dem Standardwert

KOMA-Script bietet auch eine riesige Menge an Dokumentenklassenoptionen zur leichten Anpassung der Basiseigenaschaften. Diese werden als durch Kommata gretennte Liste in das optionale Argument von `\documentclass` gesetzt. Der wichtigste Teil dieser Optionen findet sich in Tabelle 3.1 (auf den Seiten 15–15). Einige darin auftauchende Begriffe werden jedoch erst im weiteren Verlauf genauer besprochen. Die Verweise zu den jeweiligen Abschnitten sind angegeben.

3.2 Ergänzungspakete

Die große Zahl an Nutzern mit unterschiedlichen Bedürfnissen, denen LaTeX gerecht werden möchte resultiert in einer riesigen Menge an Ergänzungspaketen. Sie liefern Befehle und/oder verändern das Dokumentenlayout auf verschiedenste Weise. Normalerweise installiert man zusammen mit der Distribution automatisch die am weitesten verbreiteten Pakete. Sie müssen somit lediglich geladen werden:

`\usepackage[`*Option1*`,`*Option2*`,...]{`*Paket*`}` in der Präambel

Pakete können nur innerhalb der Präambel geladen werden. `\usepackage` muss somit zwischen `\documentclass` und `\begin{document}` stehen.

Im Laufe dieses Kurses werden wir einige wichtigere Pakete genauer betrachten. Tabelle 3.4 zeigt die diejenigen, die hier behandelt werden mitsamt einer kleinen Beschreibung.

Um zu verhindern, dass unnötige Pakete geladen werden, sollte man sich gegebenenfalls Notizen machen, welches Paket für was gebraucht wird und überflüssige Pakete auskommentieren. Das vermeidet erstens Fehlerquellen, die in der Wechselwirkung von Paketen entstehen können und verringert außerdem die Kompilierungszeit.

Globale und lokale Optionen

Zusätzlich zu den Dokumentenklassenoptionen kann man bei der Angabe der Dokumentenklasse auch globale Optionen festlegen. Die dort gewählten Optionen gelten für *alle* Packages, die geladen werden. Man kann und sollte Optionen, die von mehreren Paketen verarbeitet werden können, als Dokumentenklassenoption angeben. Insbesondere gilt dies für die Optionen `ngerman` und `draft`.

Paketanleitungen finden

Für speziellere Anwendungen oder zum Nachschlagen bestimmter Makros empfiehlt sich häufig ein Blick in die Paketanleitung. Die richtige Paketanleitung zur installierten Version findet man über das Programm `texdoc`, welches Bestandteil jeder LaTeX-Distribution ist. Die üblichste Verwendungsweise funktioniert über die Eingabe von

Kommandozeile

```
texdoc Paketname
```

und anschließendem ↵ in der Kommandozeile. Dieser Befehl sucht nach einer Dokumentation zum angegebenen Paketnamen und öffnet sie gegebenenfalls. Sollte keine Anleitung vorhanden sein, wird eine entsprechende Meldung ausgegeben.

Da sich manche Anwender mit grafischen Anwendungen leichter vertraut machen als mit Kommandozeilenprogrammen, existiert unter TeX Live zusätzlich ein sogenanntes GUI (engl: „graphical user interface“) für `texdoc`. Das Programm `texdoctk` (unter Windows auch „texdoc GUI“) lässt sich ebenfalls über die Suchfunktion finden. Dort werden die Pakete auch nach Kategorien sortiert aufgelistet. Unter Windows wird es mit TeX Live automatisch mit installiert.

Tabelle 3.4: Die Ergänzungspakete, die in diesem Kurs besprochen werden

array	Weitere Spaltenformatierungen für Tabellen; Abschnitt 11.1
babel	Sprachunterstützung, Trennregeln,...; Abschnitt 3.3.2
booktabs	Verbessertes Spacing und Linien bei Tabellen; Abschnitt 11.4
fontenc	Ausgabekodierung und Vektorschrift; Abschnitt 3.3.1
color	Ermöglicht die Verwendung von Farben für LaTeX-Elemente; Abschnitt 4.5
csquotes	Kontextsensitiver Satz von Anführungszeichen; Abschnitt 3.3.3
enumitem	Erweiterte Möglichkeiten im Umgang mit Aufzählungen; Abschnitt 8.3
geometry	Manuelle Manipulation des Seitenlayouts; Abschnitt 6.2
graphicx	Einbinden von Bildern; Kapitel 10
hyperref	Hyperlinks und erweiterte PDF-Fuktionalitäten; Abschnitt 5.4.3
inputenc	Eingabekodierung; Abschnitt 3.3.1
lmodern	Verbesserte Fassung der Schriftart Computer Modern; Abschnitt 3.3.1
longtable	Mehrseitige Tabellen; Abschnitt 11.5
microtype	Verbessertes Trennverhalten durch Berücksichtigung der Mikrotypografie; Abschnitt 3.3.2
multicol	Mehrspaltiger Textsatz;Abschnitt 6.3
multirow	Tabellenzellen über mehrere Tabellenzeilen; Kapitel 11
placeins	Einfache Möglichkeit um den Bewegungsbereich von Gleitobjekten einzuschränken; Abschnitt 12.1
ragged2e	Modifizierter Flattersatz mit Worttrennungen; Tabelle 4.4
scrlayer-scrpage	Modifikationen des Seitenstils; Abschnitt 7.2
setspace	Zeilenabstand ändern; Abschnitt 4.2.4
siunitx	Zahlen und Einheiten typographisch korrekt setzen; Abschnitt 14.17
tabularx	Tabellen mit fester Breite über die dehnbare X-Spalte; Abschnitt 11.2
tabulary	Tabellen die sich entsprechend des Textinhaltes der Spalten auf eine bestimmte Breite dehnen lassen (L-, C-, R-, J-Spaltentypen); Abschnitt 11.3
typearea	Professionelle Satzspiegelkonstruktion; Abschnitt 6.1
wrapfig	Objekte von Text umfließen lassen; Abschnitt 12.3

Auf anderen Betriebssystemen sind ggf. zusätzliche Installationen notwendig um das grafische Oberfläche nutzen zu können.

3.3 Notwendige Anpassungen an System & Sprache

Ein großer Vorteil von LaTeX ist seine Systemunabhängigkeit und, dass es für fast alle Sprachen nutzbar ist. Früher musste man dafür einige systemspezifische Anpassungen vornehmen. Heute ist es dank universeller Kodierungen möglich, eine Konfiguration für alle Systeme zu nutzen.

Für XeLaTeX und LuaLaTeX ist UTF-8 die native Kodierung. Hier sind keinerlei Anpassungen notwendig. Auch die Schriftsysteme bauen auf dieser Kodierung auf.

Für pdfLaTeX war es bis Anfang 2018 notwendig die Kodierung manuelle anzugeben. Mittlerweile genügt es das richtige Schriftformat zu laden. Da UTF-8 in pdfLaTeX derzeit (Anfang 2019) noch über Makros verwirklicht wird, ist es hier nicht möglich Umlaute in Makros oder Labels zu verwenden. Für Fließtext stellt dies jedoch in den meisten Fällen kein Problem dar.

pdfLaTeX	LuaLaTeX/XeLaTeX

`\documentclass[ngerman]{scrartcl}`
`%System älter als Frühjahr 2018`
`%\usepackage[utf8]{inputenc}`
`\usepackage[T1]{fontenc}`
`\usepackage{babel}`
 	
`\begin{document}`
`Dokumenteninhalt`
`\end{document}`
	`\documentclass[ngerman]{scrartcl}`
	
	`\usepackage{babel}`
	
	`\begin{document}`
	`Dokumenteninhalt`
	`\end{document}`

Offensichtlich entfallen für LuaLaTeX & XeLaTeX einige Schritte. Da pdfLaTeX jedoch nach wie vor weiter verbreitet ist und häufig von Vorlagen verwendet werden muss, werden diese Anpassungen dennoch besprochen.

3.3.1 Unicode Anpassungen für pdfLaTeX

Für die Verwendung von Unicode unter pdfLaTeX sind grundsätzlich zwei Schritte notwendig. Bei Versionen von pdfLaTeX seit Frühjahr 2018 wird der erste (Abschnitt 3.3.1) automatisch ausgeführt. Der zweite ist bisher (03/2019) nach wie vor unabdingbar.

Da immernoch ältere Distributionen im Umlaut sind, sollte grundsätzlich beachtet werden. Allerdings genügt es in der Regel Nutzer auf die Notwendigkeit eine neuere Version zu installieren hinzuweisen.

Eingabekodierung

Der Begriff „Eingabekodierung“ beschreibt, wie Zeichen im Quellcode abgespeichert werden. Für Text ist die Eingabekodierung wichtig, sobald man über de ASCII-Zeichensatz, also einfache Buchstaben und ein paar wenige Sonderzeichen hinausgeht. Alte Systeme sind in der Zeichenanzahl stark beschränkt und wurden somit oft abhängig von der Sprache definiert. Im Deutschen sind beispielsweise Umlaute oder auch Sonderzeichen, wie die Anführungszeichen („“) davon betroffen.

UTF-8 ist glücklicherweise mittlerweile der Standard für Textkodierung. Um diesen bei pdfLaTeX zu aktivieren ist das inputenc notwendig. Seit Frühjahr 2018 wird diese Einstellung

automatisch erledigt, für ältere Versionen ist es nach wie vor notwendig UTF-8 explizit auszuwählen:

```
\usepackage[utf8]{inputenc}
```

Um zu prüfen ob das Paket notwendig ist, kann man zu Beginn des Dokuments ein paar Umlaute eingeben. Wenn diese richtig dargestellt werden, ist die Voreinstellung korrekt.

Im Internet oder auch älteren Vorlagen findet man öfter andere Kodierungen. Sie funktionieren zwar auf den entsprechenden Systemen nach wie vor, jedoch sollte aus Kompatibilitätsgründen davon abgesehen werden. Selbiges gilt für die Option `utf8x` zu inputenc. Sie basiert auf einem Paket, dass nicht mehr weiter entwickelt wird.

Sollte man mit einer alten Vorlage arbeiten wollen/müssen, die eine andere Kodierung enthält, ist zu beachten, dass eine Änderung auf UTF-8 nicht durch das Paket durchgeführt wird. Die Quelldatei selbst muss geändert werden.

Wichtig ist auch, dass sich die Eingabekodierung erst mit dem Laden des Paketes ändert. Umlaute sollten somit erst nach dem Ladevorgang benutzt werden.

Schriftkodierung und erweiterer Zeichensatz

Ähnlich wie bei den Eingabekodierungen gibt es auch für die Schriftkodierungen und die verwendeten Zeichensätze verschiedene Varianten. Die Anpassungen sind auch nur bei pdfLaTeX notwendig, da die Problematik historisch bedingt ist.

Die Voreinstellung ist noch noch nicht an UTF-8 angepasst. In Kombination mit der voreingestellten Schriftsippe *Computer Modern* führt das dazu, dass nur ein beschränkter Zeichensatz zur Verfügung steht.

Hier lädt man die Kodierung `T1`. Sie sorgt zusätzlich dafür, dass ein „ä“ auch wirklich ein Umlaut und kein Buchstabe mit zwei Punkten als Akzent gesetzt wird. Das notwendige Paket fontenc verbessert durch die Korrekte Kodierung auch die Trennung von Wörtern mit Umlauten:

```
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

Die Schriftart Latin Modern, die über das Paket lmodern geladen wird ist nicht zwingend notwendig. Allerdings sollte eine Schriftart geladen werden, die in als `T1` vorliegt.

3.3.2 Sprachanpassung (Das babel-Paket)

Um den Satz eines Absatzes möglichst perfekt zu gestalten, benutzt TeX einen sehr effektiven Algorithmus um die Wörter möglichst gleichmäßig zu verteilen und somit unschöne Löcher im Blocksatz zu verhindern. Ein Teil dieses Vorgangs beschäftigt sich auch mit der Trennung von Wörtern am Zeilenende. Jedoch gelten in den unterschiedlichen Sprachen unterschiedliche Regeln. Daher muss eingestellt werden, welche Regeln benutzt werden sollen.

Außerdem ermöglicht eine Sprachanpassung, dass Elemente, mit automatischer Benennung (Verzeichnisse, Abbildungen, …) richtig beschriftet werden. Über dem Inhaltsverzeichnis steht dann „Inhaltsverzeichnis“ anstatt „Table of Contents“.

Für XeLaTeX und LuaLaTeX existiert neben dem babel-System auch das Paket polyglossia. Da dieses jedoch auf diese beiden Systeme beschränkt ist und die Entwicklung bei babel wieder stark vorangeschritten ist, wird in diesem Dokument nicht weiter darauf eingegangen.

Babel verfügt über verschiedenste Sprachen und über Mechanismen um innerhalb des Dokumentes die Sprache zu wechseln. Die Sprache wird, wie auch die Kodierung, mithilfe einer Paketoption ausgewählt:

```
\usepackage[ngerman]{babel}
```

ngerman steht hierbei für „new german“, also neue deutsche Rechtschreibung. Es gibt jedoch auch weitere Pakete, die eine Sprachanpassung zulassen, vor allem Pakete, die mit den Verzeichnissen, insbesondere dem Literaturverzeichnis oder dem Index arbeiten. Somit ist es sinnvoll, die Option ngerman nicht als Paketoption zu übergeben, sondern als globale Dokumentenoption (vgl. Abschnitt 3.2). Somit kann jedes Paket, dass diese Option verarbeiten kann, darauf zugreifen.

Für anders- oder mehrsprachige Dokumente gibt es äquivalente Optionen. Die zuletzt geladene Option entspricht dabei der Hauptsprache. Für mehr Übersichtlichkeit, kann babel jedoch über die Option main=*Hauptsprache* auch direkt die Hauptsprache mitgeteilt werden. Ein Dokument auf deutsch mit englischen Passagen benötigt somit die Einstellungen:

```
\documentclass[ngerman,. . . ]{scr. . . }
\usepackage[english,main=ngerman]{babel}
```

Innerhalb des Dokumentes kann dann die Sprache global

```
\selectlanguage{Sprache}
```

oder lokal

```
\foreignlanguage{Sprache}{Text}
```

geändert werden.

Die Funktionen von babel werden jedoch erst mit Beginn des Dokumentenkörpers (\begin{document}) aktiviert. Alle inhaltlichen Texteingaben, für die das Verhalten von babel eine Rolle spielt, sollten daher bis zum Beginn des Dokumentenkörpers hinausgezögert werden (vgl. Position der Makros für die Titelei, Abschnitt 3.4.1).

Leerzeichen nach Satzzeichen

Neben den Trennregeln unterscheiden sich die beiden Sprachen Deutsch und Englisch auch durch die Leerzeichen nach einem Satzendepunkt. Mit Standardeinstellungen ist bei LaTeX der Zwischenraum nach einem Punkt minimal größer als ein normaler Wortzwischenraum. Im deutschen Sprachraum sollte dieser Abstand jedoch den übrigen Wortzwischenräumen entsprechen (sogenanntes \frenchspacing). Das babel-Paket setzt auch diese Einstellungen automatisch für die gewählte Sprache.

Soll dieser zusätzliche Zwischenraum nur in einzelnen Fällen unterdrückt werden, zum Beispiel bei Abkürzungen, kann dies durch das Einfügen eines expliziten Leerzeichens (\␣) bewerkstelligt werden:

nur falls \nonfrenchspacing (z. B. Englisch)

```
lower case abbreviations, e.\,g. like this one.\\
lower case abbreviations, e.\,g.\ like this one.
```

lower case abbreviations, e. g. like this one.
lower case abbreviations, e. g. like this one.

Punkte die auf Großbuchstaben folgen, werden als Abkürzungen interpretiert. Hier folgt in beiden Fällen ein normaler Wortzwischenraum. Um LaTeX im Fall von \nonfrenchspacing einen echten Satzendepunkt zu erhalten, muss man diesem die Zeichenkombination \@ voranstellen. Zum Beispiel:

nur falls \nonfrenchspacing (z. B. Englisch)

```
Yesterday, I saw my G.P. Tomorrow I'm going to see the specialist.\\
Yesterday, I saw my G.P\@. Tomorrow I'm going to see the specialist.
```

Yesterday, I saw my G.P. Tomorrow I'm going to see the specialist.
Yesterday, I saw my G.P. Tomorrow I'm going to see the specialist.

Der zusätzliche Zwischenraum bei `\nonfrenchspacing` wird auch bei anderen Satzendezeichen (?, !, :) eingefügt. Hier kann das jedoch auf die selbe Weise wie bei einem Punkt verhindert (`\`) oder ermöglicht (`\@`).

Verbessertes Trennverhalten (Das Paket microtype)
Um ein noch besseres Trennverhalten der Wörter und zusätzliches Kerning[3] zu erhalten, empfiehlt es sich außerdem, vor allem bei längeren und wichtigeren Arbeiten zusätzlich das Paket microtype über

```
\usepackage{microtype}
```

einzubinden. Es verbessert den Trennalgorithmus um die Möglichkeiten der Mikrotypografie (optischer Randausgleicher, Wort- und Zeichendehnung) und vermeidet somit viele unschöne Zeilenumbrüche und überlange Zeilen.

Anführungszeichen und Umlaute
Eine weitere Eigenschaft des babel-Paketes ist der Satz von Anführungszeichen und Umlauten. Ein Überbleibsel aus der Zeit vor UTF-8, als man Umlaute noch nicht direkt eingeben konnte, ist dabei, dass Zeichenfolgen wie „"U“ als „Ü“ ausgegeben werden. Unter anderem ist es aus diesem Grund nicht sinnvoll das Zollzeichen " als Anführungszeichen zu verwenden. Für den korrekten Satz deutscher und englischer Anführungszeichen siehe Abschnitt 3.3.3.

3.3.3 Anführungszeichen – Zitate

Anführungszeichen stehen prinzipiell vor und hinter [vgl. 16, S.29]:

- einer wörtlich wiedergegebenen Äußerung (direkte Rede),
- einer wörtlich angeführten Textstelle (Zitat)[4]
- zitierten Überschriften, Titeln von Büchern, Filmen, Gedichten, Namen von Zeitungen und ähnlichem,
- Wortschöpfungen und Worten, die im übertragenen Sinne gemeint sind (Metaphern),
- einzelnen Wortteilen, Wörtern oder Textteilen, die hervorgehoben werden sollen.

„Lange Zitate werden nicht in Anführungszeichen eingeschlossen, sondern eingerückt mit einer `quote`- oder `quotation`-Umgebung gesetzt“[16, S.30], siehe Abschnitt 4.4.

Zitate aus Fremdsprachen werden ebenfalls in Anführungszeichen der Dokumentensprache gesetzt. Lediglich in dem Fall, dass das Zitat selbst fremdsprachliche Anführungszeichen enthält, sind diese zu übernehmen.

[3] Kerning bedeutet, dass die Abstände zwischen Buchstaben angepasst werden, je nachdem um welche Buchstaben es sich handelt

[4] Bei Zitaten ist zudem zu beachten, dass sie immer wörtlich wiederzugeben sind. Sie dürfen weder im Wortlaut noch in Rechtschreibung und Interpunktion vom Original abweichen. Eigene Korrekturen und Ergänzungen zum Zitat sind duch eckige Klammern zu kennzeichnen

Zudem bleibt zu sagen, dass bei allen Schriften in OT1-Kodierung das Kerning (die Unterscheidung verschiedener Abstände) zwischen Anführungszeichen und nachfolgenden bzw. vorangehenden Zeichen fehlt. Dies spricht wieder für die Wahl der T1-Kodierung (Abschnitt 3.3.1) und für erweitertes Kerning auch das Paket microtype (Abschnitt 3.3.2).

Eingabe von Anführungszeichen

Unter Verwendung der Eingabekodierung UTF-8 ist es möglich Anführungszeichen direkt einzugeben. Allerdings sind hierfür auf Standard-Tastaturen komplexere Kombinationen notwendig, die zusätzlich noch vom Betriebssystem abhängig sind. Daher existieren in LaTeX Abkürzungen (sog. „shorthands“), um die Eingabe zu vereinfachen. Die englischen Varianten sind immer aktiv. Um zusätzlich die Kombinationen für deutsche Anführungszeichen zu aktivieren ist das babel-Paket notwendig.

Deutsche Anführungszeichen (benötigt babel):

	Gänsefüßchen:		Spitze Form[5]:	
doppelt öffnend:	`\glqq` oder `"‘`	„	`\frqq`, `">` oder `>>`	»
doppelt schließend:	`\grqq` oder `"’`	“	`\flqq`, `"<` oder `<<`	«
einfach öffnend:	`\glq`	,	`\frq`	›
einfach schließend:	`\grq`	‘	`\flq`	‹

Englische Anführungszeichen:

doppelt öffnend:	‘‘	“
doppelt schließend:	’’	”
einfach öffnend:	‘	‘
einfach schließend:	’	’

Das Paket csquotes

Das Paket csquotes vereinfacht die Eingabe der Anführungszeichen erheblich. Es ermöglicht unter anderem mit dem Makro

`\enquote{`*Zitatinhalt*`}`

kontextabhängige und durch babel an die Sprache angepasste Zitatkennzeichnung. Folgendes Beispiel zeigt die Unterschiede:

> Die Zeitung schrieb: „Die Bahn hat bereits im Frühjahr erklärt: ‚Wir haben die feste Absicht, die Strecke stillzulegen‘ und sie hat das auf Anfrage gestern noch einmal bestätigt.“

Mit den bisher vorgestellten Varianten gibt es nun mehrere Möglichkeiten die Anführungszeichen zu erzeugen: Zunächst nur mit dem babel-Paket:

```
Die Zeitung schrieb: \glqq{}Die Bahn hat bereits im Frühjahr erklärt:
\glq{}Wir haben die feste Absicht, die Strecke stillzulegen\grq, und
sie hat das auf Anfrage gestern noch einmal bestätigt.\grqq
```

Hier mit csquotes:

[5] *Achtung:* Im Deutschen werden die Guillemets – im Gegensatz zum Französischen – nach innen zeigend verwendet.

```
Die Zeitung schrieb: \enquote{Die Bahn hat bereits im Frühjahr erklärt:
\enquote{Wir haben die feste Absicht, die Strecke stillzulegen}
und sie hat das auf Anfrage gestern noch einmal bestätigt.}
```

3.3.4 Binde- & Gedankenstrich

Auch wenn der Unterschied ohne den direkten Vergleich nicht jedem auffällt, sollte man vor allem in wichtigen Dokumenten darauf achten, den Unterschied zwischen Binde- und Gedankenstrich richtig umzusetzen. Im klassischen Englisch gibt es sogar noch einen Strichtyp mehr. Dort unterscheidet sich der „von–bis-Strich“ zusätzlich noch vom Gedankenstrich. Jedoch wird er nicht mehr durchgehend benutzt. So beschreibt der Typograf Robert Bringhurst den sogenannten Geviertstrich[6] als „zu lang für gute Typografie“ [2]. In der deutschen Typografie gibt es nur zwei unterschiedliche Strichtypen, jedoch werden sie je nach Anwendung von Leerzeichen umgeben oder nicht. LaTeX stellt um allen Ansprüchen gerecht zu werden drei verschiedene Strichtypen zur Verfügung:

`-`	-	Divis; Bindestrich
`--`	–	Halbgeviertstrich; „Von–bis-Strich“ und Gedankenstrich
`---`	—	Geviertstrich; klassischer englischer Gedankenstrich

Zusätzlich ist wichtig, dass der Halbgeviertstrich als „Von–bis-Strich“ nie (auch im englischen nicht) von Leerzeichen umgeben wird (z. B. 9–12 Uhr). Der Gedankenstrich wird im Deutschen durch Leerzeichen abgesetzt, wohingegen dieser im Englischen durch die unterschiedliche Strichlänge vom „Von–bis-Strich“ zu unterscheiden ist.

Neben den Text-Strichtypen gibt es (für alle Sprachen einheitlich) noch einen zusätzlichen Strichtyp: Das Minuszeichen. Entgegen einer weit verbreiteten Gewohnheit ist der Bindestrich „-“ kein Minuszeichen „−“. In LaTeX setzt man das Minuszeichen und den Bindestrich zwar über die gleiche Taste, jedoch im Mathemodus (vgl. Kapitel 14). Somit wird nicht nur das richtige Zeichen sondern auch die korrekten Abstände verwenden.

Das Minuszeichen ist häufig so lang wie ein Halbgeviertstrich, um zu Tabellenziffern zu passen. Dennoch sollten beide nicht austauschbar verwendet werden, um die unterschiedliche Bedeutung auch über das Markup zu wahren.

3.3.5 Akzente und Sonderzeichen

Die deutsche Sprache gehört zu denen, die spezielle Zeichen benötigen. Dank UTF-8 ist es mittlerweile möglich, alle dafür benötigten Zeichen und auch die meisten Sonderzeichen von Fremdsprachen direkt über die Tastatur einzugeben.

Bei Nutzung von XeLaTeX oder LuaLaTeX existiert hierbei keinerlei Einschränkung. Für pdfLaTeX ist es derzeit (März 2019) nicht möglich zusammengesetzte Umlaute (zuerst wird die Akzenttaste und anschließend der Buchstabe betätigt) zu setzen.

Dennoch gibt es einige Zeichen, über die nicht jede Tastatur verfügt. Die Tabellen 3.5 und 3.6 zeigen die möglichen Sonderzeichen und Akzente in der klassischen LaTeX-Notation.

3.3.6 Auslassungszeichen

Das Auslassungszeichen oder Ellipse (…) wird bei LaTeX mit dem Befehl

[6] Das Geviert ist eine typografische Längenangabe. Es entspricht meistens der breite des Buchstaben „M“ in der aktuellen Schriftgröße.

Tabelle 3.5: Akzente in der klassischen LaTeX-Notation am Beispiel des Buchstaben „o“. Die Befehle sind jedoch auf alle Buchstaben anwendbar.

ò	\‘{o}	õ	\~{o}	ǒ	\v{o}	o̧	\c{o}
ó	\’{o}	ō	\={o}	ő	\H{o}	ọ	\d{o}
ô	\^{o}	ȯ	\.{o}	o͡o	\t{oo}	o̲	\b{o}
ö	\"{o}	ŏ	\u{o}				

Tabelle 3.6: Sprachspezifische Buchstaben und Zeichen

œ	\oe	å	\aa	ł	\l	¿	?‘
Œ	\OE	Å	\AA	Ł	\L	¡	!‘
æ	\ae	ø	\o	ß	\ß		
Æ	\AE	Œ	\O				

`\ldots`

gesetzt. Beim Satz drei einfacher Punkte (...) stimmen die Abstände nicht. Bei Aufzählungen ist es zudem nötig einen kleinen Abstand zwischen die Auslassungspunkte und ein folgendes Komma zu setzen:

```
$a_i<b_i$ für $i=1$,~$2$, \ldots\,,~$n$.
```

$a_i < b_i$ für $i = 1, 2, \ldots, n$.

Hierfür bietet sich das Makro `\dots` and, welches eine Abkürzung für `\ldots\,` darstellt.

3.4 Titelei

Bei Dokumenten wird zwischen zwei verschiedenen Arten von Titeleien unterschieden: Es gibt entweder ganze Titelseiten oder lediglich einen Titelkopf.

Titelseiten zeigen den Dokumententitel zusammen mit weiteren Informationen, wie beispielsweise Autor auf einer eigenen Seite. Neben der Haupttitelseite gibt es, insbesondere bei Büchern, noch weitere Titelseiten mit Verlagsinformationen, Widmung oder ähnlichen Informationen.

Für die Erzeugung der Titelei gibt es zwei verschiedene Möglichkeiten (Es sollte immer *nur eine* davon benutzt werden, auch wenn theoretisch Kombinationen möglich sind.):

3.4.1 Automatische Titeleierzeugung mit `\maketitle`

Man übergibt hier mit Hilfe von anderen Makros, wie zum Beispiel `\author{`*Autor*`}` die Informationen an LaTeX und lässt es dann mithilfe des Befehls `\maketitle` die Titelseite(n) automatisch erzeugen. Die Standardklassen setzen auf der Titelseite lediglich die Informationen Autor, Titel und Jahr. Die KOMA-Klassen hingegen bieten eine weitaus größere Menge an Makros zur Erzeugung von Titelseiten, wie sie bei jedem Buch, insbesondere in Fachbüchern gefunden werden.

```
… Präambel mit Dokumentenklasse und Paketen …
\begin{document}

\titlehead{Titelkopf (frei gestaltbar)}
\title{Titel}
\subtitle{Untertitel}
\subject{Typisierung}
\author{Autor 1 \and Autor 2}
\date{Datum}
\maketitle[Seitennummer der ersten Titelseite]

… Inhalt des Dokumentes …
\end{document}
```

Da diese Befehle Text als Argument enthalten, sollte babel zum Zeitpunkt der Datenübergabe bereits aktiv sein. Daher empfiehlt es sich diese Makros *nicht* innerhalb der Präambel zu setzen, auch wenn es grundsätzlich möglich ist.

Optisch entspricht diese Variante dem ersten Beispieldokument (Beispiel 1 auf Seite 11). Der Titel wird abhängig von Dokumentenklasse und Einstellungen entweder auf eigenen Seiten, oder wie im Beispiel als Titelkopf gesetzt. Das bedeutet, dass sich der Dokumenteninhalt ohne Seitenumbruch der Titelei anschließt. Das Umschalten zwischen diesen beiden Varianten geschieht am einfachsten mit der Dokumentenklassenoption `titlepage`. So setzt `titlepage=true` Titelseiten und das Gegenstück `titlepage=false` einen einfachen Titelkopf. Zusätzlich erlaubt die Option noch den Wert `firstiscover`. Dieser Wert aktiviert die Ausgabe von Titelseiten und setzt zusätzlich die erste von ihnen als Umschlagseite. Das bedeutet, dass dort andere Seitenränder verwendet werden als für den Rest des Dokumentes. Diese können über Längenvariablen angepasst werden (siehe Abschnitt 5.2).

KOMA-Script bietet zudem weitere Makros zum Erzeugen von Widmungen, Schmutztitel[7], Verlagsinformationen, …

```
\extratitle{Schmutztitel}
\publishers{Verlag}
\uppertitleback{Titelrückseitenkopf}
\lowertitleback{Titelrückseitenfuß}
\dedication{Widmung}
\thanks{Fußnote}
```

mit Symbolen gekennzeichnet

3.4.2 Frei gestaltbare Titelseite

```
\begin{titlepage}
. . .
\end{titlepage}
```

[7] Der Schmutztitel ist die erste rechte Seite im Buch, auf der lediglich der Haupttitel, teilweise sogar nur eine Kurzform steht.

Tabelle 3.7: Die klassische Hierarchie der Gliederungsebenen in LaTeX

`\part`	Ebene −1 bei scrbook und scrreprt, außerdem auf eigener (bei scrbook rechter) Seite Ebene 0 bei scrartcl
`\chapter`	Ebene 0; nur bei scrbook, srcreprt bei scrreprt auf neuer Seite, bei scrbook auf nächster ungeraden (rechten) Seite
`\section`	Ebene 1
`\subsection`	Ebene 2; Letzte nummerierte Ebene in scrbook und scrartcl
`\subsubsection`	Ebene 3; Letzte nummerierte Ebene in scrartcl
`\paragraph`	Ebene 4; Kein direkter Zeilenumbruch nach der Überschrift
`\subparagraph`	Ebene 5; Optisch nicht von `\paragraph` unterscheidbar

Diese Variante bietet deutlich mehr Gestaltungsfreiraum, ist aber auch aufwendiger, da man die Positionierung der Elemente selbst übernehmen muss. Die Umgebung bewirkt einen leeren Seitenstil (vgl. Kapitel 7), sodass keine Kopf- und Fußzeile auf der Titelseite erscheint.

Die Dokumentenklassenoption `titlepage` hat hier keinen Einfluss. Es werden immer komplette Titelseiten erstellt. Die Erstellung eines Titelkopfes ist mit dieser Umgebung nicht möglich.

Enthält die Umgebung Inhalt für mehrere Seiten, so ist lediglich die erste Seite ohne Kopf- und Fußzeile. Der Stil der übrigen Seiten muss gegebenenfalls angepasst werden.

3.5 Vorspann, Hauptteil & Nachspann

Sehr lange Dokumente, hauptsächlich Bücher, werden häufig in Vorspann, Hauptteil und Nachspann unterteilt. KOMA-Script bietet in der dafür vorgesehenen Klasse scrbook Schalterbefehle um den jeweiligen Buch-Teil einzuleiten:

`\frontmatter`	*Vorspann*:	Seitennummern mit kleinen römischen Ziffern (i,ii,...), Kapitelüberschriften nicht nummeriert – Feinere Untergliederungen in Abschnitte nichtsinnvoll. Dieser Abschnitt eignet sich für die Titelei, diverse Verzeichnisse und ein Vorwort.
`\mainmatter`	*Hauptteil*:	arabische Seitenzahlen beginnend bei 1.
`\backmatter`	*Nachspann*:	Untergliederung wie beim Vorspann, Seitennummerierung wird aus dem Hauptteil fortgesetzt. Möglicher Inhalt wäre ein Literaturverzeichnis, ein Stichwortverzeichnis und/oder ein Anhang.

3.6 Gliederungsebenen

Zur Einteilung des Dokumentes und den Satz der entsprechenden Überschriften benutzt LaTeX die selben Befehle. Diese Kopplung unterstützt nicht nur das logische Markup, sonder dient auch der automatischen Erstellung des Inhaltsverzeichnisses. Die Hierarchie ist hierbei abhängig von der Dokumentenklasse, vgl. Tabelle 3.7.

Die Syntax zum Einleiten eines neuen Gliederungspunktes ist für alle Ebenen identisch:

```
\Ebene[Kurzform]{Überschrift}
\Ebene*{Überschrift}
```

Tabelle 3.8: Mögliche Werte der Dokumentenklassenoption `headings` samt Erläuterung ihrer Wirkung.

Verarbeitung der optionalen Kurzform	
`optiontotoc`	Kurzform wird nur für den Inhaltsverzeichniseintrag verwendet.
`optionhead`	Kurzform wird nur für Kolumnentitel verwendet.
`optiontoheadandtoc`	Standard. Kurzform wird sowohl für Inhaltsverzeichniseintrag, als auch Kolumnentitel verwendet.
Größe der Überschriften mit Angabe der größten relativen Größenangabe	
`big`	`\Huge`
`normal`	`\huge`
`small`	`\LARGE`
Formatierung der Kapitelüberschriften	
`twolinechapter`	Zweizeilige Kapitelüberschriften mit Präfix, wie von `\chapterformat` bestimmt.
`twolineappendix`	Zweizeilige Kapitelüberschriften mit Präfix im Anhang
`onelinechapter`	Einzeilige Kapitelüberschriften ohne Präfix
`onelineappendix`	Einzeilige Kapitelüberschriften im Anhang ohne Präfix

Für den aktuellen Abschnitt entspricht diese Syntax:

```
\section{Gliederungsebenen}
```

Diese Makros erledigen nicht nur die automatische Nummerierung und Formatierung der Überschriften, sondern tragen zusätzlich den Text der Überschrift ins Inhaltsverzeichnis und in die Kolumnentitel[8] ein. Die gesternte Version unterdrückt diesen Vorgang und setzt lediglich eine nicht nummerierte Überschrift in der entsprechenden Formatierung.

KOMA-Script ergänzt die Möglichkeiten von Standard-LaTeXnoch. Es gibt einen zusätzlichen Mechanismus um einzustellen, wie die Option zu den Gliederungsbefehlen zu verarbeiten ist. Tabelle 3.8 zeigt unter anderem die Bedeutung der zuständigen Werte für die Dokumentenklassenoption `headings`. Außerdem finden sich in der Tabelle auch die Werte zur Formatierungsänderung der Überschriften. Für weitere Formatierungsänderungen sei auch auf die zu den Gliederungsebenen gehörenden „Komafont-Elemente“ in Abschnitt 4.1.5 verwiesen.

Neben der globalen Option kann man auch für Kolumnentitel und Inhaltsverzeichniseintrag unterschiedliche Überschriften angeben. Dies ist mithilfe der Struktur

```
\Ebene[head=Kolumnentitel,tocentry=Inhaltsverzeichniseintrag]{Überschrift}
```

möglich. KOMA-Script prüft dabei ob die optionale Angabe ein Gleichheitszeichen oder ein Komma enthält und ändert seine Vorgehensweise entsprechend, falls dies der Fall ist. Benötigt man dennoch ein Komma oder Gleichheitszeichen als Bestandteil des Verzeichniseintrages, so kann das durch Gruppierung gesetzt werden. Es ist mit dieser Struktur auch möglich den Inhaltsverzeichniseintrag zu unterdrücken, indem man einen leeren Eintrag angibt (`head={}`). Echte Leere Einträge kann man mithilfe leerer Boxen (Kapitel 9) erzeugen.

Normalerweise wird bei der Nummerierung ab Kapitel/Abschnitt immer die Nummer der

[8] Kolumnentitel sind die Überschriften einzelner Buchseiten, zum Beispiel die Angabe der Kapitel-/Abschnittsüberschrift in der Kopfzeile.

nächsthöheren Ebene mit angezeigt. So erhält zum Beispiel die zweite `\section` im dritten `\chapter` die Nummer 3.2. Diese Zählung kann jedoch mit Hilfe der zugehörigen Zähler (siehe `secnumdepth` und `tocdepth` in Abschnitt 5.1) manipuliert werden. Bestehen die Objektnummern[9] nur aus arabischen Ziffern, so wird üblicherweise mit einem Punkt abgeschlossen. Sobald jedoch in einer Nummer ein anderes Zeichen auftaucht, entfällt der Punkt bei allen Objekten. Dieses Verhalten entspricht der Dokumentenklassenoption `numbers=auto`. Soll ein Punkt erzwungen oder verhindert werden, so ist das mit `numbers=endperiod` beziehungsweise `numbers=noendperiod` möglich.

3.6.1 Erweiterung der Gliederungsbefehle durch KOMA-Script

Neben den klassischen Gliederungsbefehlen existieren noch Makros, mit denen man nicht nummerierte Kapitel/Abschnitte setzen kann, die trotzdem einen Inhaltsverzeichniseintrag erzeugen und die Kolumnentitel beeinflussen.

```
\addpart[Kurzform]{Überschrift}
\addpart*{Überschrift}
\addchap[Kurzform]{Überschrift}
\addchap*{Überschrift}
\addsec[Kurzform]{Überschrift}
\addsec*{Überschrift}
```

Die gesternte Variante entspricht dabei den gesternten Standardanweisungen, jedoch wird bei ihnen im Gegensatz zu `\section*` & Co der Kolumnentitel geleert, wohingegen die Standardanweisungen die Seitenüberschriften unverändert lassen.

Zusätzlich zu den klassischen Gliederungsebenen bietet KOMA-Script den weiteren Überschriftentypus der `\minisec`.

```
\minisec{Überschrift}
```

Diese Art der Überschrift hat im Gegensatz zum Paragraph einen Zeilenumbruch nach dem Titel und kleinere Abstände. Dieses Makro setzt lediglich eine Überschrift. Es entspricht keiner Gliederungsebene und erhält somit weder eine Nummer, noch einen Eintrag in das Inhaltsverzeichnis oder in die Kolumnentitel.

3.6.2 Zusammenfassung

In den Klassen scrreprt und scrartcl existiert eine Umgebung, die für Zusammenfassungen bestimmt ist. Bei Büchern ist dies sehr unüblich, weswegen es diesen Mechanismus dort nicht gibt.

```
\begin{abstract}... Text ... \end{abstract}
```

Die Zusammenfassung folgt normalerweise direkt auf die Titelei und soll einen kurzen Überblick über das Dokument geben. Die Zusammenfassung erscheint bei Standardeinstellung beidseitig eingerückt und ohne Überschrift. Die Überschrift ist aufgrund der Positionierung und des Layouts im Allgemeinen nicht notwendig. Es ist dennoch möglich, mithilfe der Dokumentenklassenoption `abstract=true` eine entsprechende Bezeichnung zu erzeugen.

In der Dokumentenklasse scrartcl folgt die Zusammenfassung direkt dem Titelkopf. Bei scrreprt erscheint sie auf einer eigenen Seite vertikal zentriert.

[9] Dazu zählen neben den Überschriftennummern auch die Abbildungs-, Tabellen-, und Gleichungsnummern.

3.6.3 Anhang

Im Gegensatz zur Grobaufteilung (Abschnitt 3.5) ist ein Anhang bei jedem der Standard-Dokumententypen zu finden. Er wird analog zur Aufteilung mit einem Schalterbefehl eingeleitet:

`\appendix`

Dieser Befehl erzeugt keinerlei Ausgabe. Er stellt lediglich die Nummerierung der zweithöchsten Gliederungsebene (`\chapter` oder `\section`) auf Großbuchstaben um. Kleinere Ebenen haben entsprechend die Form „A.1“.

Da die Nummern der Abschnitte nun nicht mehr nur arabische Ziffern enthalten, führt ein Anhang auch dazu, dass der abschließende Punkt bei Objektnummerierungen unterdrückt wird (vgl. Dokumentenklassenoption `numbers` auf Seite 15).

Entgegen einer weit verbreiteten Ansicht, dass die erste Seite das Anhangs explizit im Inhaltsverzeichnis erscheinen muss, ist dies bei LaTeX nicht der Fall. Der Anhang wird vom Leser durch die Großbuchstaben in der Nummerierung als solcher erkannt. In der Regel ist diese Kennzeichnung vollkommen ausreichend. In Spezialfällen, sollte für diese Entscheidung jedoch die Konzeption der Anhanges berücksichtigt werden.

3.7 Das Inhaltsverzeichnis

LaTeX kann durch seine Konzeption automatisch ein Inhaltsverzeichnis anlegen, in welches die Überschriften mit der zugehörigen Seitennummer eingetragen werden. Hierfür verwendet das Programm eine Hilfsdatei mit der Endung `.toc`. Der Übersichtlichkeit halber werden jedoch nur die obersten Gliederungsebenen in das Inhaltsverzeichnis eingetragen (vgl. Abschnitt 5.1).

`\tableofcontents`

Dieser Befehl erzeugt an der entsprechenden Stelle im Dokument das Inhaltsverzeichnis. Wenn das aktuelle Dokumente mindestens ein Verzeichnis enthält, so muss das Dokument mindestens zweimal kompiliert werden. Im ersten Durchlauf wird die Hilfsdatei (in diesem Fall `.toc`) erstellt. Beim zweiten Durchlauf wird die Datei geladen und ihr Inhalt entsprechend formatiert.

Einfache Formatierungsänderungen geschehen mithilfe der Klassenoption `toc`. Tabelle 3.9 zeigt die möglichen Werte mitsamt einer kurzen Erläuterung. Für weitere Informationen zur Formatierung von Verzeichnissen allgemein wird auf Kapitel 13 verwiesen.

3.8 Dokumente aufteilen

Bei längeren Dokumenten empfiehlt es sich aus mehreren Gründen, das Dokument in einzelne `.tex`-Dateien aufzuteilen. Dies macht die Dokumente übersichtlicher und spart insbesondere auch Zeit beim Kompilieren, wenn jeweils nur die aktuell bearbeitete Datei neu kompiliert wird.

`\input{`*Dateiname*`}`

Das `\input`-Makro fügt dabei den Inhalt der `.tex`-Datei ohne Modifikationen an der Position des Befehls ein. Die Benutzung ist auch innerhalb der Präambel möglich. Es eignet sich somit auch bestens dafür sämtliche persönliche Anpassungen in einer selbst erstellten Präambel-Datei auszulagern und diese zu Beginn jedes Dokumentes des gleichen Typs einzubinden. Eine Dateiendung muss hierbei lediglich dann angegeben werden, falls sie sich von `.tex` unterscheidet.

Tabelle 3.9: Werte für die KOMA-Dokumentklassenoption `toc`

`bib`	Literaturverzeichnis erscheint im Inhaltsverzeichnis.
`bibnumbered`	Literaturverzeichnis erscheint im Inhaltsverzeichnis und erhält zusätzlich eine Nummer.
`flat`	Das Inhaltsverzeichnis ist tabellarisch. In der ersten Spalte stehen die Gliederungsnummern, in der Zweiten die Überschriften und in der Letzten die Seitenzahlen.
`graduated`	Das Inhaltsverzeichnis ist hierarchisch aufgebaut mit begrenztem Platz für die Gliederungsnummern.
`idx`	Das Stichwortverzeichnis erscheint im Inhaltsverzeichnis ohne Nummerierung.
`listof`	Abbildungs- und Tabellenverzeichnis erscheinen im Inhaltsverzeichnis ohne Nummerierung.
`listofnumbered`	Wie `listof`, jedoch mit Nummerierung.
`nobib`	Kein Literaturverzeichnis im Inhaltsverzeichnis.
`noidx`	Kein Stichwortverzeichnis im Inhaltsverzeichnis.
`nolistof`	Kein Abbildungs- und kein Tabellenverzeichnis im Inhaltsverzeichnis.
`nonumberline`	Standardeinstellung. `numberline` ist deaktiviert.
`numberline`	Nicht nummerierte Einträge bündig mit dem Text nummerierter Einträge gleicher Ebene.

Zudem ist es möglich `\input` zu schachteln, dies bedeutet, dass Dateien, die mit `\input` geladen werden, ebenso das Makro `\input` enthalten dürfen.

Innerhalb des Textkörpers empfiehlt es sich für jedes Kapitel eine eigene Datei anzulegen. Hierfür ist das `\include`-Makro zu bevorzugen.

`\include{`*Dateiname*`}`

Der Befehl bindet den Code aus der Datei so ein, dass der Dateiinhalt auf einer neuen Seite beginnt und mit `\clearpage` abgeschlossen wird. Die Funktionsweise entspricht in vereinfachter Form der Abfolge

```
\clearpage\input{Dateiname}\clearpage
```

Zusätzlich verfügt `\include` noch über eine eigene Struktur, die es ermöglicht nach einer Dateienliste das Einbinden lediglich für bestimmte Dateien zu aktivieren:

`\includeonly{`*Dateienliste*`}` Präambel

Dieses in die Präambel gehörende Makro bewirkt, dass lediglich bestimmte Teildateien kompiliert werden, die mit `include` eingefügt wurden.

Der Vorteil dieser Vorgehensweise offenbart sich, wenn man zuvor das gesamte Dokument schon einmal kompiliert hat und sich dann erst der Bearbeitung einer Teildatei widmet, so bleiben Nummerierungen und Querverweise erhalten.

```
\includeonly{kapitel-1,kapitel-4}
... %enthält \begin{document}
\include{kapitel-1}
...
\include{kapitel-n}
```

Wenn die Datei schon einmal ohne `\includeonly` kompiliert wurde, so bleibt Kapitel 1 auch nach dem Einschub Kapitel 1 und Kapitel 4 erhält weiterhin die Nummer 4, auch wenn 2 und 3 im aktuellen Ausgabedokument gar nicht auftauchen. Selbiges gilt für die Seitenzahlen. Dies bleibt solange erhalten, solange man an der Struktur der Datei keine Änderungen vornimmt.

4 Textformatierungen

4.1 Schriftarten und Textauszeichnung

Schriftarten werden nach Sippe, Familie, Form und Serie klassifiziert. Die Standard-Schriftsippe in LaTeX ist Computer Modern. Wie bereits in Abschnitt 3.3.1 erwähnt, sollte sie bei pdfLaTeX durch die erweiterte Version „Latin Modern“ ersetzt werden. Falls gewünscht, ist auch das Einbinden anderer Schriftarten möglich. Dies funktioniert in den meisten Fällen vollkommen analog zu Latin Modern über das Einbinden der Pakete. Beim Ändern der Schriftsippe ist jedoch immer zu beachten, dass sie alle benötigten Zeichen auch tatsächlich enthält. So bietet zum Beispiel nicht jede Schriftsippe mathematische Symbole oder alle Familien. Eine sehr gute Übersicht über die möglichen Schriftarten und deren Verwendung bietet „The LaTeX Font Catalogue“ `http://www.tug.dk/FontCatalogue/`.

pdfLaTeX kann nicht auf die Standard-Systemschriften zurückgreifen, allerdings ist es ohne viel Aufwand möglich den Compiler auf LuaLaTeX (oder XeLaTeX) zu wechseln. Dies ist z. B. dann nötig, wenn man eine exakte Schriftvorgabe hat und dieser Schrifttyp nicht für pdfLaTeX existiert.

Hier werden lediglich die grundlegenden Makros sowie der Basismechanismus zur Änderung der Schriftart betrachtet. Die Befehle funktionieren für andere Schriftarten analog solange der Zeichensatz dies unterstützt.

4.1.1 Unterschiede zwischen den Compilern

Der Mechanismus um Schriftarten zu laden unterscheidet sich stark zwischen pdfLaTeX und LuaLaTeX. XeLaTeX funktioniert ähnlich wie LuaLaTeX, und wird daher hier nicht weiter betrachtet.

Wie bereits in

4.1.2 Schriftattribute

Bei TeX und somit auch LaTeX werden die einzelnen Schriftschnitte typografisch klassifiziert. Die größe Gruppe stellt dabei eine Schriftsippe dar. Das ist eine Gruppe von Schriftfamilien, die in unterschiedlichen Familien vorliegt.

Die Einordnung der Standard-LaTeX-Befehle orientiert sich an der TeX-Standardschrift, der Computer Modern. Bei dieser existieren eine Antiqua (Serifenschrift), eine Grotesk (Serifenlose) sowie eine nichtproportionale Schrift (Monofont, Schreibmaschinenschrift):

`\rmfamily`	Serifenschrift
`\sffamily`	Serifenlose
`\ttfamily`	

Es gibt Schriftsippen, die über deutlich mehr unterschiedliche Schriftschnitte verfügen. Da die Anzahl an im Fließtext verwendeten Schrifttypen so gering wie möglich gehalten werden soll, werden üblicherweise nicht alle als Makro zur Verfügung gestellt.

Neben den eben gezeigten Makros, bei denen es sich um Schalterbefehle handelt, existieren noch Textmakros, die die Wirkung auf ihr Argument beschränken:

Tabelle 4.1: Übersicht über die verfügbaren Kombinationen aus Familie, Form und Serie für die Latin Modern Schriftsippe

Familie	*Form*	*Serie*	*Schalter*	*Befehl*	
Roman			`\rmfamily`	`\textrm{`*Text*`}`	Beispieltext
	aufrecht		`\upshape`	`\textup{`*Text*`}`	Beispieltext
		medium	`\mdseries`	`\textmd{`*Text*`}`	Beispieltext
		fett	`\bfseries`	`\textbf{`*Text*`}`	**Beispieltext**
	kursiv		`\itshape`	`\textit{`*Text*`}`	*Beispieltext*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*Beispieltext*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***Beispieltext***
	geneigt		`\slshape`	`\textsl{`*Text*`}`	*Beispieltext*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*Beispieltext*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***Beispieltext***
	Kapitälchen		`\scshape`	`\textsc{`*Text*`}`	BEISPIELTEXT
Sans Serif			`\sffamily`	`\textsf{`*Text*`}`	Beispieltext
	aufrecht		`\upshape`	`\textup{`*Text*`}`	Beispieltext
		medium	`\mdseries`	`\textmd{`*Text*`}`	Beispieltext
		fett	`\bfseries`	`\textbf{`*Text*`}`	**Beispieltext**
	geneigt		`\slshape`	`\textsl{`*Text*`}`	*Beispieltext*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*Beispieltext*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***Beispieltext***
`Typewriter`			`\ttfamily`	`\texttt{`*Text*`}`	`Beispieltext`
	aufrecht		`\upshape`	`\textup{`*Text*`}`	`Beispieltext`
		medium	`\mdseries`	`\textmd{`*Text*`}`	`Beispieltext`
		fett	`\bfseries`	`\textbf{`*Text*`}`	**`Beispieltext`**
	kursiv		`\itshape`	`\textit{`*Text*`}`	*`Beispieltext`*
	geneigt		`\slshape`	`\textsl{`*Text*`}`	*`Beispieltext`*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*`Beispieltext`*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***`Beispieltext`***
	Kapitälchen		`\scshape`	`\textsc{`*Text*`}`	`BEISPIELTEXT`

`\textrm{`*Text*`}`	Latin Modern Roman (mit Serifen)
`\textsf{`*Text*`}`	ohne Serifen)
`\texttt{`*Text*`}`	`Latin Modern Typewriter (Monofont)`

Jede dieser Familien unterteilt sich in verschiedene Formen und Serien. Tabelle 4.1 zeigt die üblicherweise vefügbaren Kombinationen.

Da die aufrechte Form `\upshape` und die Serie Medium `\mdseries` Standard sind machen diese Befehle nur Sinn, wenn man die Form und Serie zuvor geändert hat.
Möchte man wieder zum Standardfont des Dokumentes wechseln, benötigt man lediglich einen der Befehle

```
\normalfont
\textnormal{Text}
```

Dies erlaubt es auch einen bestimmten Schrifttyp zu wählen, unabhängig davon, welcher

Schrifttyp zuvor aktiviert war.

Neben `\emph` und den manuellen Möglichkeiten der Schriftattributumschaltung existieren bei KOMA-Script noch *zwei* Makros für hoch- bzw. tiefgestellten Text[1] Diese passen automatisch die Schriftgröße entsprechend an.

`\textsuperscript{`*Text*`}`	Hochstellen
`\textsubscript{`*Text*`}`	Tiefstellen

Kursivkorrektur

Wenn ein aufrechtes Zeichen einem kursiven folgt, so kann – je nach verwendeter Schriftart – der Abstand zwischen beiden zu klein werden, sodass beide Zeichen überlappen. Dies kann man mit der sogenannten Kursivkorrektur beheben:

`\/`

Das Makro `\textit` erledigt dies in so gut wie allen Fällen automatisch. Bei Benutzung des Schalterbefehles `\itshape` muss die Korrektur jedoch manuell gesetzt werden. [Beispiel nach 23, S. 97]

```
[Das {\itshape Schiff}]\\\relax
[Das \textit{Schiff}]\\\relax
[Das {\itshape Schiff\/}]
```

[Das *Schiff*]
[Das *Schiff*]
[Das *Schiff*]

Ligaturen

Ligaturen sind die Zusammenfassung mehrerer Buchstaben zu einem einzigen Zeichen. Sie stammen aus dem klassischen Bleisatz und wurden damals aus Stabilitätsgründen als einzelnes Zeichen gedruckt [23, S.97]. Dieses Verhalten wurde im Digitaldruck beibehalten. Somit werden heutzutage hauptsächlich die Buchstabenfolgen `fl`,`ff`,`fi`,`ffi` und `ffl` in Ligaturen gewandelt. Je nach Schriftart gibt es auch mehr oder teilweise auch gar keine Ligaturen.

4.1.3 Schriftgrößen

Die Schriftgrößen skalieren immer mit der im Befehl `\documentclass` festgelegten Standardgröße für die Schrift. So wird mit `\normalsize` die Standardschriftgröße abgefragt. Alle zur Auswahl stehenden Schriftgrößen mitsamt Beispiel findet man in Tabelle 4.2.

4.1.4 Textauszeichnung

Es gibt verschiedene Möglichkeiten Text hervorzuheben. Um das logische MarkUp des Dokumentes zu wahren, sollte der Befehl:

`\emph{`*Text*`}`

benutzt werden. Er setzt den Text kursiv, erlaubt jedoch im Gegensatz zu `\textit` Schachtelungen:

```
\emph{Erste Hervorhebungsebene \emph{zweite Ebene} Ende der ersten}
```

Erste Hervorhebungsebene zweite Ebene *Ende der ersten*

[1] Bei den Standardklassen für LaTeX 2$_\varepsilon$ existiert nur die Variante zum hochstellen von Elementen.

Tabelle 4.2: Skalierung der Schriften bei Hauptschrift 11 pt und Angaben der tatsächlichen Schriftgrößen bei unterschiedlichen Basisgrößen.

Schalter	*Beispiel*	*Tatsächliche Größen:*	10 pt	11 pt	12 pt
`\tiny`	tiny		5 pt	6 pt	6 pt
`\scriptsize`	scriptsize		7 pt	8 pt	8 pt
`\footnotesize`	footnotesize		8 pt	9 pt	10 pt
`\small`	small		9 pt	10 pt	11 pt
`\normalsize`	normalsize		10 pt	11 pt	12 pt
`\large`	large		12 pt	12 pt	14 pt
`\Large`	Large		14 pt	14 pt	17 pt
`\LARGE`	LARGE		17 pt	17 pt	20 pt
`\huge`	huge		20 pt	20 pt	25 pt
`\Huge`	Huge		25 pt	25 pt	25 pt

Die Kursive ist im Allgemeinen sehr gut für Hervorhebungen geeignet. Im Gegensatz zu anderen Schriftformen kann Sie im Fließtext benutzt werden, ohne den Lesefluss zu stark zu beeinträchtigen.

Selbstverständlich kann man Hervorhebungen auch durch Änderung der Schriftattribute setzen, allerdings sollte man dabei immer darauf achten, den Textfluss nicht durch zu häufige oder zu auffällige Änderungen zu unterbrechen. Als weitere Möglichkeit der Auszeichnung ist auch eine Unterstreichung mit

`\underline{`*Text*`}`

möglich. Die Unterstreichung gilt jedoch als veraltete Auszeichnungsart aus dem Schreibmaschinenzeitalter. Sie sollte somit aus typografischen Gründen nicht mehr verwendet werden!

4.1.5 Globale Formatierungsänderungen für Elemente

Wenn man dauerhaft die Textformatierung für verschiedene spezielle Textelemente, wie zum Beispiel die Kopf- und Fußzeile (Kapitel 7), verändern will, kann man dies mit einem der folgenden Befehle erreichen:

`\setkomafont{`*Element*`}{`*Schriftformatierungsbefehle*`}`
`\addtokomafont{`*Element*`}{`*Schriftformatierungsbefehle*`}`

`\setkomafont` definiert die Formatierung völlig neu, während `\addtokomafont` die existierende Definition erweitert. Wichtig ist es immer die Reihenfolge der Anwendung der verschiedenen Elemente zu beachten. So wird zum Beispiel das Element `disposition` auf jeden Gliederungsbefehl *vor* der spezifischen Einstellung angewandt. Außerdem ist zu Beachten, dass die *Schriftformatierungsbefehle* tatsächlich nur Schalterbefehle zur Schriftumschaltung enthalten sollen (Farbänderungen sind auch zulässig). Jedoch sollte es unbedingt vermieden werden diese Elemente für die Übergabe von Textausgaben zu benutzen oder Umdefinitionen darin vorzunehmen.

Mithilfe des Befehls

`\usekomafont{`*Element*`}`

kann man direkt auf die Schriftart von einzelnen Elementen zugreifen bzw. umschalten.

Die wichtigsten Elemente werden im Folgenden in alphabetischer Reihenfolge erläutert (Eine Übersicht für alle Elemente findet man in [9]):

`author`	Autor auf der Titelseite bei Verwendung von `\maketitle` (Abschnitt 3.4.1)
`caption`	Gleitumgebungsbeschreibung (Abschnitt 12.2)
`captionlabel`	Gleitumgebungslabel (Abschnitt 12.2)
`chapter`	Kapitelüberschriften (Abschnitt 3.6)
`chapterentry`	Inhaltsverzeichniseintrag von Kapiteln (Abschnitt 3.7)
`chapterentrypagenumber`	Zugehörige Seitenzahl (Abschnitt 3.7)
`chapterprefix`	Präfix von Kapiteln/Anhängen (bei Option `headings=twolinechapter/-appendix`) (Abschnitt 3.6 & Tabelle 3.8)
`date`	Datumsangabe auf der Titelseite bei Verwendung von `\maketitle` (Abschnitt 3.4.1)
`descriptionlabel`	Label einer `description`-Umgebung (Kapitel 8)
`disposition`	Alle Gliederungsüberschriften, sowie die Überschrift der Zusammenfassung. Anwendung vor spezifischen Elementen (Abschnitt 3.6)
`footnote`	Marke und Text von Fußnoten (Abschnitt 5.4.2)
`footnotelabel`	Marke einer Fußnote, Anwendung nach dem `footnote`-Element (Abschnitt 5.4.2)
`footnotereference`	Referenzierung der Fußnotenmarke (Abbildung 5.4.2)
`footnoterule`	Linie über den Fußnoten (Abschnitt 5.4.2)
`labelinglabel`	Label der `labeling`-Umgebung (Abschnitt 8.2)
`labelingseparator`	Trennzeichen einer `labeling`-Umgebung (Abschnitt 8.2)
`minisec`	Überschrift einer `\minisec` (Abschnitt 3.6.1)
`pagefoot`	Seitenfuß bei Verwendung von scrlayer-scrpage (Abschnitt 7.2)
`pagenumber`	Seitenzahl, die mit `\pagemark` gesetzt wird (Kapitel 7)
`part`	Überschrift der Ebene `\part` (Abschnitt 3.6)
`section`	Abschnittsüberschrift (Abschnitt 3.6)
`subject`	Typisierung des Dokumentes (Abschnitt 3.4)
`subsection`	Überschrift von `\subsection` (Abschnitt 3.6)
`subsubsection`	Überschrift von `\subsubsection` (Abschnitt 3.6)
`subtitle`	Untertitel auf der Titelseite bei Verwendung von `\maketitle` (Abschnitt 3.4.1)
`title`	Titel auf der Titelseite bei Verwendung von `\maketitle` (Abschnitt 3.4.1)
`titlehead`	Kopf über dem Haupttitel auf der Titelseite bei Verwendung von `\maketitle` (Abschnitt 3.4.1)

Das folgende Beispiel bewirkt, dass `captionlabel` genauso formatiert wird, wie `descriptionlabel`:

```
\setkomafont{captionlabel}{\normalfont\usekomafont{descriptionlabel}}
```

`\normalfont` wird benötigt, da vor dem Element `captionlabel` noch das Element `caption` angewandt wird. Um eine genaue Übereinstimmung zu erhalten, muss die Wirkung von `\usekomafont{caption}` wieder zurückgenommen werden.

Neben der Benutzung des gesamten Stils eines Elements kann man auch lediglich einzelne Stilelemente, wie die Schriftfamilie, -form oder -serie auswählen. Dies ist mithilfe der folgenden Makros möglich:

`\usefontofkomafont{`*Element*`}`	Schriftgröße
`\useencodingofkomafont{`*Element*`}`	Grundlinienabstand
`\usesizeofkomafont{`*Element*`}`	Kodierung
`\usefamilyofkomafont{`*Element*`}`	Familie
`\useseriesofkomafont{`*Element*`}`	Form und Serie
`\useshapeofkomafont{`*Element*`}`	Kodierung

4.2 Umbrüche

4.2.1 Absatzumbruch

LaTeX orientiert sich bei der Formatierung des Textes nicht an Zeilen, sondern an Absätzen. Es wird somit zunächst der gesamte Inhalt eines Absatzes analysiert und dann die beste Möglichkeit des Zeilenumbruchs automatisch gewählt. Zeilenumbrüche im Code werden somit einfach ignoriert. Der Umbruch von Absätzen erfolgt hingegen einfach durch eine Leerzeile, oder mit dem Befehl

```
\par
```

Die Art und Weise, wie Absatzümbrüche gekennzeichnet werden sollen, wird über die Dokumentenklassenoption `parskip=`*Methode* festgelegt. Die Unterschiedlichen Methoden für den Umbruch finden sich in Tabelle 4.4. Prinzipiell wird zwischen zwei Varianten der Absatzkennzeichnug unterschieden:

- Einrückung der ersten Zeile eines neuen Absatzes. Ein neuer Sinnabschnitt wird nicht eingerückt und durch Abstand gekennzeichnet. (Standardeinstellung, `parskip=false`) Bei dieser Einstellung kann jedoch, wenn es zum Auffüllen der Seite notwendig ist ein Abstand zwischen den Absätzen auftreten (vgl. `\flushbottom`, Abschnitt 6.1). Möchte man dies in jedem Fall verhindern, so benutzt man `parskip=never`.

- Absatzkennzeichnung durch Abstand. Hier wird nach jedem Absatz ein Abstand von einer halben oder ganzen Zeile eingefügt (`parskip=full` oder `parskip=half`). Am Ende eines Sinnabschnittes wird entsprechend mehr Platz gelassen[2].

Es existieren keine festen Regeln, welche Variante man zu wählen hat. Der Vorteil der Einrückung liegt darin, dass der Anfang des Absatzes gekennzeichnet wird und nicht das Ende. Dies ermöglicht Beispielsweise eine Unterscheidung wenn ein Seiten- mit einem Absatzumbruch zusammenfällt. Auch bei Verwendung abgesetzter Mathematischer Formeln (Kapitel 14) hat diese Variante Vorteile. So kann man trotz Abstand immer erkennen, ob an der Formel ein Absatzumbruch stattfindet oder nicht. Die Absatzkennzeichnung durch Abstand eignet sich dahingegen besonders für den Satz mehrerer (schmaler) Spalten, da die Zeilen mitsamt einer Einrückung in diesem Fall zu kurz geraten können.

[2] Hier finden die Makros `\smallskip`/`\medskip`/`\bigskip` Anwendung, die in Abschnitt 5.2.3 erklärt wurden.

Tabelle 4.4: Werte für die KOMA-Dokumentklassenoption `parskip`

`false`	Einzug von 1 em in der ersten Zeile, wobei der erste Absatz eines Abschnitts nicht eingezogen wird
`full`	Vertikaler Abstand von einer Zeile, Absatzenden haben Leerraum von mindestens 1 em
`full-`	Vertikaler Abstand von einer Zeile
`full+`	Vertikaler Abstand von einer Zeile, Absatzenden haben Leerraum von mind. 1/4 einer normalen Zeile
`full*`	Vertikaler Abstand von einer Zeile, Absatzenden haben Leerraum von mind. 1/3 einer normalen Zeile
`half`	Vertikaler Abstand von 1/2 Zeile, Absatzenden haben Leerraum von mindestens 1 em
`half-`	Vertikaler Abstand von 1/2 Zeile
`half+`	Vertikaler Abstand von 1/2 Zeile, Absatzenden haben Leerraum von mind. 1/4 einer normalen Zeile
`half*`	Vertikaler Abstand von 1/2 Zeile, Absatzenden haben Leerraum von mind. 1/3 einer normalen Zeile

Möchte man den Einzug der ersten Zeile lediglich lokal verhindern, oder einen Einzug an einer Stelle, wo normalerweise keiner stattfinden würde setzen, so benutzt man die Makros

```
\noindent
\indent
```

4.2.2 Zeilenumbruch

`\\`	Zeilenumbruch (einfach)
`\\[`*Abstand*`]`	Zeilenumbruch mit optionalen Abstand zur nächsten Zeile
`\\*[`*Abstand*`]`	Wie `\\`, jedoch kann kein Seitenumbruch vor der nächsten Zeile stattfinden.
`\newline`	Zeilenumbruch, wie `\\`
`\linebreak[`*Priorität*`]`	Zeilenumbruch, Zahl entspricht dabei der Priorität (0=niedrig bis 4=zwingend).

Vorsicht: Im LR-Modus (vgl. Abschnitt 9.1) von LaTeX (z. B. `\mbox`) ist kein Zeilenumbruch erlaubt, ein Befehl für einen Zeilenumbruch wird ignoriert und erzeugt eine Warnung.

Um den Unterschied zwischen den verschiedenen Makros zu verdeutlichen hier ein kleines Beispiel:

Dies ist eine sehr lange Zeile Text bis zu einem Umbruch, um den Unterschied zwischen `\linebreak` und `\\` zu demonstrieren. (ohne manuellen Umbruch)

Dies ist eine sehr lange Zeile Text bis zu einem Umbruch, um den Unterschied zwischen `\linebreak` und `\\` zu demonstrieren. (mit `\linebreak`)

Dies ist eine sehr lange Zeile Text bis zu einem Umbruch, um den
Unterschied zwischen `\linebreak` und `\\` zu demonstrieren. (mit `\\`)

`\linebreak` erhält somit den Blocksatz. Die Priorität gibt dabei an inwieweit die Vorgaben für den Wortabstand berücksichtigt werden müssen. Ohne Angabe der Priorität entspricht das Makro dem Wert 4 und erzwingt somit einen Umbruch, was zu sehr unschönen riesigen

Wortabständen führen kann.

4.2.3 Zeilenumbruch verhindern

`~`	sog. „geschütztes Leerzeichen“; kein Zeilenumbruch zwischen Wörtern; Beispiel `Stufe~5`
`\nolinebreak[`*Priorität*`]`	kein Zeilenumbruch

4.2.4 Zeilenabstand ändern

Mit Änderungen am Zeilenabstand sollte man prinzipiell vorsichtig sein. Die Standardeinstellungen von LaTeX sind in der Regel so gut, dass laienhaftes Eingreifen sie eigentlich nur verschlechtern kann. Jedoch gibt es leider oft Vorgaben für Haus- oder Abschlussarbeiten, die unschöne Einstellungen, wie beispielsweise doppelten Zeilenabstand verlangen[3].

Falls Sie mit einer Vorgabe für den Zeilenabstand konfrontiert werden, so sollten Sie beachten, dass bei Verwendung von doppelten Zeilenabstand, lediglich der Fließtext mit eben diesem gesetzt wird. Für die Titelseite oder Verzeichnisse wechselt man auf einfachen Zeilenabstand.

Möchte man solche Vorgaben erfüllen und eineinhalbfachen oder doppelten Zeilenabstand einstellen, so bindet man am besten das Paket setspace ein

```
\usepackage{setspace}
```

und verwendet die Befehle

```
\onehalfspacing
\doublespacing
```

Möchte man zum einfachen Zeilenabstand zurückkehren, so verwendet man den Befehl:

```
\singlespacing
```

Alternativ kann man auch die Paketoptionen *onehalfspacing* und *doublespacing* benutzen. Um weiterhin einen sauberen Textsatz zu erreichen, lohnt es sich den Satzspiegel anschließend neu zu berechnen (Kapitel 6).

4.2.5 Seitenumbruch

`\newpage`	Seitenumbruch; Gleitobjekte (vgl. Kapitel 12) können jedoch noch positioniert werden
`\pagebreak[`*Priorität*`]`	Seitenumbruch mit Angabe der Priorität (0=niedrig bis 4=zwingend)
`\clearpage`	beendet die aktuelle Seite sofort; Übrige Gleitobjekte werden direkt im Anschluss ausgegeben
`\cleardoublepage`	wie `\clearpage`, allerdings beginnt der Text mit der nächsten ungeraden (rechten) Seite

Der Unterschied zwischen einem Seitenumbruch mit `\newpage` und `\pagebreak` ist quasi derselbe, wie zwischen `\newline` und `\linebreak`: Bei `\pagebreak` wird der Blocksatz eingehalten. Jedoch wird hier nicht der Wortabstand entsprechend vergrößert, sondern der Text bis zum Zeilenende ausgeschrieben, bevor ein Seitenumbruch erfolgt. `\pagebreak` innerhalb eines Absatzes erzwingt somit lediglich einen Seitenumbruch am Zeilenende, wohingegen `\newpage` sofort die

[3] Unschön deshalb, weil der Streifeneffekt bei zu großem Zeilenabstand zunimmt und somit der gleichmäßige Graueindruck der Seite gestört wird. Dies verschlechtert die Lesbarkeit.

Seite beendet und im Anschluss an eventuell ausgegebene Gleitobjekte, siehe Kapitel 12, auf einer neuen Seite fortsetzt.

`\clearpage` beendet ebenfalls die Seite sofort wie `\newpage` und setzt im Anschluss auf einer oder mehreren Seiten sämtliche noch zu setzende Gleitobjekte, siehe ebenfalls Kapitel 12.

Zweispaltiger Textsatz
Wenn die Dokumentenklassenoption `twocolumn` gesetzt ist, oder das Makro `\twocolumn` aufgerufen wurde, dann beenden `\pagebreak` und `\newpage` lediglich die aktuelle Spalte und beginnen eine neue. Möchte man die komplette Seite beenden, was möglicherweise eine leere Rechte Spalte zur Folge hat, benötigt man eines der Makros `\clearpage` oder `\cleardoublepage`.

4.2.6 Seitenumbruch verhindern

`\nopagebreak[`*Priorität*`]`
`\begin{samepage}...\end{samepage}`

kein Seitenumbruch, mit Angabe der Priorität (0=niedrig bis 4=zwingend), Seitenumbruch nur zwischen Absätzen, außer er wird erzwungen `\nopagebreak` verhindert zwischen zwei Absätzen, dass dort einen Seitenumbruch eingefügt wird. Innerhalb eines Absatzes verhindert es einen Seitenumbruch nach der aktuellen Zeile. die `samepage`-Umgebung erlaubt für Ihren Inhalt nur Seitenumbrüche bei Absatzumbrüchen.

4.2.7 Unsaubere Seitenumbrüche – Seitengröße in Sonderfällen anpassen

In Sonderfällen ist es manchmal erforderlich den Textbereich einer einzelnen Seite zu vergrößern, da beispielsweise die nächste Seite lediglich eine Textzeile zeigt. Für diesen Fall gibt es das Makro

`\enlargethispage{`*Länge*`}`
`\enlargethispage*{`*Länge*`}`

Der Stern sorgt dafür, dass die relevanten elastischen Maße (z. B. der Absatzabstand) auf den Minimalwert gesetzt werden, um die Abweichung gegenüber den anderen Seiten möglichst gering zu halten. Beispielsweise sieht eine Vergrößerung des Textbereiches um eine einzelne Zeile so aus:

```
\enlargethispage*{\baselineskip}
```

4.3 Trennung und Bindestrich

4.3.1 Worttrennung

LaTeX trennt Wörter, wenn der Zeilenumbruch rechtsbündig zwischen Wörtern nicht gelingt. Allerdings kennt LaTeX den Sinn der Wörter nicht und trennt deswegen manche zusammengesetzten Wörter nicht korrekt oder sinnvoll, z. B.: Urinstinkt.

`\-`	Trennmöglichkeit, die andere Trennungen ausschließt (`Ur\-instinkt`, `Stau\-becken`)
`"-`	Trennmöglichkeit, die andere Trennungen nicht ausschließt (wenn LaTeX eine Trennstelle einfach nicht kennt)
`""`	Trennmöglichkeit, bei der kein Trennstrich benötigt wird

```
\begin{enumerate}[nosep,itemindent=*]
    \item[1] Jetzt kommt hier eine
    ganz normale Trennstelle
    \item[2] Jetzt kommt hier eine
    ganz nor\-male Trennstelle
    \item[3] Jetzt kommt hier eine
    Bindestrich-Trennstelle
    \item[4] Jetzt kommt hier eine
    echte Binde"=strich-Trennstelle
    \item[5] Jetzt kommt hier eine
    echte Binde"~strich-Trennstelle
    \item[6] Jetzt kommt hier eine
    echte Bin\-de"~strich-Trennstelle
    \item[7] Jetzt kommt hier eine
    echte Linie/Kurve-Trennstelle
    \item[8] Jetzt kommt hier eine
    echte Linie/""Kurve-Trennstelle
    \item[9] Jetzt kommt hier eine
    echte Linie\slash{}Kurve-Trennstelle
\end{enumerate}
```

1 Jetzt kommt hier eine ganz normale Trennstelle
2 Jetzt kommt hier eine ganz normale Trennstelle
3 Jetzt kommt hier eine Bindestrich-Trennstelle
4 Jetzt kommt hier eine echte Binde-strich-Trennstelle
5 Jetzt kommt hier eine echte Binde-strich-Trennstelle
6 Jetzt kommt hier eine echte Bin-de-strich-Trennstelle
7 Jetzt kommt hier eine echte Linie/Kurve-Trennstelle
8 Jetzt kommt hier eine echte Linie/Kurve-Trennstelle
9 Jetzt kommt hier eine echte Linie/Kurve-Trennstelle

Beispiel 2: Manipulation der Worttrennung (nach [23, Bsp. 03-07-07])

`"|`	Ligaturauflösung und Schaffung einer Trennmöglichkeit (`Auf"|forderung` setzt Aufforderung statt Auffforderung)

Beispiele für die verschiedenen Trennmöglichkeiten und Bindestriche finden sich in Beispiel 2.

Kommt ein Wort öfters vor, sollte es in die Trennnstellenliste aufgenommen werden. Diese Liste lässt sich über das folgende Makro erweitern:

`\hyphenation{`*Wort-lis-te*`}`

Zum Beispiel wird das Wort „Staubecken“ mithilfe der Angabe der Trennstellen (`\hyphenation{Stau-be-cken}`) lediglich an den vorgegebenen Stellen getrennt. Sollen Trennstellen für mehrere Wörter angegeben werden, so werden sie als Liste innerhalb des `\hyphenation`-Makros durch ein Leerzeichen getrennt angegeben, z. B.

```
\hyphenation{Ma-nu-skript Com-pu-ter Stau-be-cken}
```

4.3.2 Geschützte Leerzeichen

Wie bereits in Abschnitt 4.2.6 erwähnt kann man den Zeilenumbruch zwischen zwei Wörtern verhindern, indem man sie durch ein geschütztes Leerzeichen trennt. Dies ist insbesondere dann nötig, wenn eine Trennung die Lesbarkeit verschlechtern würde.

`~`	geschütztes Leerzeichen mit normalen Abstand. Beispiel: `Dr.~Mustermann`
`\,`	geschütztes Leerzeichen mit kleineren Abstand. Beispiel: `z.\,B.` oder `50\,kg`

4.3.3 Bindestrich

`-`	Bindestrich
`"=`	der andere Trennungen unterdrückt
`"~`	Bindestrich

LaTeX verbietet im Allgemeinen eine Trennung von Wörtern mit Bindestrich mit Ausnahme der Bindestrichposition. Für sehr lange Wörter kann eine zusätzliche Trennstelle dennoch notwendig sein. Dann muss Sie mithilfe der eben gezeigten Zeichenkombinationen aktiviert werden.

4.4 Textausrichtung

Standardmäßig setzt LaTeX jeden Text im Blocksatz, also links- und rechtsseitig bündig. Es gibt daher keinen Befehl für Blocksatz, man erhält ihn indem man eine andere Form der Ausrichtung beendet. Um Text im Flattersatz zu setzen gibt es die in Tabelle 4.5 dargestellten Befehle und Umgebungen.

Zusätzlich zu dieser Ausrichtung ist auch eine Absatzeinrückung möglich. Standardmäßig wird diese mit Hilfe der beiden Umgebungen

```
\begin{quote}. . . \end{quote}
\begin{quotation}. . . \end{quotation}
```

gesetzt. Bei beiden wird der Text abgesetzt und beidseitig eingerückt. Sie unterscheiden sich lediglich dadurch, dass bei `quote` der Beginn eines neuen Absatzes durch Abstand und bei `quotation` durch Einrückung gekennzeichnet wird.

KOMA-Script fügt zudem einen Befehl hinzu der es erlaubt einzelne Absätze beliebig weit einzurücken:

```
\begin{addmargin}[linker Einzug]{rechter Einzug}. . . \end{addmargin}
\begin{addmargin*}[innerer Einzug]{äußerer Einzug}. . . \end{addmargin*}
```

Die Stern-Variante ist für zweiseitigen Textsatz gedacht und addiert den Einzug anstatt von links und rechts, auf der Innen- bzw. Außenseite. Wird lediglich das notwendige Argument für den Einzug angegeben, so wird dieser für beide Seiten verwendet.

```
\begin{addmargin}[2cm]{3cm}
```

setzt somit auf der linken Seite einen Einzug von 2 cm und rechts 3 cm.

```
\begin{addmargin}{1cm}
```

setzt auf beiden Seiten einen Einzug von 1 cm.

Verbesserte Ausrichtung mit ragged2e – Worttrennung bei Flattersatz
Bei Benutzung der normalen Makros für den Flattersatz, wird die Worttrennung deaktiviert. Dies kann insbesondere bei schmalen Textspalten, beispielsweise innerhalb von Tabellen, zu sehr unschönen Ergebnissen führen. Das Paket ragged2e bietet einen Ersatz für die herkömmlichen Umgebungen und Makros für den Flattersatz, welche die Worttrennung nicht deaktivieren, siehe Tabelle 4.5. Zusätzlich lassen Sie sich in ihrem verhalten durch eine Vielzahl von Parametern beeinflussen. Diese können der Paketdokumentation [14] entnommen werden. Zusätzlich gibt es die Möglichkeit die normalen Makros durch die des Paketes ersetzen zu lassen. In diesem Fall muss das Paket mit der Option `newcommands` geladen werden.

Tabelle 4.5: Gegenüberstellung von Umgebungen und Schalterbefehlen für die Ausrichtung von Text aus Standard-LaTeX und ragged2e

Ausrichtung	Standard-LaTeX	ragged2e	Standard-LaTeX	ragged2e
	Umgebungen		*Befehle*	
zentriert	`center`	`Center`	`\centering`	`\Centering`
linksbündig	`flushleft`	`FlushLeft`	`\raggedright`	`\RaggedRight`
rechtsbündig	`flushright`	`FlushRight`	`\raggedleft`	`\RaggedLeft`

4.5 Farben - Das color-Paket

Mit dem Paket color und einem der folgenden Befehle kann die Farbe des Textes verändert werden.

```
\color{Farbe}
\textcolor{Farbe}{Text}
```

Die vordefinierten Farben sind white[4], black, red, green, blue, cyan, magenta, yellow[4]. Weitere Farben können in der Form

```
\color[Farbmodell]{Farbdefinition}
```

gewählt werden, wobei das Farbmodell eines aus `rgb` (red,green,blue), `cmyk` (cyan, magenta, yellow, black), `gray` oder `named` ist. Die Farbdefinition ist eine Komma-getrennte Liste mit Werten von 0 bis 1, wobei der Wert den Anteil der Farbkomponenten des Modells widerspiegelt. Der Code `[rgb]{1,0,0}` definiert somit Rot und `[cmyk]{0,1,0,0}` definiert Magenta.

Eigene Farbdefintionen sind ebenfalls möglich:

```
\definecolor{Farbname}{Farbmodell}{Farbdefinition}
```

```
\definecolor{eisvogelblau}{cmyk}{0.9,0,0.3,0}
Dies ist \textcolor{eisvogelblau}{Eisvogelblau}
```

Dies ist Eisvogelblau

Es sind auch mit Farbe gefüllte Boxen möglich:

```
\colorbox[Farbmodell]{Farbe}{Text}
\fcolorbox[Farbmodell]{Rahmenfarbe}{Hintergrundfarbe}{Text}
```

Um letzten Endes wieder zur Ausgangsfarbe zurückzukehren, existiert das Makro

```
\normalcolor
```

Es wechselt zurück zu der Farbe, die am Ende der Präambel gewählt war.

4.6 Code „wörtlich“ ausdrucken

Oft, zum Beispiel zum schreiben eines Skriptes über LaTeX ist es nötig Code wörtlich im Dokument abdrucken zu können. Hierfür existieren die Beiden Umgebungen

[4] Der Text wurde zur besseren Darstellung in eine `colorbox` eingebunden.

```
\begin{verbatim} Text \end{verbatim}
\begin{verbatim*} Text \end{verbatim*}
```

Die Sternchenform unterscheidet sich von der ungesternten dadurch, dass in ihr Leerzeichen als ␣ sichtbar gemacht werden:

```
\begin{verbatim*}
\documentclass[ngerman]{scrartcl}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{babel}
\begin{document}
Ein bisschen Text\dots
\end{document}
\end{verbatim*}
```

```
\documentclass[ngerman]{scrartcl}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{babel}
\begin{document}
Ein␣bisschen␣Text\dots
\end{document}
```

Zusätzlich gibt es noch zwei Makros für kleinere Codeschnipsel:

```
\verb|Code|
\verb*|Code|
```

Das Zeichen | dient hierbei als Gruppierungszeichen. Es sind sämtliche Sonderzeichen zulässig, die noch keine besondere Aufgabe haben (z. B. auch +).

5 Strukturautomatisierung

5.1 Zähler

LaTeX nummeriert die Elemente eines Dokuments, wie z. B. die Seiten, Abbildungen oder Überschriften. Hinter jeder solchen Nummerierung steckt ein entsprechender Zähler, der zu Beginn eines neuen Elements des Typs um eins erhöht wird. Die Zähler sind in der Regel bezeichnend benannt, sodass eine Zuordnung relativ intuitiv möglich ist. Die Namen der wichtigsten Zähler mit ihrer Funktion sind in Tabelle 5.1 gelistet.

Der Wert eines Zählers kann mit folgenden Befehlen manipuliert werden:

`\setcounter{`*Zähler*`}{`*neuer Wert*`}`	Weist dem Zähler den neuen Wert zu
`\addtocounter{`*Zähler*`}{`*ganze Zahl*`}`	Addiert die Zahl (auch negative Zahl möglich) zum Wert des Counters
`\stepcounter{`*Zähler*`}`	Inkrement von 1

Die Wertmanipulationen von Zählern sind alle global. Sie gelten auch außerhalb der aktuellen Umgebung weiter. So wird verhindert, dass ein Wert doppelt belegt wird. Neben den Werten direkt ist auch die Ausgabe der Werte ein wichtiger Aspekt für die Dokumentenerstellung.

`\value{`*Counter*`}`	Wert; Kann überall dort als Argument angegeben werden, wo LaTeX eine Wertangabe erwartet. Erzeugt keine Ausgabe.
`\arabic{`*Counter*`}`	arabische Ziffern
`\Roman{`*Counter*`}`	große römische Ziffern
`\roman{`*Counter*`}`	kleine römische Ziffern
`\Alph{`*Counter*`}`	Großbuchstaben A–Z (Werte von 1–26)
`\alph{`*Counter*`}`	Kleinbuchstaben a–z (Werte von 1–26)
`\fnsymbol{`*Counter*`}`	Symbole (Werte von 1–9); gedacht für die Fußnoten

Die interne Abfrage, z. B. zur Ausgabe der Seitenzahl oder der Kapitelnummerierung geschieht

Tabelle 5.1: Auslistung wichtiger Zähler und ihrer Aufgaben

Zählername	*Funktion*
part, chapter, …, subparagraph	Zähler für part, chapter, …
enumi, enumii, enumiii, enumiv	Aufzählungszähler für die Ebenen 1 bis 4
page	Seitennummer
footnote	Fußnotenzähler
equation	Formel-Zähler
figure	Bilder-Zähler
table	Tabellen-Zähler
secnumdepth	Nummerierungstiefe bei Überschriften
tocdepth	Nummerierungstiefe im Inhaltsverzeichnis

jedoch mit

`\the`*Countername*

Für die Seitenzahl bedeutet dies

```
\thepage
```

52

Über diese Befehle kann man auch die Art der Nummerierung ändern. Möchte man zum Beispiel bei einer Aufzählung Buchstaben statt Zahlen verwenden, so kann man dies durch Umdefinition des Befehles (vgl. Abschnitt 5.3) `\thepage` erreichen:

```
\renewcommand{\thepage}{\Roman{page}}
\thepage
```

LII

Bei den Seitennummerierungen gibt es noch eine weitere Möglichkeit:

`\pagenumbering{`*Nummernstil*`}`

`\pagenumbering` ist ein globales Makro. Die Änderung gilt ab dem Makroaufruf bis zur nächsten Änderung des gleichen Typs. Die Nummerierungsstile sind wieder `arabic`, `roman`, `Roman`, `alph`, `Alph` oder `fnsymbol`, jedoch werden hier nicht die Makros sondern lediglich der Name übergeben:

```
\pagenumbering{alph}
\thepage
```

a

`\pagenumbering` setzt zudem die Seitenzahl zurück und beginnt ab der aktuellen Seite wieder mit 1. Die Seitenzahl kann jederzeit mit dem `\setcounter`-Befehl neu gesetzt werden und mit `\thepage` abgefragt werden.

Eigene Zähler definieren

Ein beliebiger neuer Zähler wird mit

`\newcounter{`*Zählername*`}[`*Reset-Counter*`]`

definiert. Wichtig ist, dass der Name noch nicht für einen anderen Zähler belegt sein darf. Er kann auch als eine Parkvariable bei Umdeklaration von anderen Countern dienen oder um Werte zwischenzuspeichern. Zusätzlich kann man optional einen weiteren Counter als Reset-Referenz angeben. Wird dieser Referenzcounter verändert, so wird der neu definierte Counter zurückgesetzt.

Das Beispiel definiert einen neuen Counter namens `mycounter`, der zu Beginn jedes neuen Abschnittes zurückgesetzt wird.

Besondere Makros für die Manipulation von `secnumdepth` und `tocdepth`

```
\partnumdepth
\chapternumdepth
\sectionnumdepth
\subsectionnumdepth
\subsubsectionnumdepth
```

```
\paragraphnumdepth
\subparagraphnumdepth
```

Die einer Ebene zugewiesene Nummer unterscheidet sich je nachdem, ob die Dokumentenklasse über die Ebene `\chapter` verfügt oder nicht. Damit man sich diese Unterscheidung nicht merken muss, deklariert KOMA-Script spezielle Makros, die je nach Ebene der zugewiesenen Zahl entsprechen. Somit ist es beispielsweise möglich die Nummerierung bis zur Ebene `\subsubsection` mit

```
\setcounter{secnumdepth}{\subsubsectionnumdepth}
```

einzuschalten.

5.2 Längen und Zwischenräume

5.2.1 Längen

LaTeX speichert für die Erzeugung der Seiten Längenmaße, wie z. B.. die Breite des Textes, in entsprechenden Variablen. Die Variablen sind wieder bezeichnend benannt. Ein wichtiger Unterschied zu den Zählern ist, dass Längen die Struktur eines Makros haben (z. B. `\linewidth`), also mit Backslash benutzt werden. Einige Beispiele für wichtige Längen sind in Tabelle 5.2 zu finden.

Tabelle 5.2: Die wichtigsten Längenparameter mit Angabe der Funktion.

Länge	*Funktion*
`\baselineskip`	Abstand zwischen zwei Zeilen innerhalb eines Absatzes
`\evensidemargin`	linker Rand für gerade Seiten
`\footskip`	Abstand Unterkante Rumpf bis Unterkante Fußzeile
`\headsep`	Abstand Unterkante Kopf bis Oberkante Rumpf
`\oddsidemargin`	linker Rand allgemein oder für ungerade Seiten
`\paperheight`	Seitenhöhe
`\paperwidth`	Seitenbreite
`\parindent`	Einrückabstand der ersten Zeile eines Absatzes
`\parskip`	Absatzabstand
`\tabcolsep`	halbe Breite des Spaltenzwischenraums für `tabulars`
`\textheight`	Texthöhe
`\textwidth`	Textbreite
`\topmargin`	Abstand oberer Rand bis Oberkante Kopfzeile
`\topskip`	Abstand Oberkante Rumpf bis Grundlinie der ersten Zeile

Die Manipulation einer Länge folgt einer ähnlichen Struktur, wie die der Zähler, ist jedoch nicht global.

```
\setlength{Länge}{neues Maß}
\setlength{Länge}{Maß plusMaß1 minusMaß2}
\addtolength{Länge}{Maß}
```

Länge auf neues Maß setzen, elastische Definition einer Länge, Addition des Maßes zur Länge Allgemein sollte man mit der direkten Änderung von Längen sehr vorsichtig sein,

Tabelle 5.3: Typografische Einheiten, die von LaTeX verstanden werden. Die Einheiten sind der Größe nach beginnend von der kleinsten sortiert.

Kürzel	*Name*	*Definition*	*Wert* [pt]	*Wert* [µm]
pc	pica		12	4218
bp	big point	1/72 in	1,003 75	$352\frac{7}{9}$
pt	point	1/72,27 in	1	351,46
sp	scaled point		$1{,}5 \cdot 10^{-5}$	0,005 36
em	Alte typografische Einheit, die mit der Schrift skaliert. Ihr Wert entspricht ungefähr der Breite des Großbuchstaben „M“ in der aktuellen Schriftart.			
ex	Analog em; Entspricht ungefähr der Höhe von „x“ in der aktuellen Schriftart.			

da Längenvariablen oft auch für weitere Funktionen als ihr Hauptfunktion genutzt werden. Es empfiehlt sich, Änderungen an Maßen die den Satzspiegel beeinflussen, weitestgehend zu unterlassen, beziehungsweise nur lokal zu setzen.

Bei manchen Längen empfiehlt es sich, LaTeX ein gewisses Spektrum zu geben, aus dem es die Länge wählen kann. So sind die meisten Abstände im Satzspiegel dehnbar definiert, um unter anderem die oberste und unterste Zeile bündig zum Rand zu setzen. Beispiele für Längenänderungen:

```
\setlength{\textwidth}{15cm}
\setlength{\textwidth}{\paperwidth}
\setlength{\parskip}{
    1ex plus .5ex minus .2ex}
```

Textbreite auf 15 cm
Textbreite = Papierbreite
Absatzabstand = 1 ex dehnbar, höchstens 1.5ex und mindestens .8ex

Alle Längenangaben erfordern immer auch eine Einheit. Die Einheit kann entweder direkt angegeben werden oder man benutzt eine bereits existierende Länge als Referenz. Somit ist es auch möglich einen Bruchteil einer Länge zu benutzen (z. B. `.5\linewidth`). Für die direkte Angabe der Einheit stehen metrische Einheiten (cm/mm) , Zoll (in) und einige Typografische Einheiten zur Verfügung. Diese sind in Tabelle 5.3 gezeigt.

Bei manchen Befehlen wird vom Autor eine eigene Längenangabe gefordert. Manchmal ist es aber auch wünschenswert die Längenangabe gerade so zu wählen, wie ein einzugebender Text breit oder hoch ist. Hierfür gibt es zwei Befehle, die eine beliebige Textbreite oder Texthöhe in einer Länge ablegen:

```
\settowidth{Länge}{Text}
\settoheight{Länge}{Text}
```

Somit kann man zum Beispiel Text exakt so weit einrücken, wie eine Beschriftung:

```
\newlength{\mylength}
\settowidth{\mylength}{Bezeichnung: }
Bezeichnung: Text\\
\hspace*{\mylength}Fortsetzung des Textes
```

Eigene Längen

Eigene Längen werden ähnlich wie Zähler definiert:

```
\newlength{neue Länge}
```

Die üblichste Nutzung ist hierfür das Zwischenspeichern von Maßen. Somit kann man zum Beispiel temporär den Absatzeinzug auf exakt 1 cm setzen.

```
Ein kleines bisschen Text mit nachfolgendem normalen Ansatzumbruch.\par
Ein kleines bisschen Text mit nachfolgendem Absatzabstand von 1cm.
\newlength{\parindentsaved}
\setlength{\parindentsaved}{\parindent}
\setlength{\parindent}{1cm}\par
Ein kleines bisschen Text mit nachfolgendem normalen Absatzumbruch.
\setlength{\parindent}{\parindentsaved}\par
Ein kleines bisschen Text.
```

Ein kleines bisschen Text mit nachfolgendem normalen Ansatzumbruch.
Ein kleines bisschen Text mit nachfolgendem Absatzabstand von 1cm.
Ein kleines bisschen Text mit nachfolgendem normalen Absatzumbruch.
Ein kleines bisschen Text.

Auslesen von Maßen

Manchmal ist es praktisch die Maße von Längen explizit auslesen zu können. Dies ist durch das TEX-Makro

`\the`

möglich. Man gibt einfach das Längenmakro im Anschluss an `\the` an: Die Ausgabe auf diese Weise erfolgt immer in der Einheit pt. Dieser Wert kann dann entsprechend in die benötigte Einheit umgerechnet werden.

5.2.2 Zwischenräume

`\hspace{`*Länge*`}`
`\hspace*{`*Länge*`}`

Das `\hspace`-Makro setzt an die aktuelle Position eine horizontale Lücke der angegebenen Länge. Der optionale Stern erzeugt den Zwischenraum auch wenn Zeilenumbrüche involviert sind. Ohne das Sternchen wird ein solcher Abstand zu Beginn einer Zeile nicht gesetzt. Steht zusätzlich vor oder nach dem Befehl ein Leerzeichen, so wird dieses zum Abstand hinzugefügt:

```
Das ist\hspace{1cm}1\,cm.\\
Das ist \hspace{1cm}1\,cm.\\
Das ist \hspace{1cm} 1\,cm.\\
\hspace{1cm}Funktioniert nicht.\\
\hspace*{1cm}unter Zwang.
```

Das ist 1 cm.
Das ist 1 cm.
Das ist 1 cm.
Funktioniert nicht.
unter Zwang.

Ein weiterer sehr nützlicher Befehl, bei dem LATEX das Maß selber elastisch vorgibt ist

`\hfill`

Dabei wird soviel Zwischenraum eingefügt, dass die laufende Zeile links- und rechtsbündig abschließt. Ein mehrfaches Anwenden führt zusätzlich zu gleichen Abständen:

```
Herr Müller \hfill Brief \hfill Regensburg, den \today
```

Herr Müller Brief Regensburg, den 20. April 2019

Es gibt noch zwei Varianten, die dne Zwischenraum mit Punkten oder einer Linie auffüllen.

`\dotfill`
`\hrulefill`

Um das gesamte Dokument über ein einheitliches Layout zu erhalten, gibt es auch ein paar vorgefertigte Abstände mit fester Größe.

`\quad`	Zwischenraum der Breite 1 em
`\qquad`	Zweimal `\quad` als Zwischenraum
`\,`	3/18 em
`\:`	4/18 em
`\;`	5/18 em
`\!`	−3/18 em
`\␣`	ein explizites Leerzeichen

Diese Makros können auch innerhalb von Mathe-Umgebungen benutzt werden (siehe Kapitel 14).

5.2.3 Vertikale Abstände

Analog zu den horizontalen Zwischenräumen, existieren die Makros

`\vspace{`*Maß*`}`
`\vspace*{`*Maß*`}`
`\vfill`

Sauber verwendet werden diese Makros bei Absatzumbrüchen, da LaTeX an anderen Positionen keine Notwendigkeit für vertikale Abstände sieht. Somit wird der Abstand verzögert, falls kein Umbruch vorhanden ist:

```
Wort1\vspace{5mm}Wort2

Wort3
\vspace{5mm}

Wort4
```

Wort1Wort2

Wort3

Wort4

Analog zu den horizontalen Abständen gibt es auch hier vorgefertigte Abstände. Da vertikale Abstände häufig dehnbar gesetzt werden, sind auch diese Abstände flexibel definiert:

`\bigskip`	`\bigskipamount = 12pt plus 4pt minus 4pt`
`\medskip`	`\medskipamount = 6pt plus 2pt minus 2pt`
`\smallskip`	`\smallskipamount = 3pt plus 1pt minus 1pt`

5.3 Eigene Befehle

LaTeX gestattet die Deklaration eigener Befehle bzw. die Umdefinition bereits vorhandener Makros. Dies ermöglicht Abkürzungen oder die einheitliche Formatierung besonderer Strukturen. Die Syntax für Deklarationen lautet:

`\newcommand{\`*Befehlsname*`}{`*Definition*`}`
`\newcommand*{\`*Befehlsname*`}{`*Definition*`}`
`\renewcommand{\`*Befehlsname*`}{`*Definition*`}`

```
\renewcommand*{\Befehlsname}{Definition}
```

Die Sternchenversion ist immer dann zu bevorzugen, wenn das Makro entweder keine Argumente erhält oder diese Argumente keine Absatzumbrüche enthalten sollen. Sie erlaubt keine Absatzumbrüche innerhalb der Argumente und erleichtert somit die Fehlersuche.

```
\newcommand{\Autor}{Marei Peischl}
\Autor
```

Marei Peischl

`\renewcommand` erlaubt es, bereits vorhandene Befehle zu überschreiben. Dies sollte daher nur verwendet werden, wenn man genau weiß, wofür das Makro, welches man überschreiben möchte verwendet wird. Bei intern verwendeten Makros kann dies sonst schnell zu unvorhersehbarem Verhalten führen.

Grundsätzlich sollten Neu- und Umdefinitionen *global* vorgenommen werden um das Dokument möglichst einheitlich zu gestaltet. Die Makros wirken jedoch lokal. Somit gelten Änderungen nur innerhalb der aktuellen Gruppe oder Umgebung und werden nach Beendigung wieder in den Ursprungszustand zurückgesetzt.

5.3.1 Eigene Makros mit Argumenten

Um auch in eigenen Makros Argumente verarbeiten zu können verfügen die Makros zur Neu- und Umdefinition noch über optionale Parameter:

```
\newcommand{\Befehlsname}[Anzahl][Standardwert]{Definition}
\newcommand*{\Befehlsname}[Anzahl][Standardwert]{Definition}
\renewcommand{\Befehlsname}[Anzahl][Standardwert]{Definition}
\renewcommand*{\Befehlsname}[Anzahl][Standardwert]{Definition}
```

Die *Anzahl* gibt die Gesamtzahl der Argumente an. Der *Standardwert* markiert das erste Argument als optional und hinterlegt einen Wert, falls dieses Argument nicht übergeben wird.

Zur Abfrage der Argumente dient das #-Zeichen. An den Platz, an dem die Argumente eingefügt werden sollen, wird der jeweilige Platzhalter #1 für das erste Argument, #2 für das Zweite, usw. eingefügt.

```
\relax
    \newcommand*{\blau}[1]{\textcolor{blue}{#1}}
    \blau{Blauer Text}\\
    \newcommand*{\bunt}[2][red]{\textcolor{#1}{#2}}
    \bunt[blue]{Blauer Text} \bunt{Roter Text}\\
    Vor der Änderung: \thepage
    \renewcommand*{\thepage}{\Roman{page}}
    %Seitenzahlen ab diesem Punkt in großen römischen Ziffern
    Nach der Änderung: \thepage
```

Blauer Text
Blauer Text Roter Text
Vor der Änderung: 57 Nach der Änderung: LVII

5.3.2 Eigene Umgebungen

Analog zu den Makros ist es auch möglich eigene Umgebungen zu deklarieren.

```
\newenvironment[Anzahl][Standard]{Start-Code}{Ende-Code}
\newenvironment*[Anzahl][Standard]{Start-Code}{Ende-Code}
\renewenvironment[Anzahl][Standard]{Start-Code}{Ende-Code}
\renewenvironment*[Anzahl][Standard]{Start-Code}{Ende-Code}
```

Die Umgebung kann dann, wie die vordefinierten Umgebungen benutzt werden, wobei die Argumente immer nur beim Umgebungsanfang angegeben werden.

```
\begin{Umgebungsname}[optionales Argument]{Argument}
...
\end{Umgebungsname}
```

5.4 Querverweise

5.4.1 Einfache Querverweise

An jeder Stelle im Text kann mit

```
\label{Markername}
```

ein Marker gesetzt werden, auf den man sich mit

```
\ref{Markername}
\pageref{Markername}
```

beziehen kann. Der Bezug zielt je nach Ort des `\label`-Befehls auf eine Abschnittsüberschrift, ein Bild oder auch eine mathematische Gleichung. `\ref` gibt dabei die entsprechende Elementnummer zurück, wobei `\pageref` die zum Marker gehörige Seitenzahl ausgibt.

```
\label{sec:references}Wir befinden uns gerade in Abschnitt
\ref{sec:references} auf Seite \pageref{sec:references}.
```

Wir befinden uns gerade in Abschnitt 5.4.1 auf Seite 58.

Das `\label` bezieht sich dabei immer auf das letzte referenzierbare Objekt. In diesem Fall ist das letzte Objekt vor der Markierung die Abschnittsüberschrift.

5.4.2 Fußnoten und Randnotizen

Das ist eine Randnotiz.

Eine Randnotiz bzw. Fußnote[1], kann mit

```
\marginpar[Notiz falls links]{Notiz falls rechts}
\footnote{Fußnotentext}
```

erstellt werden.

Fußnoten
Mit der Fußnoten werden mittels des zugehörigen Zählers (`footnote`) automatisch nummeriert. Der Befehl zur Erzeugung von Fußnoten darf nur im normalen Textmodus Verwendung finden, ansonsten wird die Fußnote im Ausgabefile nicht erzeugt. Somit ist es normalerweise nicht möglich Fußnoten in Tabellen, oder anderen Boxen zu setzen. Ist eine Fußnote erforderlich, so muss sie manuell eingefügt werden:

[1] Dies ist eine Fußnote

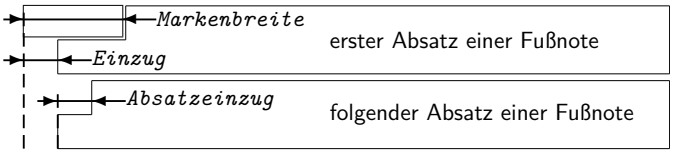


Abbildung 5.1: Parameter für die Formatierung von Fußnoten. Abbildung aus [8, S. 106].

```
\footnotemark[Nummer]
\footnotetext[Nummer]{Fußnote}
```

Die Nummer wird dabei im laufenden Text mit dem ersten Befehl gesetzt. Der zweite Befehl muss im normalen Textmodus aufgerufen werden und erzeugt die Fußnote. Hiermit ist es möglich Fußnoten innerhalb von LR-Boxen, Tabellen und im Mathemodus zu erzeugen.

KOMA-Script erlaubt außerdem die Verwendung eines Trennzeichens für Fußnoten (bei Standardeinstellungen ist das Trennzeichen ein Komma). Umschalten geschieht über die Dokumentenklassenoption `footnotes` mit den möglichen Werten `multiple` (aktiviert das Trennzeichen) und `nomultiple` (deaktiviert das Trennzeichen).

Die Formatierung der Fußnoten kann mit den Makros

```
\deffootnote[Markenbreite]{Einzug}{Absatzeinzug}{Markendefinition}
\deffootnotemark{Markendefinition}
\thefootnotemark
```

verändert werden. Die Bedeutung der Parameter ist in Abschnitt 5.4.2 gezeigt. Die Standardeinstellung für die Fußnoten lautet wie folgt:

```
\deffootnote[1em]{1.5em}{1em}{\textsuperscript{\thefootnotemark}}
```

Das Makro `\deffootnotemark` definiert die Fußnotenmarken im Text. Voreingestellt ist:

```
\deffootnotemark{\textsuperscript{\thefootnotemark}}
```

`\thefootnotemark` setzt dabei jeweils die Fußnotenmarkierung im eingestellten Schriftstil (siehe Abschnitt 4.1.5).

Neben der Formatierung der Fußnoten an sich, ist es auch möglich die Trennlinie zwischen Text- und Fußnotenbereich abzuändern.

```
\setfootnoterule[Höhe]{Länge}
```

Bleibt einer der Parameter leer, so wird die Entsprechende Eigenschaft nicht geändert. Beide Werte werden immer bei der relativen Schriftgröße `\normalsize` ausgewertet werden. Sie sind somit von der aktuellen relativen Schriftgröße unabhängig.

Referenzen auf Fußnoten setzen

Mit KOMA-Script ist es möglich, Fußnoten mit einem `\label` zu versehen und anschließend mit

```
\footref{Markername}
```

zu referenzieren. Dies macht es erheblich einfacher Fußnoten mehrfach zu benutzen.

Randnotizen

Randnotizen haben zwei argumente, um den Text zu übergeben:

```
\marginpar[linker Text]{rechter Text}
```

Somit ist es möglich Randnotizen zu erzeugen, die unterschiedlich sind, je nachdem ob Sie auf

dem rechten oder linken Rand gedruckt werden. Dies ist insbesondere im zweiseitigen Satz wichtig, da hier die Fußnoten immer außen gedruckt werden, also auf linken Seiten links und auf rechten Seiten rechts. Zum Beispiel wurden die beiden Pfeile als Randnotizen auf dieser
⟶ Seite und Seite 62 mit ein- und demselben Makro erzeugt:

```
\marginpar[\hfill$\longrightarrow$]{$\longleftarrow$}
```

KOMA-Script verfügt zusätzlich noch über eine Variante, die Randnotizen auf linken Seiten rechts- und auf rechten Seiten linksbündig setzt.

`\marginline{`*Randnotiz*`}`

Natürlich könnte das auch mit `\marginpar` erzeugt werden, jedoch mit erhöhtem Aufwand. Letztlich hat `\marginline` keine andere Bedeutung als

```
\marginpar[\raggedleft#1]{\raggedright#1}
```

5.4.3 Hyperlinks - Das hyperref-Paket

Das hyperref-Paket erweitert die Möglichkeiten im Umgang mit Textmarken und -referenzen um ein Vielfaches. Am meisten verbreitet ist die Einbindung des Paketes, um Hyperlinks zu erzeugen, das bedeutet, wenn man auf einen Link im PDF-Dokument klickt, springt das Dokument sofort an die entsprechende Stelle. Da das Paket hierzu viele LaTeX-interne Makros ändert, sollte immer darauf geachtet werden, dass dieses Paket so spät wie möglich geladen wird.

`\usepackage[`*Optionen*`]{hyperref}`

Da es auch Dokumentenklassen und Pakete gibt, welche hyperref voraussetzen, gibt es auch eine Möglichkeit die Optionen nach dem Laden des Pakets anzupassen.

`\hypersetup{`*Optionsliste*`}`

Die Einstelllungen müssen jedoch für eine saubere Umsetzung innerhalb der Präambel gesetzt werden.

Farbanpassugen

Bei Standardeinstellung werden dokumenteninterne Verweise mit roten Rahmen gekennzeichnet. Es ist jedoch auch möglich die Links über die Schriftfarbe hervorzuheben. Die Kennzeichnung kann über Paketoptionen eingestellt werden

```
\usepackage[colorlinks=true,linkcolor=blue]{hyperref}
```

`colorlinks=true` färbt dabei alle Hyperlinks ein, anstatt Rahmen zu erzeugen und `linkcolor=`*Farbe* wählt zusätzlich noch die Farbe für dokumenteninterne Verweise (nicht Literaturverweise). Es Die übrigen Arten von Referenzen (Literaturangaben, URLs, …) bleiben jedoch in der Ihnen zugewiesenen Standardfarbe. Tabelle 5.4 zeigt die Möglichkeiten zur Farbänderung auf. Die Änderung der Farbe, welcher Art auch immer, gehört zum Text, dies bedeutet, wenn man blaue Links wählt und später die entsprechende Seite druckt, so werden die Links auch blau ausgedruckt.

Neben der Farbänderung auf farbige links, ist es selbstverständlich auch möglich die Farbe der farbigen Rahmen zu ändern. Der Vorteil der Rahmen liegt darin, dass Sie nicht gedruckt werden. Es gibt also automatisch eine Unterscheidung zwischen digitaler und Printversion. Tabelle 5.5 zeigt die Möglichkeiten. Wichtig im Gegensatz zu farbigen Links ist, dass die Farbe als rgb-Farbwert angegeben wird, da diese Farbe anders verarbeitet wird.

Tabelle 5.4: hyperref-Optionen zum Setzen der Farbwerte einzelner Linktypen. Die Option `colorlinks` muss auf `true` gesetzt sein, damit die Farbe auch genutzt wird.

Option	*Standardwert*	*Beschreibung*
`linkcolor`	red	Farbe für einfache, dokumenteninterne Verweise
`anchorcolor`	black	Farbe für Linktext
`citecolor`	green	Farbe für Literaturveweise
`filecolor`	cyan	Farbe für Links, die Dateien öffnen
`menucolor`	red	Farbe für Acrobat Menüeinträge
`runcolor`	filecolor	Farbe für die Links, die Anmerkungen aufdecken
`urlcolor`	magenta	Farbe für URLs
`allcolors`		Weist allen Farbwerten den gleichen Wert zu

Tabelle 5.5: hyperref-Optionen zur Farbumschaltung von Rahmen. Die Werte müssen als Leerzeichengetrenntes rgb-Tripel übergeben werden.

Option	*Standardwert*	*Element*
`citebodercolor`	`0 1 0`	Literaturverweise
`filebordercolor`	`0 .5 .5`	Dateiverweise
`linkbordercolor`	`1 0 0`	Einfache Querverweise
`menubordercolor`	`1 0 0`	Menü-Links
`urlbodercolor`	`0 1 1`	URLs
`runbodercolor`	`0 .7 .7`	Ausführbare Links
`allbodercolors`		Alle Rahmenfarben

Tabelle 5.6: Metadaten setzen mit hyperref

Option	*Beschreibung*
`pdftitle=`*Titel*	Setzt das Feld „Titel“ in den `.pdf`-Eigenschaften
`pdfauthor=`*Autor*	Setzt das Feld „Autor“ in den `.pdf`-Eigenschaften
`pdfsubject=`*Thema*	Setzt das Feld „Thema“ in den `.pdf`-Eigenschaften
`pdfkeywords=`*Stichwörter*	Setzt das Feld „Stichwörter“ in den `.pdf`-Eigenschaften

→ Diese Randnotiz gehört zu einem Beispiel von Abschnitt 5.4.2

Zusätzlich bietet das hyperref-Paket die Möglichkeit die `.pdf`-Metadaten zu ändern. Die Metadaten sind sozusagen der Bucumschlag eines Dokuments. Dort werden Daten wie Autor und Titel hinterlegt. Diese Daten werden bei einer Suchanfrage vom Betriebssystem mit berücksichtigt und vereinfachen so die Archivierung. Bei Dokumenten, die für die Weitergabe oder gar Veröffentlichung bestimmt sind, sollte man diese Daten entsprechend setzen. Tabelle 5.6 zeigt die entsprechenden Funktionen.

Man sollte hierbei *immer* mindestens die Felder `pdfauthor` und `pfftitle` setzen. Zusammen mit der Option `hidelinks` werden Links nicht farbig gekennzeichnet (wie in diesem Dokument).

```
\usepackage[
    pdftitle={Kursskript:Einführung in LaTeX},
    pdfauthor={Marei Peischl},
    hidelinks
    ]{hyperref}
```

Neben den bereits genannten Funktionen, bietet hyperref eine Reihe von Makros für verschiedenste Referenzen:

`\autoref{`*Markername*`}`

Eine Alternative zum normalen `\ref` der Link wird um den Typnamen des Elements auf welches verlinkt wird ergänzt, z. B.:

`\autoref{sec:hyperref}`	Abschnitt 5.4.3

Die Namen werden über das babel-System übersetzt. Ihre Makros samt der Standarddefinitionen im Englischen und Deutschen finden sich in Tabelle 5.7.

Neben den hier bereits gezeigten Funktionen gibt es mit hyperref Möglichkeiten die `.pdf`-Funktionalität noch weiter zu erweitern. Somit kann man mithilfe dieses Paketes zum Beispiel auch `.pdf`-Formulare erzeugen. Für diese spezielleren Features sei auf die Paketanleitung [19] verwiesen.

Tabelle 5.7: `\autoref`-Namen für hyperref samt der deutschen Übersetzungen mit dem babel-System

Makro	*Deutsch*	*Englisch*
`\figureautorefname`	Abbildung	Figure
`\tableautorefname`	Tabelle	Table
`\partautorefname`	Teil	Part
`\appendixautorefname`	Anhang	Appendix
`\equationautorefname`	Gleichung	Equation
`\Itemautorefname`	Punkt	item
`\chapterautorefname`	Kapitel	chapter
`\sectionautorefname`	Abschnitt	section
`\subsectionautorefname`	Unterabschnitt	subsection
`\subsubsectionautorefname`	Unterunterabschnitt	subsubsection
`\paragraphautorefname`	Absatz	paragraph
`\subparagraphautorefname`	Unterabsatz	subparagraph
`\footnoteautorefname`	Fußnote	footnote
`\AMSautorefname`	Gleichung	Equation
`\theoremautorefname`	Theorem	Theorem
`\pageautorefname`	Seite	page

Teil II

Das Seitenlayout

6 Der Satzspiegel

> Der Satzspiegel ist wie der Rahmen eines Bildes. Ein echter Rembrandt in einem schiefen, bunten, neonfarbenen PVC-Rahmen wird immer wie eine billige Kopie wirken. Ebenso wird ein inhaltlich perfektes Dokument mit verkorkstem Satzspiegel nicht die Geltung erfahren, die es verdient.
>
> *Markus Kohm & Jens-Uwe Morawski* [11, S.485]

6.1 Satzspiegelkonstruktion mit typearea

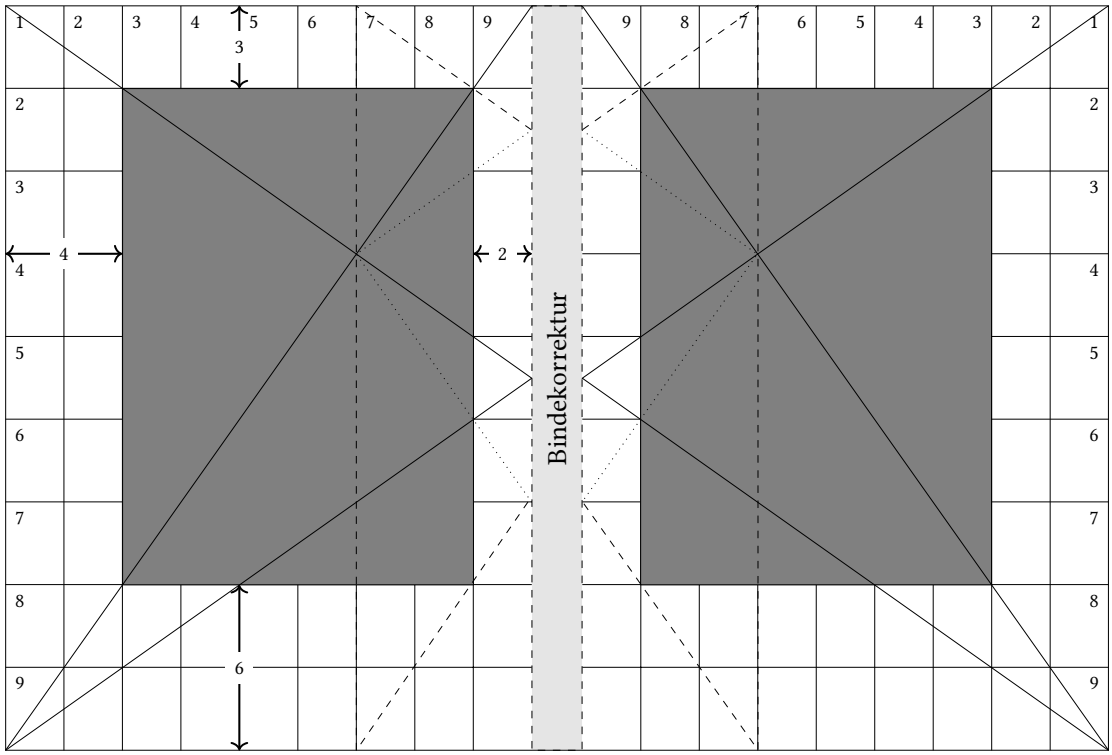


Abbildung 6.1: Doppelseite mit Satzspiegelkonstruktion in klassischer Neunerteilung (entspricht `DIV=9`) [nach 11, S.32]

Die Satzspiegelkonstruktion übernimmt bei KOMA-Klassen das in KOMA-Script integrierte typearea-Paket. Gleiche Ergebnisse in Nicht-KOMA-Klassen können durch zusätzliches laden dieses Paketes erfolgen. Im Folgenden wird nun zunächst das Prinzip der Satzspiegelkonstruktion durch Teilung betrachtet, mit dem man relativ einfach einen harmonischen Satzspiegel erzeugen kann.

Das Prinzip basiert auf der klassischen Neunerteilung (siehe Abbildung 6.1), bei der die Seite sowohl horizontal, als auch vertikal in neun Streifen geteilt wird. Über Berechnungen mit Hilfe des gerundeten ganzzahligen „Goldenen Schnittes“ ergibt sich das Ränderverhältnis

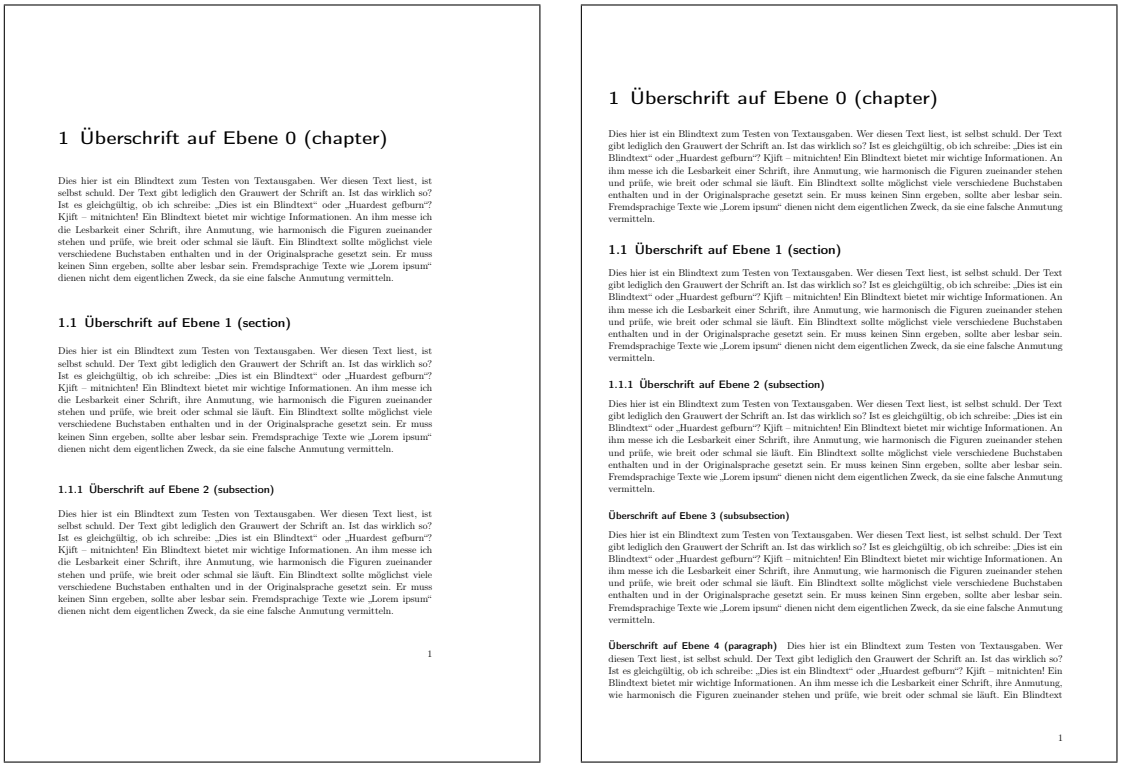


Abbildung 6.2: Vergleich der DIV-Werte 10 und 20 bei der Grundschriftgröße von 11 pt. `DIV=10` wirkt deutlich übersichtlicher und professioneller, wohingegen `DIV=20` lediglich Papier einspart.

innen:oben:außen:unten von 2:3:4:6 (genauer gesagt von 2:2,8:4:5,7 bei A4 ohne Bindekorrektur). Die Bindekorrektur (`BCOR`) bleibt davon als unsichtbarer Teil der Seite vollkommen unberücksichtigt.

Die Anzahl der Streifen kann über die KOMA-Klassenoption (oder typearea-Paketoption)

```
\documentclass[DIV=Anzahl, Optionen]{scr...}
\usepackage[DIV=Anzahl, Optionen]{typearea}
```

nur bei Nicht-KOMA-Klassen nötig

beliebig modifiziert werden. Das Teilungsverhältnis von 2:3:4:6 bleibt jedoch erhalten. Somit wird klar, dass bei einem größeren `DIV`-Wert auch ein größerer Teil der Seite beschrieben wird. Standardmäßig verwendet typearea `DIV=8` für die Grundschrift 10 pt, `DIV=10` für 11 pt und `DIV=12` für 12 pt Grundschriftgröße. Abbildung 6.2 zeigt den Unterschied zwischen `DIV=10` und `DIV=20` für die Grundschriftgröße 11 pt. Zusätzlich gibt es Paketoptionswerte für eine automatisierte Berechnung:

`DIV=calc` Berechnung des optimalen Satzspiegels (empfohlen für Papierformate ungleich A4)

`DIV=classic` Bestimmung des DIV-Wertes, welcher dem spätmittelalterlichen Buchseitenformat (Satzspiegelkonstruktion durch Kreisschlagen) am nächsten kommt.

`DIV=current` Neuberechnung des Satzspiegels mit aktuellem `DIV`-Wert

`DIV=last` Neuberechnung des Satzspiegels aufgrund des zuletzt gewählten `DIV`-Wertes

`DIV=default` Neuberechnung des Satzspiegels für das aktuelle Papierformat und die aktuelle Grundschrift

Bindet man eine neue Schriftart ein, sollte der Satzspiegel unter Berücksichtigung der Schrift neu berechnet werden. Normalerweise wird der Satzspiegel zu Beginn des Dokumenten (direkt bei `\begin{document}`) berechnet. Nimmt man anschließend noch Änderungen an Schriftsippe, Basisschriftgröße,... vor, so ist eine Neuberechnung erforderlich. Diese funktioniert auch innerhalb des Dokumentes mit den Befehlen:

```
\typearea[BCOR]{DIV}
\areaset[BCOR]{Breite}{Höhe}
\recalctypearea
```

Ein- und zweiseitiges Seitenlayout

Die vorherigen Beispiele betrachten immer ein doppelseitiges Layout. Jedoch ist es gerade bei Abschlussarbeiten häufig üblich einseitig zu drucken. In diesem Modus sind der linke und der rechte Rand gleich breit. Eine mögliche Bindekorrektur wird immer links angesetzt. Das Umschalten zwischen beiden Möglichkeiten ist über die Dokumentenklassenoption `twoside` möglich.

Neben den Rändern wird durch das Umschalten auch das Verhalten von LaTeX geändert wie die Seite gefüllt werden soll. So ist im doppelseitigen Satz das Makro

```
\flushbottom
```

aktiv. Es sorgt dafür, dass dehnbare vertikale Abstände (auch `\parskip`, selbst bei `parskip=false` vgl. Abschnitt 4.2.1) so weit gedehnt werden, dass die unterste Zeile auf jeder Seite die gleiche Position hat und bündig mit dem Rand abschließt.

```
\raggedbottom
```

unterdrückt dieses Verhalten und belässt die vertikalen Abstände bei ihren Idealwerten. Im einseitigen Satz ist `raggedbottom` aktiv, da es hier kaum bemerkbar ist, ob die unterste Zeile immer bündig mit dem Rand schließt oder nicht.

Kopf- und Fußzeile

Neben den Einstellungen für die Ausmaße des Satzspiegels ist selbstverständlich auch die Frage wichtig, welche Elemente überhaupt zum Satzspiegel gehören. Dass der Text dazugehört ist wohl selbstverständlich, wie es jedoch mit der Kopf- und Fußzeile aussieht ist im Allgemeinen nicht so einfach zu beantworten.

Wichtig ist hierbei wie die Seite bei Betrachtung aus der Ferne wirkt. Ob die Kopf-/Fußzeile mehr als Teil des Randes oder des Textblockes gesehen werden kann. Grundsätzlich kann man sagen, dass sobald sich eine Trennlinie zwischen (Fuß-)Kopfzeile und Textblock befindet, die (Fuß-)Kopfzeile zum Satzspiegel zu rechnen ist. Ebenso verhält es sich mit langen Kolumnentiteln. Ist die Kopf- oder Fußzeile sowohl am linken (inneren) als auch am rechten (äußeren) Rand beschrieben, so gehört sie ebenfalls zum Satzspiegel. Dahingegen sollte man eine (Fuß-)Kopfzeile, die lediglich die Seitennummer enthält zum Rand rechnen. Schwierig ist der Fall bei schwankenden Längen der Kolumnentitel. Es empfiehlt sich hierbei die Kopf-/Fußzeile in fraglichen Fällen eher zum Satzspiegel zu rechnen als zum Rand [8].

Die Einstellung was zum Satzspiegel gehört erfolgt über die Dokumentenklassen- bzw. typearea-Optionen

```
headinclude=true/false
footinclude=true/false
```

Diese Einstellung wird jedoch nur berücksichtigt solange KOMA-Script und damit das typearea-Paket den Satzspiegel bestimmt. Bei einer Einstellung mithilfe von geometry (siehe

folgender Abschnitt) greifen selbstverständlich alle typearea betreffenden Optionen nicht.

6.2 Das Seitenlayout manuell einstellen mit geometry

Manchmal ist es notwendig bestimmte Größen im Seitenlayout manuell festzusetzen. Das geometry-Paket von Hideo Umeki bietet eine sehr benutzerfreundliche Möglichkeit sowohl Papierformat als auch Satzspiegel manuell anzupassen. Diese Variante ist jedoch nicht sonderlich elegant, weswegen man auf diese Variante nur dann zurückgreifen sollte, wenn es absolut keine andere Möglichkeit gibt.

```
\usepackage[Option1, Option2, ...]{geometry}
```

Die Basis-Einstellung werden in der Regel über *Optionen* (siehe Tabelle 6.1) vorgenommen. Zusätzlich existieren Befehle, um das Layout innerhalb des Dokumentes zu ändern:

```
\geometry{Option1,...}
\newgeometry{Option1,...}
\restoregeometry
\savegeometry{Name}
\loadgeometry{Name}
```

Ändert das Layout entsprechend der *Optionen*.
Die Optionen werden zusätzlich zu vorherigen verwendet. Sollte nur in der Präambel verwendet werden., Überschreibt die bisherigen Optionen.
Die Papiergröße ist damit nicht einstellbar., Aktiviert wieder die ursprünglichen Einstellungen aus dem Header., Speichert die aktuellen Einstellungen unter *Name* ab., Aktiviert die unter *Name* gespeicherten Einstellungen.
Zusätzlich zu den genannten Optionen ist es auch möglich wie bei typearea den Kopf-, bzw. den Fußbereich mit zum Satzspiegel zu zählen[1]. Hierfür existieren die Optionen

```
includehead
includefoot
includeheadfoot
```

6.3 Mehrspaltiger Textsatz

Zweispaltiger Text kann mithilfe der Dokumentenklassenoption `twocolumn=true` erreicht werden. Zusätzlich existieren in Standard-LaTeX die Makros

```
\twocolumn[Überschrift]
\onecolumn
```

Beendet die aktuelle Seite. Die folgende Seite wird zweispaltig gesetzt. Die optionale Überschrift wird auf der neuen Seite über die Breite der gesamten Seite gesetzt., Beendet die aktuelle zweispaltige Seite und beginnt eine neue einspaltige.

Das multicol-Paket
Das multicol-Paket behebt einige Unschönheiten von Standard-LaTeX. Beispielsweise füllt es auf jeder Seite beide Spalten gleichmäßig auf und nicht zuerst die linke Spalte vollständig. Zudem erlaubt es mehrspaltigen Textsatz bis hin zu zehnspaltigem Satz. Mit der Umgebung

[1] Für genauere Hinweise zu diesem Thema, siehe Ende des Abschnitts 6.1.

Tabelle 6.1: Übersicht über die wichtigsten Paketoptionen von geometry

Option	*mögliche Werte*	*Erläuterung*
`paper`	`a0paper`, `a1paper`, …, `a6paper`, `b0paper`, `b1paper`, …, `b6paper`, `c0paper`, `c1paper`, …, `c6paper`, `b0j`, `b1j`, …, `b6j`, `letterpaper`, `executivepaper`, `legalpaper`, …	Papierformat
`screen`		225 mm × 180 mm (Präsentationen)
`paperwidth`	*Länge*	
`paperheight`	*Länge*	
`papersize`	{*Breite*,*Höhe*} oder *Länge*	manuelle Angabe der Seitengröße; einzelne Angabe erzeugt ein quadratisches Format
`landscape`		Querformat
`portrait`		Hochformat
`left`/`inner`	*Länge*	linker/innerer Rand
`right`/`outer`	*Länge*	rechter/äußerer Rand
`top`	*Länge*	oberer Rand
`bottom`	*Länge*	unterer Rand
`hmarginratio`	*links (innen)*:*rechts (außen)*	Größenverhältnis der Seitenränder. Standardwert ist 1:1 für einseitigen und 2:3 für zweiseitigen Satz.
`vmarginratio`	*oben*:*unten*	oberer:unterer Rand; Standard ist 2:3
`bindingoffset`	*Länge*	Bindekorrektur

```
\begin{multicols}{Spaltenanzahl}[Überschrift]
Text der mehrspaltig gesetzt werden soll
\end{multicols}
```

kann man mitten auf der Seite in mehrspaltigen Text wechseln. Die Überschrift wird über alle Spalten gesetzt, bevor der mehrspaltige Text beginnt.

7 Der Seitenstil

7.1 Das Konzept der Seitenstile

Die Gepflogenheiten der Typografie erfordern es häufig, verschiedene Seitenlayouts innerhalb eines Dokumentes zu verwenden. So wird die Titelseite beispielsweise nicht nummeriert und die Anfangsseite eines neuen Kapitels besitzt meistens keine Kopfzeile. Dieses Umschalten funktioniert bei LaTeX oft automatisch über die Dokumentenklasse, allerdings kann es erforderlich sein, manuell einzugreifen, zum Beispiel um ein Bild zu positionieren, das die Kopfzeile teilweise überdeckt. Das manuelle Umschalten kann über die Befehle

```
\pagestyle{Stil}
\thispagestyle{lokaler Stil}
```

erfolgen. Dabei ist `\pagestyle` ein Schalter, der den Seitenstil global umstellt, wohingegen sich das Makro `\thispagestyle` lediglich auf die aktuelle Seite bezieht. Außerdem bietet KOMA-Script durch die Befehle

`\titlepagestyle`	Stil der Seite mit der Titelei bei `titlepage=false` (Titelkopf)
`\partpagestyle`	Stil der Anfangsseiten von `\part` (nicht bei scrartcl)
`\chapterpagestyle`	Stil der Kapitelanfangsseiten (nicht bei scrartcl)
`\indexpagestyle`	Stil der Anfangsseite des Index

die Möglichkeit den Stil bestimmter Seitentypen mithilfe von `\renewcommand` zu ändern. Diese Makros entsprechen dem entsprechenden Seitenstilnamen.

Die vordefinierten Seitenstile finden sich in Tabelle 7.1. Beim Seitenstil myheadings wird der Kolumnentitel nicht automatisch ausgegeben. Hier muss den Inhalt manuell setzen, hat aber dadurch die Wahl, was genau in der Kopfzeile ausgegeben werden soll. Hierfür gibt es die Befehle

```
\markboth{linke Marke}{rechte Marke}
\markright{rechte Marke}
```

Die rechte Marke wird im Kopf rechter (ungerader) Seiten verwendet, die linke Marke analog links. Bei einseitigem Satz gibt es nur die rechte Marke. Für weitere Anpassungen und den Befehl `\markleft` empfiehlt es sich das scrlayer-scrpage-Paket zu verwenden (siehe Abschnitt 7.2).

7.2 Kopf- und Fußzeilen mit scrlayer-scrpage

Mit dem Paket scrlayer-scrpage lassen sich recht einfach komplexere Kopf- und Fußzeilen erzeugen. Es basiert auf dem Paket scrlayer, welches ein Ebenenmodell, wie es von Bildbearbeitungsprogrammen bekannt sein könnte, sowie ein darauf basierendes Seitenstil-Modell anbietet. Zu den grundlegenden Funktionen gehören zwei individuell konfigurierbare Seitenstile: `scrheadings` und `plain.scrheadings`. Diese Seitenstile können, wie in Kapitel 7 beschrieben, mithilfe von

Tabelle 7.1: Übersicht über die vordefinierten Seitenstile

Seitenstil	*Erklärung*
`empty`	Kopf und Fußzeilen bleiben vollständig leer.
`headings`	Seitenstil für lebende Kolumnentitel. Die Überschriften werden automatisch in die Kopfzeile übernommen. In den Klassen scrbook und scrreprt werden, im doppelseitigen Layout die Kapitel- (links) und Abschnittsüberschriften (rechts) außen in der Kopfzeile wiederholt. Die Seitenzahl befindet sich im Fuß außen. Im einseitigen Layout werden nur die Überschriften der Kapitel verwendet und wie die Pagina zentriert ausgegeben.
`myheadings`	Eigene Kolumnentitel. Diese werden nicht automatisch erzeugt, sondern mit den Anweisungen `\markboth` und `\markright` gesetzt. Ansonsten entspricht dieser Stil dem Stil `headings`.
`plain`	Hierbei werden keinerlei Kolumnentitel verwendet, sondern lediglich die Seitenzahl im Fuß ausgegeben. Im doppelseitigen Layout außen, sonst zentriert.

```
\pagestyle{scrheadings}
```

aktiviert werden. Damit wird auch automatisch der Seitenstil `plain` automatisch durch `plain.scrheadings` ersetzt. Man kann jedoch auch `plain.scrheadings` direkt aktivieren.

Für automatische Kolumnentitel sollte die Option

```
\usepackage[automark]{scrlayer-scrpage}
```

gesetzt werden. Standardmäßig werden somit die Marken so gesetzt, dass in Klassen mit `\chapter` die Kapitelüberschrift links und die Abschnittsüberschrift rechts gesetzt wird. In den übrigen Klassen ist dieses Verhalten eine Ebene nach unten verschoben.

7.2.1 Höhe von Kopf und Fuß

Normalerweise ist die Fußzeile bei LaTeX-Dokumenten immer einzeilig. scrlayer führt nun eine Höhe `\footheight` analog zu `\headheight` ein. scrlayer-scrpage interpretiert den Abstand `\footskip` so, „dass es den Abstand der letzten Grundlinie des Texbereichs von der ersten Standard-Grundlinie des Fußes darstellt“ [8, S.264].

7.2.2 Seitenstile modifizieren

Die Modifiationen der Seitenstile funktionieren nun ähnlich zu `myheadings` mit Befehlen der Form:

```
\Feld[plain.scrheadings-Inhalt]{scrheadings-Inhalt}
```

Die Benennung der Felder kann Abbildung 7.1 entnommen werden. Will man nun zum Beispiel erreichen, dass die Seitenzahl, sowohl bei `scrheadings`, als auch bei `plain.scrheadings` auf den rechten Seiten, außen in der Kopfzeile steht, benutzt man einfach

```
\rohead[\pagemark]{\pagemark}
```

Sollen zusätzlich bei scrheadings die Kolumnentitel auf allen Seiten im Kopf innen stehen, setzt man zusätzlich zum obigen Makro

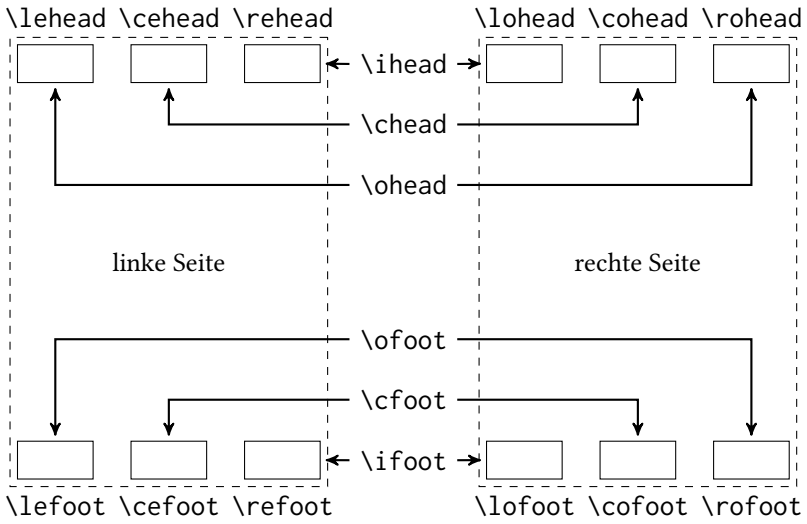


Abbildung 7.1: Zuordnung der Befehle für scrlayer-scrpage [8, S.270/273]

```
\ihead{\headmark}
```

Die Kopf- und Fußzeile sind jedoch vor den Änderungen meistens nicht leer. Um zu vermeiden, dass zusätzliche Einträge in den Kopf- beziehungsweise Fußzeilen auftreten, die noch von der Dokumentenklasse stammen, benutzt man am besten

```
\clearmainofpairofpagestyles
\clearplainofpairofpagestyles
\clearpairofpagestyles
```

`\clearmainofpairofpagestyles` leert dabei lediglich alle Felder des scrheadings-Seitenstils, wohingegen `\clearplainofpairofpagestyles` dasselbe für plain.scrheadings erfüllt. Das dritte Makro löscht sämtliche Kopf und Fußzeilen beider Seitenstile. Dies erspart beispielsweise viel Schreibarbeit, wenn man nur zwei der sechs möglichen Felder benutzen will. Zum Beispiel bewirken die beiden folgenden Varianten dasselbe:

```
\clearpairofpagestyles
\ohead{\headmark}
\ofoot[\pagemark]{\pagemark}
```

```
\ihead[]{}
\chead[]{}
\ohead{\headmark}
\ifoot[]{}
\cfoot[]{}
\ofoot[\pagemark]{\pagemark}
```

In den vorangehenden Beispielen wurden Befehle benutzt, die noch gar nicht besprochen wurden. Dies wird sofort nachgeholt.

Um die Kolumnentitel möglichst einfach zu setzen, gibt es vordefinierte Befehle, die es vereinfachen lebende Kolumnentitel[1] zu erzeugen.

`\leftmark`, `\rightmark` Diese Befehle greifen auf die Kolumnentitel zurück, die für die linke

[1] Lebende Kolumnentitel sind Seitenüberschriften, die sich im Verlauf des Buches ändern. Zum Beispiel die Überschrift der Seite durch den jeweils aktuellen Kapitelnamen zählt zu den lebenden Kolumnentiteln. Tote Kolumnentitel ändern dagegen sich im Verlauf des Werkes nicht.

bzw. für die rechte Seite gedacht sind. (Standard: links Kapitelnummer mit -name; rechts Abschnittsnummer mit -name)

\headmark Dieser Befehl ermöglicht es ebenfalls auf den Inhalt der Kolumnentitel zurückzugreifen. Jedoch muss man hierbei nicht auf die richtige Zuordnung von linker und rechter Seite achten.

\pagemark Entspricht der Seitenzahl, der Stil kann mit `\pagenumbering{`*Stil*`}` geändert werden, vgl. Tabelle 5.1.

\manualmark Wie der Name schon sagt, deaktiviert dieser Befehl die automatische Aktualisierung der Kolumnentitel. Man kann nun mit den Befehlen `\markboth` bzw. `\markright` die Kolumnentitel manuell ändern.

\automark*[*rechte Seite***]{***linke Seite***}** Dieses Makro aktiviert die automatische Aktualisierung des Kolumnentitels. Für die Parameter kann man einfach Gliederungsebenen (Abschnitt 3.6) einsetzen. Möchte man erreichen, dass auf den linken Seiten der Titel des Kapitels und auf den rechten Seiten der des Abschnittes steht, benötigt man den Befehl

```
\automark[section]{chapter}
```

`\leftmark` und `\rightmark` werden dementsprechend modifiziert. Die Sternchenversion unterscheidet sich dadurch, dass `\automark` alle vorherigen Vorgaben aufhebt und `\automark*` lediglich die Aktionen für die angegebenen Gliederungsebenen ändert.

7.2.3 Formatierung der Kopf- und Fußzeilen

Eine Änderung der Schriftart innerhalb der Kopf und Fußzeilen funktioniert vollkommen analog zum Ändern der Überschriftenschriftart (siehe Abschnitt 4.1.5), z. B.:

```
\setkomafont{pagehead}{\normalfont\sffamily\bfseries}
\setkomafont{pagefoot}{\normalfont\sffamily}
\setkomafont{pagenumber}{\normalfont\slshape}
```

`headwidth=`*Breite*`:`*Verschiebung*	
`footwidth=`*Breite*`:`*Verschiebung*	

Bei Standardeinstellungen entspricht die Breite der Kopf bzw. Fußzeile der des Textblockes. Manchmal ist es jedoch nötig diese Breiten entsprechend anzupassen. Für Standardfälle gibt es symbolische Werte für das Argument *Breitenoption* (Die Namen sind selbsterklärend.): `page`, `text`, `textwithmarginpar`, `head`, `foot`.

Die Verschiebung wird in richtung des äußeren Randes gemessen. Es ist auch möglich zwei verschiedene Offsets anzugeben. In diesem Fall wird der erste für ungerade (rechte) Seiten und der zweite für gerade (linke) Seiten verwendet.

Neben der Schriftart und der Breite existieren verschiedene Trennlinien, um den Seitenkopf und -fuß zu formatieren.

`headtopline=`*Dicke*`:`*Länge*	Linie über dem Seitenkopf
`headsepline=`*Dicke*`:`*Länge*	Linie zwischen Kopf und Textkörper
`footsepline=`*Dicke*`:`*Länge*	Linie zwischen Text und Fuß
`footbotline=`*Dicke*`:`*Länge*	Linie unter dem Fuß

Standardmäßig werden diese Linien zentriert, um die Ausrichtung zu ändern, gibt es drei verschiedene Paketoptionen für scrlayer-scrpage:

`ilines`	innen bündige Linien
`clines`	zentrierte Linien
`olines`	außen bündige Linien

Um neben der Länge und der Dicke auch andere Eigenschaften, wie beispielsweise die Farbe zu ändern, existieren entsprechende Komafonts. Möchte man z. B. die `headtopline` blau einfärben, so schreibt man (nachdem das color-Paket geladen wurde):

```
\addtokomafont{headtopline}{\color{blue}}
```

Um die Einstellung der Trennlinien auch für `plain.scrheadings` zu übernehmen gibt es zusätzliche Optionen, denen man einen Wahrheitswert zuweisen kann:

`plainheadtopline=`*Wahrheitswert*
`plainheadsepline=`*Wahrheitswert*
`plainfootsepline=`*Wahrheitswert*
`plainfootbotline=`*Wahrheitswert*

Teil III

Weitere wichtige Elemente

8 Aufzählungen

Hierfür existieren drei Umgebungen:

```
\begin{itemize} \item ...\item ...\end{itemize}
\begin{enumerate} \item ... \item ... \end{enumerate}
\begin{description} \item[Name1] ... \end{description}
```

Bei der `itemize`-Umgebung werden die Punkte durch ein Aufzählungszeichen getrennt und die einzelnen Texte voneinander abgesetzt. Bei der `enumerate`-Umgebung wird fortlaufend numeriert und bei der `description`-Umgebung erscheint der optionale Name, als zu erläuternder Begriff fett gedruckt.

Die Aufzählungsumgebungen können bis zu viermal ineinander verschachtelt werden und je nach Verschachtelungstiefe ändert sich das Markierungszeichen sowie die Einrückung am Anfang, siehe Beispiel 3.

8.1 Aufzählungslabels ändern

Die Aufzählungszeichen kann über die Umdefinition der entsprechenden Makros verändert werden:

```
\labelitemi   \labelitemii   \labelitemiii   \labelitemiv
\labelenumi   \labelenumii   \labelenumiii   \labelenumiv
```

Es sei noch erwähnt, dass LaTeX bei `enumerate` mit den Aufzählungsebenen vier Counter (vgl. Abschnitt 5.1), `enumi`, `enumii`, `enumiii` und `enumiv` betreibt. Das Beispiel liefert in der zweiten Ebene die Nummerierung mit Abschnittsnummer und Zähler der zweiten Stufe.

```
\renewcommand{\labelenumii}{\thesection-\arabic{enumii}.)}
```

Der Schriftyp kann über die entsprechenden KOMAfont-Elemente (Abschnitt 4.1.5) angepasst werden.

Allgemein ist es (wie für andere Elemente auch) empfehlenswert, Aufzählungen für das gesamte Dokument einheitlich zu gestalten.

8.2 KOMA-Auflistung

KOMA-Script liefert noch eine weitere Auflistungsumgebung:

```
\begin{labeling}[Trennzeichen]{Muster} ... \end{labeling}
```

Das Muster gibt die Einrücktiefe bei den Stichpunkten an und das Trennzeichen ist beliebig. Genau wie bei den Standardaufzählugen beginnt jeder Punkt mit einem `\item`. Beispiel 4 zeigt die Funktionsweise, für genauere Informationen und weitere Beispiele sei auf [11] verwiesen.

```latex
\begin{itemize}
\item 1. Stufe 1. Zeile
\begin{itemize}
\item 2. Stufe 1. Zeile
\begin{itemize}
\item dritte Stufe 1. Zeile
\begin{itemize}
\item vierte Stufe 1. Zeile
\item vierte Stufe 2. Zeile
\end{itemize}
\item dritte Stufe 2. Zeile
\end{itemize}
\item 2. Stufe 2. Zeile
\end{itemize}
\item 1. Stufe 2. Zeile
\end{itemize}
```

- 1. Stufe 1. Zeile
  - 2. Stufe 1. Zeile
    - dritte Stufe 1. Zeile
      - vierte Stufe 1. Zeile
      - vierte Stufe 2. Zeile
    - dritte Stufe 2. Zeile
  - 2. Stufe 2. Zeile
- 1. Stufe 2. Zeile

Beispiel 3: Eine verschachtelte Aufzählung

```latex
\begin{labeling}[~--]{description}
    \item[itemize] Listen mit Aufzählungspunkten
    \item[enumerate] Nummerierte Listen
    \item[description] Begriffserläuterungen
    \item[labeling] Personalisierte Listen
\end{labeling}
```

itemize – Listen mit Aufzählungspunkten

enumerate – Nummerierte Listen

description – Begriffserläuterungen

labeling – Personalisierte Listen

Beispiel 4: Beispiel für die Verwendung der `labeling`-Umgebung [nach 11, S. 132]

8.3 Eigene Auflistungen und Listenlayouts (Das enumitem-Paket)

```
\usepackage{enumitem}
```

definiert zu den drei Umgebungen ein zusätzliches optionales Argument, worüber viele Anpassungen gemacht werden können.
Hier sollen nur wenige Möglichkeiten gezeigt werden, alle Befehle und Optionen finden sich in der Paketdokumentation [6].

Eine mögliche Option ist `label`. Damit lässt sich die Beschriftung der Listenpunkte für die jeweilige Ebene ändern. Die Änderung wirkt aber nur innerhalb der Umgebung, wo die Option gesetzt wurde. Für die `itemize`-Umgebung kann man beliebige Symbole benutzen wie z.B das Pfeilsymbol → (`$\rightarrow$`), für die `enumerate`-Umgebung gibt es `\alph*`, `\Alph*`, `\arabic*`, `\roman*` und `Roman*`, wobei statt dem Stern auch ein Zählername in geschweiften Klammern stehen kann.

Sind zwei Listen durch Text getrennt und die zweite Liste soll die erste fortführen (und nicht wieder mit der Zählung von vorne beginnen), so gibt man bei den Optionen `resume` mit an (was natürlich auch über `\setcounter` machbar wäre, siehe Kapitel 5.1). Sollen zusätzlich die Formatdefinitionen aus der ersten Liste übernommen werden (falls in der Ebene vorhanden) so benutzt man `resume*`.

```
\begin{itemize}[Optionen]...
\begin{enumerate}[Optionen]...
\begin{description}[Optionen]...
```

Möchte man die Beschriftung dauerhaft ändern so kann man die bestehenden Listen (`itemize`, `enumerate`, `description`) neu definieren oder komplett neue erstellen.

```
\newlist{Listenname}{Typ}{max. Tiefe}
\renewlist{Listenname}{Typ}{max. Tiefe}
```

Für *Typ* kann man `enumerate`, `itemize` oder `description` einsetzen. *Listenname* ist ein beliebiger Name für die neue Liste, und *max. Tiefe* gibt die Anzahl der Ebenen an, die die Aufzählung maximal haben darf. Möchte man die Standardlisten neu definieren so benutzt man stattdessen `\renewlist`. Als nächstes müssen die Eigenschaften der Liste festgelegt werden. Dies geschieht mit

```
\setlist[Listenname, Ebene]{Optionen}
```

Wenn man bei der Definition mehrere Ebenen zulässt, so muss für jede Ebene eine Beschriftung definiert sein. Wenn man *Ebene* weglässt, gilt die *Option* für alle Ebenen, kann aber für einzelne Ebenen wieder überschrieben werden.

Zusätzlich zur Option, die das Label verändert gibt es noch eine Reihe von Optionen um die Formatierung der Liste an sich zu ändern. Im Folgenden finden sich einige Beispiele:

`leftmargin=`*Länge* Rückt den Beschriftungstext der Labels um *Länge* ein, die Labels beginnen somit bei `\leftmargin-\labelwidth`.

`labelwidth=`*Länge* Setzt die Breite des Platzes der für die Labels reserviert ist auf *Länge*

`align=left/right` Richtet die Labels entweder links- oder rechtsbündig aus

`itemsep=`*Länge* Setzt den Abstand zwischen den einzelnen Items auf *Länge*.

`parsep=`*Länge*	Setzt den Abstand zur Absatzkennzeichnung innerhalb der Liste auf *Länge*.
`start=`*Nummer*	Setzt den Startwert der Liste auf *Nummer*.

9 Das Prinzip der Boxen

TeX berechnet viele Elemente, wie z. B. Absatzumbrüche, nicht am Text sondern an einer Reihe von Boxen ohne Inhalt, die lediglich als Platzhalter fungieren.

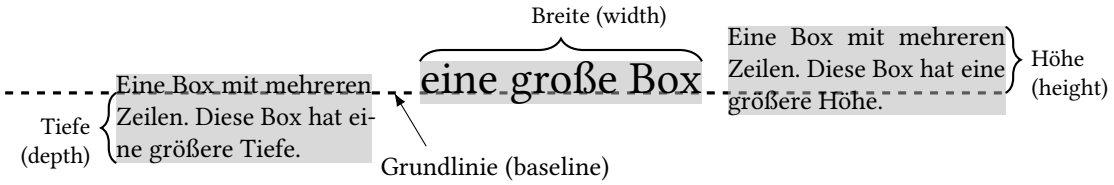
vieleklleineBoxen

Wenn zwischen zwei Zeichen keine Trennung möglich sein soll, so werden sie zu einer großen Box zusammengefasst:

eine große Box

Eine gute Vertrautheit mit diesen Prinzipien ist wichtig, da sie von TEX beim Satz von Tabulatoren und Tabellen verwendet werden.

Unabhängig von Ihrem Inhalt verfügt jede Box über drei Größenparameter, die es ermöglichen den Platzhalter für die Box zu rekonstruieren. Diese Größen sind die einzigen Informationen über die Box, die LATEX außerhalb der Box kennt:



9.1 Die drei Bearbeitungsmodi

Innerhalb des Textkörpers kennt LATEX drei unterschiedliche Modi:

1. **Paragraph Mode:** normaler Textmodus
   z. B. `\begin{document}...\end{document}`, `\parbox{2cm}{...}`

2. **Mathematical Mode:** mathematischer Modus
   z. B. `\begin{equation}...\end{equation}` (Kapitel 14)

3. **LR Mode:** (Left–Right)-Mode; Wie Paragraph Mode, jedoch mit dem Verbot des Zeilenumbruchs. Der Text wird ohne Umbruch von links nach rechts gesetzt.
   z. B. `\mbox{}, \fbox{}` (Abschnitt 9.2)

Innerhalb einer bestimmten Box kann immer nur ein Modus aktiv sein. Eine gerade zu Beginn recht häufige Fehlerquelle ist somit, dass man sich im falschen Modus befindet. So funktionieren mathematische Befehle nun einmal nur im Mathemodus. Häufig kann man diese Probleme, wie zum Beispiel den verbotenen Zeilenumbruch im LR-Mode dadurch umgehen, dass man innerhalb „des Modus“ eine Box im Paragraph Mode erstellt.

9.2 Rahmenboxen

Die einfachste Möglichkeit Code zu einer großen Box zusammenzufassen bieten die Boxen im LR-Mode, die sogenannten mboxen.

`\makebox[`*Breite*`][`*Position*`]{`*Text*`}`	Diese Variante besitzt mehr Argumente
`\mbox{`*Text*`}`	

Diese Boxen sind jedoch unsichtbar, weshalb sich ihre Funktionsweise leichter mit den sichtbaren Rahmenboxen verdeutlichen lässt.

`\frame{`*Text*`}`

Das `\frame`-Makro verhält sich identisch zur `\mbox` und setzt somit sein Argument in den LR-Mode. Der Rahmen ist hierbei jedoch sichtbar:

```
\frame{ein bisschen Text in einem \texttt{\textbackslash{}frame}}
```

ein bisschen Text in einem `\frame`

Die Abstände sind jedoch, wie man sieht, für Text vollkommen ungeeignet. Frames sollten somit lediglich zum Umrahmen von Objekten, wie z. B. Abbildungen verwendet werden. Um einen einfachen Rahmen um Text zu ziehen gibt es die `\fbox`. Die ausgeschriebene Variante dieses Makros ermöglicht es weitere Parameter anzugeben:

`\framebox[`*Breite*`][`*Position*`]{`*Text*`}`
`\fbox{`*Text*`}`

Der Text wird entsprechend dem LR-Modus nicht umgebrochen und ragt über die Box hinaus, falls diese für den Text zu klein ist. Als Position zur laufenden Zeile gelten die Optionen l, r, c, s für left, right, center und stretched (gleichmäßig verteilt). Beispiel:

```
\framebox[14cm][r]{\small Dies ist Text innerhalb einer 14\,cm breiten
\texttt{framebox} mit Ausrichtung rechts.}
```

Dies ist Text innerhalb einer 14 cm breiten `framebox` mit Ausrichtung rechts.

Die Dicke der Rahmen, sowie der Abstand des Rahmens vom Inhalt, kann über die beiden Längen

`\fboxrule`	bestimmt die Dicke der Rahmenlinien
`\fboxsep`	bestimmt den Abstand zwischen Rahmen und Inhalt

gesteuert werden, vgl. Abschnitt 5.2.

Positionierung von LR-Boxen

`\raisebox{`*Offset*`}[`*Oberlänge*`][`*Unterlänge*`]{`*Text*`}`

`\raisebox` erzeugt ebenfalls eine `\mbox`. Jedoch können die darin enthaltenen Boxen (bzw. Zeichen) mit einem beliebigen vertikalen Offset zur laufenden Zeile positioniert werden. Ober- und Unterlänge sind die obere und untere Grenze der Box relativ zur Grundlinie und definieren den Abstand zur nächsten und vorherigen Zeile. Die Wirkungsweise wird wohl am deutlichsten an einem Beispiel, bei dem die Boxen sichtbar gemacht werden mit dem `\fbox`-Kommando. Im zweiten Beispiel wurde die Ober- und Unterlänge mit 5 ex bzw. 3 ex korrigiert (Beispiel 5).

```
vorherige Zeile . . . \\
. . . \raisebox{1ex}{1ex oben}\raisebox{-1ex}{1ex
unten}\fbox{\raisebox{2ex}[5ex][3ex]{2ex
hoch}}\fbox{\raisebox{-2ex}[5ex][3ex]{2ex unten}}. . . \\
. . . nächste Zeile
```

vorherige Zeile . . . . . vorherige Zeile . . . . . vorherige Zeile

laufende Zeile 1ex oben 1ex unten 2ex hoch 2ex unten laufende Zeile nächste Zeile nächste Zeile

nächste Zeile . . . . . nächste Zeile

Beispiel 5: Unterschiedliche Möglichkeiten zur Größe und Positionierung von Boxen

```
\begin{minipage}[b]{4.6cm}
    Dies ist . . .
\end{minipage}\qquad
\parbox{2cm}{Mitte . . . }\qquad
\begin{minipage}[t]{4cm}
    die oberste . . .
\end{minipage}
```

Dies ist eine minipage-Umgebung, sie erzeugt eine vertikale Box wie der \parbox-Befehl. Die unterste Zeile dieser minipage ist auf die

Mitte dieser schmalen \parbox ausgerichtet, auf die andererseits

die oberste Zeile der rechten minipage ausgerichtet ist.

Beispiel 6: Positionsparameter für Absatzboxen

9.3 Absatzboxen

Es stellt sich auch die Notwendigkeit nach Boxen mit Zeilenumbruch. Hierfür gibt es die folgenden zwei Möglichkeiten, wobei die `minipage`-Variante die mächtigere darstellt und wieder alle Möglichkeiten des Textmodus eröffnet:

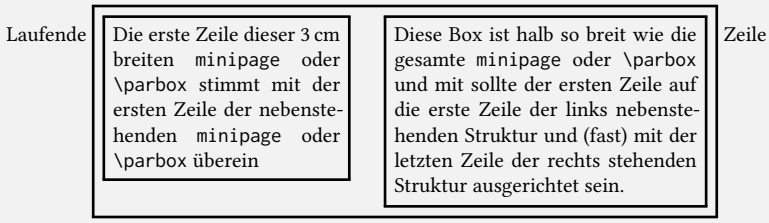
```
\parbox[Position][Höhe][vertikal Pos.]{Breite}{Text}
\begin{minipage}[Position][Höhe][vertikale Pos.]{Breite}Text\end{minipage}
```

Als Position zur laufenden Zeile können die Optionen `t`, `b`, `c` für top, bottom und center und als vertikale Positionen innerhalb der box auch `t`, `b`, `c` für top, bottom und center Verwendung finden, vgl. Beispiel 6

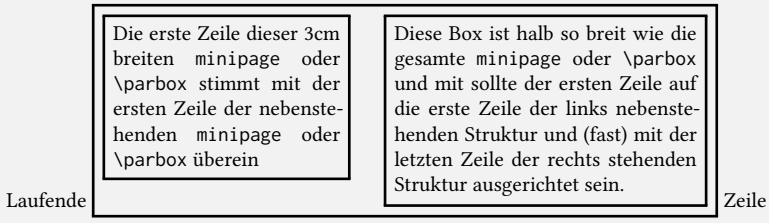
Innerhalb einer Minipage wird eine Parbox als eine Bearbeitungseinheit angesehen, also wie ein einzelnes Zeichen, was gerne zu überraschenden Ergebnissen führt. So bringt die Kombination

```
\begin{minipage}[b]{...}
    \parbox[t]{...}{...}...\parbox[t]{...}{...}
\end{minipage}
```

nicht das gewünschte Ergebnis von zwei oben gleich ausgerichteten Boxen deren unterste Zeile zur laufenden Zeile ausgerichtet ist:



Behoben werden kann das ganze mit einer sogenannten Nullzeile. Sie bewirkt, dass die minipage zweizeilig wird und somit an der untersten Zeile ausgerichtet werden kann.



Erzeugt mit dem Code:

```
\begin{minipage}[b]{..}
\parbox[t]{..}{..} \hfill \parbox[t]{..}{..}\vspace{0pt}
\end{minipage}
```

9.4 Balkenboxen

Ein einfacher Balken wird mit

`\rule[`*vert. Offset*`]{`*Breite*`}{`*Höhe*`}`

gebildet, z. B.  (`\rule[.5em]{1cm}{0.2cm}`) Er kann mit einer Höhen- oder Breitenangabe von 0 pt unsichtbar gemacht werden. Dies ist zum Beispiel für vertikale/horizontale Positionierungen anstelle von \vspace/\hspace sinnvoll.

9.5 Boxen speichern

Manchmal kann es vorkommen, dass man eine bestimmte Box, also ein Element mehrfach benötigt. Hierfür bietet LaTeX die Möglichkeit, ganze Boxen abzuspeichern um Sie mehrfach verwenden zu können. Hierzu muss zunächst eine neue Speicherbox definiert werden:

`\newsavebox{`*\Boxname*`}`

Um unter dem Makro \Boxname dann etwas zu speichern gibt es drei verschiedene Möglichkeiten:

```
\sbox{\Boxname}{Code}
\savebox{\Boxname}[Breite][Position]{Code}
\begin{lrbox}{\Boxname} Code \end{lrbox}
```

Anschließend kann die Box immer wieder eingefügt werden.

```
\usebox{\Boxname}
```

```
\newsavebox{\mybox}
\savebox{\mybox}[2cm]{\fcolorbox{red}{white}{Text}}
laufende Zeile\usebox{\mybox}laufende Zeile
```

laufende Zeile Text laufende Zeile

10 Grafiken

Dieses Kapitel befasst sich lediglich mit den einbinden von Grafiken. Für die Positionierung, Bildbeschriftung oder die Erzeugung eines Abbildungsverzeichnisses, wird auf Kapitel 12 verwiesen.

Bilder können mit dem Paket graphicx ganz einfach eingebunden werden. Das Makro hierfür lautet:

`\includegraphics[`*Optionen*`]{`*Bildpfad*`}`

Es platziert die unter *Bildpfad* gespeicherte Datei an der Position des Makros. Mithilfe der Optionen kann man Größe und Drehwinkel des Bildes einstellen:

`width=`*Breite*
`height=`*Höhe*
`scale=`*Faktor* `angle=`*Winkel*

Es ist auch möglich diese Optionen zu kombinieren. Jedoch sollte dabei darauf geachtet werden, dass die Kombinationen sinn ergibt. Zum Beispiel wird bei Angabe von Breite und Höhe das Bild verzerrt. Bei den Größenangaben ist es empfehlenswert, das Bild mit dem Satzspiegel skalieren zu lassen. So passt eine Abbildung auch in den Satzspiegel, falls später die Ränder oder das Papierformat geändert werden.

Die möglichen Bildformate hängen hierbei vom verwendeten Compiler ab. Wird `pdflatex` verwendet, können `.pdf`, `.png` und `.jpg`-Dateien eingebunden werden, wird `latex` dagegen verwendet ist man auf PostScript-Dateien, wie `.ps` und `.eps`, eingeschränkt. Mit entsprechenden Zusätzen kann auch `pdflatex` PostScript verarbeiten. In der TEX Live-Distribution ist hierfür nicht einmal ein Zusatzpaket notwendig. Bei MiKTEX benötigt man hierfür zusätzlich das Paket epstopdf.

Im Allgemeinen sind Vektorgrafiken zu bevorzugen. Diese sind, falls es sich um echte Vektorgrafiken handelt, vollständig skalierbar. Falls ein Bild jedoch nur als `.png` oder `.jpg` vorliergt, wird eine Umwandlung die Qualität selbstverständlich nicht verbessern.

Die Bilddatei sollte, damit LATEX sie findet, im selben Ordner, wie das zu kompilierende Dokument liegen. Falls Sie in einem Unterordner liegt, was für die bessere Übersicht der Dateistruktur sinnvoller ist, muss der Unterordner angegeben werden. Bei der Pfadangabe müssen dann jedoch die \ durch / ersetzt werden. Das folgende Beispiel bindet eine Abbildung namens „Grafik“ im Unterordner „Bilder“ ein.

```
\includegraphics[width=\linewidth]{Bilder/Grafik}
```

Absolute Pfadangaben (z. B. `C:/Users/Marei/LaTeX/Bilder/Grafik`) sind nicht sinnvoll, da die Datei dann entweder außerhalb des Projektordners liegt und bei einem Kopiervorgang schnell vergessen wird oder der Pfad nicht mehr funktioniert, sobald der Projektordner verschoben wird.

11 Tabellen

Mit den box-Befehlen können beliebige tabellenartige Strukturen erzeugt werden, jedoch würden sich dabei viele Arbeitsschritte wiederholen. Sinnvoller ist es daher, folgende Umgebungen zu verwenden:

`\begin{tabular}[`*Position*`]{`*Spaltenformatierung*`}`…`\end{tabular}`	
`\begin{array}[`*Position*`]{`*Spaltenformatierung*`}`…`\end{array}`	Mathe-Modus

`array` ist dabei die Variante für den Mathe-Modus (vgl. Kapitel 14). Die Funktionsweise ist jedoch identisch. Beide Umgebungen entsprechen einer Absatzbox, wie die `minipage`. Sie erzeugen jeweils eine Tabelle aktuellen Position. Der optionale Positionsparameter ist, wie bei den Boxen für die vertikale Ausrichtung bezüglich der laufenden Zeile zuständig.

Das Argument für die Spaltenformatierung erhält einen Spaltentypparameter pro Tabellenspalte. Außerdem können hier Formatierungen, die für die gesamte Spalte gelten sollen positioniert werden.

`l`, `r`, `c`	links, rechts und zentriert
`p{`*Breite*`}`	Der Text dieser Spalte wird in eine Absatzbox gesetzt (Blocksatz, mit automatischen Zeilenumbruch). Die oberste Zeile ist dabei auf die laufende Tabellenzeile ausgerichtet.
`*{`*Anzahl*`}{`*Spaltenf.*`}`	Die Spaltenformatierung wird *Anzahl*-mal reproduziert. `*{3}{|c}|` ist somit dasselbe, wie `|c|c|c|`
`|`, `||`	Fügt eine bzw. zwei vertikale Linien zwischen den benachbarten Spalten ein.
`@{`…`}`	Dieser Ausdruck definiert den Zwischenraum zwischen Spalten oder vor der ersten bzw. nach der letzten Spalte und kann beispielsweise mit `\hspace` verändert werden. … steht dabei für einen beliebigen Textmodus-Befehl. Wird das Argument leer gelassen, so wird kein Zwischenraum eingefügt. Dies ist nützlich um überstehende Linien zu vermeiden.

Weitere sinnvolle Befehle innerhalb der Tabelle:

`&`	liefert einen Spaltenwechsel innerhalb der Tabellenumgebung
`\\[`*Abstand*`]`	beendet die gesamte Tabellenzeile
`\tabularnewline[`*Abstand*`]`	liefert unabhängig vom Spaltentyp einen Zeilenumbruch innerhalb der Tabelle
`\newline`	kommt innerhalb einer `p`, `m`, `b`-Spalte (siehe auch Abschnitt 11.1) für eine neue Zeile zum Einsatz
`\hline`	darf nur vor der ersten Zeile oder nach einem Zeilenumbruch stehen und erzeugt eine horizontale Linie der Breite der Tabelle. Entsprechend erzeugt `\hline\hline` eine Doppellinie.

`\cline{`*n*`-`*m*`}`	darf nur vor der ersten Zeile oder nach einem Zeilenumbruch stehen und erzeugt eine horizontale Linie vom linken Rand der *n*.ten bis zum rechten Rand der *m*.ten Spalte.
`\vline`	erzeugt einen vertikalen Strich über die Höhe der aktuellen Zelle
`\multicolumn{`*Spaltenanzahl*`}` `{`*Spaltenformatierung*`}{`*Text*`}`	erzeugt eine gemeinsame Spalte aus den folgenden Spalten, samt Zwischenräume.
`\multirow{`*Zeilenanzahl*`}` `{`*Breite*`}{`Text`}`	aus dem multirow-Paket fasst Zeilen zusammen und liefert eine bequeme Möglichkeit vertikal zu zentrieren. Als Breite kann einfach auch * angegeben werden, damit die Breite auf die tatsächliche Textbreite gesetzt wird.
`@{}`	Der Zwischenraum zwischen den Spalten wird mit diesem Spaltenformatierungsbefehl unterdrückt.

Tabellen mit fester Breite

```
\begin{tabular*}[Pos.]{Breite}{Spaltendefinition}
...
\end{tabular*}
```

Mit der Sternchenversion der Tabellenumgebung ist es möglich, der Tabelle eine feste Breite zu geben. Bei richtiger Verwendung, werden dann die Spaltenzwischenräume so weit gedehnt, dass die angegebene Breite erreicht wird. Dafür muss in der Spaltendefinition zwischen zwei Spalten

```
@{\extracolsep{\fill}}
```

stehen. Alle darauf folgenden Spaltenzwischenräume werden entsprechend gestreckt. Das Folgende Beispiel verdeutlicht dies durch die Linien in der zweiten Zeile, welche die Spaltenbreiten ausfüllen.

```
\begin{tabular*}{\linewidth}{|p{1.2cm}|
        p{4.65cm}|@{\extracolsep{\fill}}p{2cm}|p{2cm}|}
    ...
\end{tabular*}
```

A	B	C	D

11.1 Weitere Spaltenformatierungen – Das array-Paket

Dieses Paket erweitert die `tabular`- und `array`-Umgebung unter anderem um drei weitere Spaltenformatierungen:

`m{`*Breite*`}`	erzeugt wie `p` eine Spalte mir der angegebenen *Breite* mit vertikal zentrierter Ausrichtung
`b{`*Breite*`}`	analog `m`, wobei die Zellen am Boden ausgerichtet sind

```
\begin{tabularx}{\linewidth}{lXX} . . .
```

Hier etwas linksbündiger Text	Die erste X-Spalte. Wie man sieht, steht dieser Text im Blocksatz.	Die zweite X-Spalte ist genauso breit wie die erste
Im Gegensatz zu	der tabular*-Umgebung wird	hier die Spaltenbreite und nicht der Zwischenraum vergrößert.

Beispiel 7: Die tabularx-Umgebung

>{*Code*} <{*Code*}

setzt man vor (>{}) bzw. nach (<{}) den Spaltenformatierung und fügt den *Code* zu Beginn, bzw. am Ende jeder Zelle dieser Spalte ein

Besonders die letzte Erweiterung erspart viel Schreibarbeit, da man damit Formatierungne oder auch Teileinträge für die gesamte Spalte setzen kann. Im Beispiel wird eine Spalte mit blauem Text erzeugt:

```
\begin{tabular}{>{\color{blue}}c} ... \end{tabular}
```

Wenn man öfter Spalten mit gleicher Formatierung benötigt, dann ist es noch eleganter wenn man einen neuen Spaltentyp definiert:

\newcolumntype{*Name*}{>{*Code*}{*Spaltendefinition*}<{*Code*}}

Für obiges Beispiel, würde dies eine Spaltendefinition mit

```
\newcolumntype{B}{>{\color{blue}}c}
```

bedeutet. Der neue Spaltentyp B kann dann wie die Standardtypen benutzt werden.

Es wäre auch möglich \newcolumntype mit Argument zu nutzen:

```
\newcolumntype{V}[1]{>{\hspace{#1}}c}
```

Hier wird ein Argument Übergeben, dass den zusätzlichen Abstand zu Beginn der c-Spalte festlegt. In der Tabelle wird dieser Typ, wie die p-Spalte mit dem Argument in geschweiften Klammern benutzt (z. B. V{2cm}).

11.2 Variable Spaltenbreite – Das tabularx-Paket

Wird dieses Paket benutzt, so wird automatisch auch das array-Paket geladen (es stehen also auch alle Befehle des array-Pakets bereit, siehe Abschnitt 11.1). Deshalb muss das array-Paket nicht direkt geladen werden.

Mit dem tabularx ist es möglich, Tabellen mit einer bestimmten Breite zu erstellen. Im Gegensatz zu tabular* wird hier nicht der Spaltenzwischenraum, sondern die Spaltenbreite selbst variiert. Man muss also nicht die Breite der Spalten selbst festlegen, sondern gibt die Breite der Tabelle vor und alle X-Spalten werden entsprechend angepasst. Ist der Text innerhalb der flexiblen Spalte breiter als der verfügbare Platz, wird automatisch im Blocksatz umgebrochen. Werden mehrere X-Spalten verwendet, so wird der verfügbare Platz gleichmäßig aufgeteilt.

\begin{tabularx}{*Breite*}{*Spaltenformatierungen*} ... \end{tabularx}

```
\begin{tabulary}{\linewidth}{lCL} . . .
```

Hier etwas Text.	Die erste dehnbare Spalte. Wie man sieht, steht dieser Text zentriert.	Die zweite dehnbare-Spalte ist linksbündig und breiter wie die erste, da hier mehr Text steht.
Der Text	der einzelnen Zellen	schließen relativ bündig ab, zumindest bei den dehnbaren Spalten

Beispiel 8: Die `tabulary`-Umgebung

11.3 Variable Spaltenbreite mit Ausrichtung – Das tabulary-Paket

Das tabulary-Paket liefert ähnlich dem tabularx-Paket (siehe Abschnitt 11.2) eine Tabellenumgebung mit der man die Breite der Tabelle festlegen kann. Auch hier wird das array-Paket automatisch geladen.

Der Spalteninhalt wird automatisch an die Breite der Tabelle angepasst. Im Gegensatz zum `X`-Typ sind die Spalten von tabulary jedoch nicht dehnbar. Sie können nur gestaucht werden, wenn der Text einer Spalte breiter ist als der verfügbare Platz.

`R`	rechtsbündig
`C`	zentriert
`L`	linksbündig
`J`	Blocksatz („justified“)

Die Stärke dieses Pakets liegt in der Verwendung mehrerer flexibler Spalten. Der Platz wird hier nicht auf alle Spalten gleichmäßig aufgeteilt, sondern die tatsächliche Breite ist auch abhängig vom Spalteninhalt. Spalten mit mehr Inhalt bekommen auch mehr Platz zugewiesen.

`\begin{tabulary}{`*maximale Breite*`}{`*Spaltenformatierungen*`}` ... `\end{tabulary}`

11.4 Verbesserte Tabellenlayouts – Das booktabs-Paket

Der Standard-LaTeX-Befehl `\hline` für horizontale Linien erzeugt ungleiche Abstände. Zusätzlich sind die Abstände sehr knapp bemessen. Das booktabs-Paket verbessert diese Strukturen und ergänzt die Möglichkeiten für horizontale Linien:

`\toprule[`*Dicke*`]`
`\midrule[`*Dicke*`]`
`\bottomrule[`*Dicke*`]`

Die äußeren beiden Makros sind für Abgrenungslinien am Rand der Tabelle gedacht. Sie sind dicker als die Standard-Linien und haben nach oben (`\toprule`) bzw. unten (`\bottomrule`) mehr Abstand. Für Linien innerhalb der Tabelle benutzt man `\midrule`. Dies ist eine dünne Linie mit gleichem Abstand nach oben und unten.

Es gibt auch eine verbesserte Version der `\cline`:

`\cmidrule[`*Dicke*`](`*Zuschnitt*`){`*n*`-`*m*`}`

Optisch entspricht diese Linie eine verkürzten `\midrule`. Für den optionalen Zuschnittsparameter kann entweder `l`, `r`, `l{`*Länge*`}`, `r{`*Länge*`}` oder eine Kombination von diesen verwendet werden. Dies hat den Effekt, dass die Linie links und/oder rechts verkürzt wird. Wird die *Länge* nicht mit angegeben, so wird eine entsprechende Länge verwendet. Bei Standardeinstellungen ist diese Länge so groß, wie ein halber Spaltenzwischenraum, sodass die Linie nach dem Zuschnitt bündig zu Zelleninhalt abschließt.

Neben der Länge für den Zuschnitt ist es auch möglich die Dicken der Linien global anzupassen:

`\heavyrulewidth`	Dicke der `\toprule` & `\bottomrule`
`\lightrulewidth`	Dicke der `\midrule`
`\cmidrulewidth`	Dicke der `\cmidrule`

11.5 Mehrseitige Tabellen – Das longtable-Paket

Das longtable-Paket von David Carlisle bietet die Möglichkeit mehrseitige Tabellen zu setzen. Die Syntax entspricht weitestgehend der einer normalen `tabular`-Umgebung:

`\begin{longtable}[`*horizontale Pos.*`]{`*Spaltendefinitionen*`} ... \end{longtable}`

Für die optionale horizontale Positionierung können hierbei die Werte `l`, `c` oder `r` (`c` ist Standard) verwendet werden.

Ein Seitenumbruch ist nur nach Abschluss einer Tabellenzeile und nicht innerhalb einzelner Zellen möglich. Der Umbruch kann wie üblich auch duch `\\*` oder `\nopagebreak` verhindert beziehungsweise mit `\newpage` erzwungen werden.

Weitere Formatierungsmöglichkeiten für die `longtable` können der Paketdokumententation [3] entnommen werden.

12 Gleitobjekte – Positionierung von Bildern und Tabellen

Große Objekte, wie Tabellen oder Bilder, sollten in ihrer Position gleitend definiert werden, da sonst der Seitenumbruch nicht immer besonders platzsparend gelingt. LaTeX liefert hier dem Autor die Möglichkeit: Ist genug Platz für das Objekt vorhanden dann setze es bitte an den Ort seiner Erscheinung entsprechend dem Quellcode, ansonsten führe den Text weiter und positioniere das Objekt an geeigneter Stelle neu. Dies alles geschieht mit

```
\begin{table}[Position]
eigentliche Tabelle
\caption[optionaler Verzeichniseintrag]
  {Tabellenunterschrift}
\label{Marker}
\end{table}
```

```
\begin{figure}[Position]
eigentliche Abbildung
\caption[optionaler Verzeichniseintrag]
  {Bildunterschrift}
\label{Marker}
\end{figure}
```

Bei der Verwendung einer zweispaltigen Formatierung ist es sinnvoll einen `*` nach `table` oder `figure` einzufügen, damit wird -- im Gegensatz zur ungesternten Form — ein über 2 Spalten reichender Platz für das Objekt reserviert. Die optional gewünschte Position des Autors kann als

[1] Diese Fußnote hat keinen Textbezug. Sie dient lediglich der Veranschaulichung der Standard-Positionierung von Fußnoten im Zusammenhang mit Gleitobjekten, die unten auf der Seite positioniert werden (Positionsparameter b). Für eine genauere Beschreibung dieses Sachverhaltes siehe den entsprechenden Absatz auf Seite 100.

Abbildung 12.1: Jo, der persönliche Assistent und Haustier der Kursleiterin.

Propritätenliste aus `t` „top“, `b` „bottom“, `h` „here“ oder `p` „page“ (extra Seite) gesetzt werden. In der Stern-Variante sind `b` und `h` nicht möglich. Der erste Wert der Liste hat die höchste Priorität und wird wenn möglich verwirklicht. Der Standard ist `tbp`, kein `h`! Dies ist dadurch begründet, dass `h` eine Positionierung mitten auf der Seite bedeutet. Die daraus resultierende Spaltung des Satzspiegels ist typografisch fragwürdig.

Ein `!` vor der Positionsangaben verstärkt den Positionierungswunsch des Autors und versucht die gewünschte Position zu erzwingen. Um Warnungen vorzubeugen, sollte in jeder Liste, und wenn auch am letzten Punkt, der Parameter `p` stehen.

Mit dem `caption`-Befehl wird eine Bild- oder Tabellenbeschreibung eingefügt, die standardmäßig auch im Abbildungs- bzw. Tabellenverzeichnis steht (ein optionaler Text erlaubt dagegen andere Einträge in die entsprechenden Verzeichnisse). Das `\label`-Makro funktioniert vollkommen analog zu anderen Bezügen (Abschnitt 5.4.1). Ein einfaches Beispiel für ein Bild ist Abbildung 12.1. Oft möchte man jedoch die Bildbeschreibung neben dem Bild positionieren, hierfür gibt es die beiden Möglichkeiten in Abbildung 12.2 und Abschnitt 12.2.1.

Regeln nach denen Gleitobjekte positioniert werden
Bei der Positionierung der Gleitobjekte befolgt LaTeX strenge Regeln, in ihrem Rahmen lässt sich jedoch sagen, dass jedes Gleitobjekt so früh wie möglich ausgegeben wird:

- Gleitobjekte werden niemals auf einer früheren Seite ausgegeben als der, auf welcher sie definiert wurden.
- Gleitobjekte werden in der Reihenfolge ausgegeben, in der sie definiert wurden.
- Gleitobjekte werden nur an einer nach den Positionsparametern erlaubten Position ausgegeben.
- Wenn kein `!` unter den Positionsparametern ist, so hält sich LaTeX and die Vorgaben der Stilparameter. Einige davon sind in Tabelle 12.1 angegeben, eine komplette Liste findet sich in [13, S.171ff.].
- Im Fall der Kombination `ht` ist `h` von höherer Priorität. Das Gleitobjekt wird am Punkt der Definition eingefügt, auch wenn oben auf einer Seite genug Platz wäre.
- Gleitobjekte, die beim Auftreten eines `\clearpage`-Befehls noch nicht positioniert wurden, werden unabhängig von der Wahl der Positionsparameter auf einer eigenen Seite oder Spalte ausgegeben.

Möchte man noch weitere Positionseinschränkungen vornehmen, dann finden sich ein paar Tricks in Abschnitt 12.1 und Tabelle 12.1.

Fußnoten und Gleitobjekte
Erscheinen Fußnoten auf der selben Seite wie ein Gleitobjekt mit Positionsparameter `b` (Gleitobjekt unten auf der Seite), so werden die Fußnoten nicht als unterstes Objekt ausgegeben (Ein Beispiel für dieses Verhalten ist die Seite 99). Dies ist keinesfalls ein Fehler, vielmehr hat das Ganze typografische Ursachen. Ein Gleitobjekt enthält für gewöhnlich Abbildungen oder Tabellen und somit vergleichsweise wenig Text. Fußnoten bestehen dahingegen jedoch hauptsächlich auch Text. Somit gehören sie eher zum Textkörper als Gleitobjekte und werden auch näher zu diesem gesetzt.

Möchte man dieses Verhalten unterbinden, beispielsweise, weil die Gleitobjekte ebenfalls viel Text enthalten, so ist das mithilfe des footmisc-Paketes und seiner Option `bottom` möglich (Für weiterführende Informationen zu diesem Paket sei auf die Paketanleitung [20] verwiesen).

Abbildung 12.2: Noch einmal der Assistent Jo, nur diesmal in einer `figure`-Umgebung mit zwei `minipages`: eine mit dem Bild, die andere mit der `caption`, beide relativ zueinander vertikal zentriert. Darunter befindet sich der entsprechende Code.

```
\begin{figure}
    \begin{minipage}[c]{4cm}
        \includegraphics[width=\linewidth]{Bilder/jo}
    \end{minipage}
    \hfill
    \begin{minipage}[c]{.7\linewidth}
        \caption[Jo 2 . . . ]{Noch einmal . . . }\label{Jo2}
    \end{minipage}
\end{figure}
```

12.1 Floating einschränken

Die Positionierung von Gleitobjekten kann neben den Positionsparametern noch durch das Makro

`\suppresfloats[`*Position*`]`

eingeschränkt werden. Dieses verhindert das Auftreten von Gleitobjekten auf der aktuellen Seite, wobei es durch den optionalen Parameter *Position*, der entweder den Wert `t` oder `b` annehmen kann, möglich ist weiter zu spezifizieren. So unterdrückt die Angabe `t` ein Gleitobjekt mit `t` Positionierung, also nur oben auf der Seite.

Barrieren setzen – Das placeins-Paket
Mit

`\usepackage{placeins}`

kann der Autor mittels

`\FloatBarrier`

Barrieren setzen, die Gleitumgebungen nicht überwinden können. Sollen Gleitumgebungen pauschal nicht über eine `section` hinausgleiten, so erreicht man dies mit

`\usepackage[section]{placeins}`

Hier wird der `\section`-Befehl einfach um ein `\FloatBarrier` erweitert.

`\FloatBarrier` geht sehr strikt vor. Man kann dies soweit lockern, dass Gleitobjekte die Barriere insoweit noch überwinden können, damit sie noch auf der selben Seite eingebunden werden. Für das Überwinden nach „oben“ nimmt man die Paketoption `above`, nach unten `below`.

Tabelle 12.1: Tricks für die Positionierung von Gleitobjekten. Zähler können mithilfe der in Abschnitt 5.1 besprochenen Befehle manipuliert werden. Makros können über `\renewcommand` geändert werden (Abschnitt 5.3). Zulässig sind Werte < 1. Eine Zusammenfassung aller Gleitobjektparameter findet sich in [15, Abschnitt 6.1]

`topnumber`	Zähler (Standardwert: 2), der die maximale Anzahl von Gleitobjekten, die oben auf der Seite positioniert werden können angibt. (Bei zweispaltiger Formatierung und Verwendung der Sternchenversion der Gleitumgebungen heißt der Counter `dbltopnumber`)
`bottomnumber`	Wie `topnumber` für Positionierung unten auf der Seite. (Standardwert: 1)
`totalnumber`	Counter (Standardwert: 3), welcher die maximale Anzahl der Gleitobjekte pro Seite angibt.
`\topfraction`	maximaler Seitenbruchteil für `t`-Platzierungen. (Standardwert: 0.7)
`\bottomfraction`	maximaler Seitenbruchteil für `b`-Platzierungen. (Standardwert: 0.3)
`\textfraction`	minimaler Seitenbruchteil für Text (Standardwert: 0.2)
`\floatpagefraction`	Der Bruchteil einer Gleitseite, welcher von Gleitobjekten belegt sein muss (Standard 0.5). Dieser Bruchteil limitiert den Freiraum auf einer Gleitseite.

12.2 KOMA-spezielle `caption`-Einstellungen

12.2.1 Position der `caption`

LaTeX geht beim Befehl `caption` immer von einer *Unter*schrift der Gleitumgebungen aus. Große Tabellen sollten aber eine *Über*schrift aufweisen, damit der Leser zu Beginn weiß worum es sich auch in der nachfolgenden Tabelle handelt (eine Überschrift hat eine andere Formatierung wie eine Unterschrift, man beachte Abstand Tabelle und Über- bzw. Unterschrift).

`\documentclass[captions=`*Option*`]{scr…}`

Zu beachten ist, dass dies nur die Abstände korrigiert, die `caption` taucht immer nur da auf, wo sie auch definiert wurde. Soll bei einzelnen Tabellen gewechselt werden so geschieht dies mit

`\captionabove[`*Verzeichniseintrag*`]{`*Text*`}`	Gleitobjektüberschrift
`\captionbelow[`*Verzeichniseintrag*`]{`*Text*`}`	Gleitobjektunterschrift

anstelle von `caption`.

Für Beschreibungen neben dem Gleitobjekt kann man anstelle zweier Parboxen jetzt auch die folgende Umgebung verwenden

```
\begin{captionbeside}[Verzeichniseintrag]{Beschriftung}[Anordnung][Breite]
  [Offset]
…
\end{captionsbeside}
```

Diese Umgebung steht innerhalb der Gleitumgebung und umschließt selbst das Gleitobjekt. Als *Anordnung* gibt es `l`(links), `r`(rechts), `i`(innen) oder `o`(außen). Wird eine *Breite* angegeben, so wird die genutzte *Breite* bezüglich dem Textkörper zentriert. Ein positiver Wert von *Offset*

Tabelle 12.2: Werte für die Dokumentklassenoption captions

heading	Abstände von \caption immer wie \captionabove
signature	Abstände immer wie bei Bildunterschrift (Standard)
figureheading	Wie heading, jedoch nur für figure
figuresignature	Wie signature, jedoch nur für figure
tableheading	Wie heading, jedoch nur für table
tablesignature	Wie signature, jedoch nur für table
bottombeside	Titel für captionsbeside-Umgebung auf unterster Linie des Gleitobjekts
topbeside	Titel für captionsbeside-Umgebung auf oberster Linie des Gleitobjekts
centeredbeside	Titel für captionsbeside-Umgebung vertikal zentriert rel. zum Gleitobjekt
innerbeside	Titel für captionsbeside-Umgebung bei zweiseitigen Dokumenten innen
leftbeside	Titel für captionsbeside-Umgebung links von Gleitobjekt
rightbeside	Titel für captionsbeside-Umgebung rechts von Gleitobjekt
outerbeside	Titel für captionsbeside-Umgebung bei zweiseitigen Dokumenten außen

entspricht einer Verschiebung nach rechts. Wird hinter dem optionalen Parameter *Offset* noch ein Stern gesetzt, so stellt *Offset* im doppelseitigen Druck auf linken Seiten eine Verschiebung relativ zum rechten Rand dar. Ein *Offset* von 0 pt wäre somit bündig zum inneren Rand. In Tabelle 12.2 sind unter anderem Werte für die Dokumentklassenoption captions angegeben, die für die Umgebung captionbeside wichtig sind.

12.2.2 Formatierung der caption

Unterscheidung zwischen einzeilig und mehrzeilig

Mit dem Wert nooneline für die Dokumentklassenoption captions wird die automatische Zentrierung bei einzeiligen Über- bzw. Unterschriften abgeschaltet.

Schriftattribute

Man kann die Schrifteinstellungen des Gleitumgebungslabels („Abbildung“, „Tabelle“) und der Gleitumgebungsbeschreibung mit dem \setkomafont-Befehl bearbeiten, welcher in Abschnitt 4.1.5 bereits erklärt wurde.

Hängend oder eingerückt

Bei den KOMA-Klassen „hängt“ die Beschreibung nach Standardeinstellungen am Bezeichner. Wenn man davon abweichen möchte, so kann man dies mit:

```
\setcapindent{Einzug}
\setcapindent*{XEinzug}
```

Einzug ist ein Maß wie weit die zweite Zeile eingerückt wird. Ist der Wert negativ, so bekommt man nach dem Label einen Zeilenumbruch und nur die erste Zeile wird eingerückt. Soll ein Zeilenumbruch nach dem Label erfolgen und alle Zeilen um *XEinzug* eingerückt werden, so nimmt man die gesternte Form.

Soll auf den Standard zurückgeschaltet werden, so geht dies mit dem Makro

```
\setcaphanging
```

Beispiel 9 zeigt die verschiedenen Möglichkeiten zum Einrücken der Gleitobjektbeschriftungen.

Breite und Ränder

Die `Breite` der `caption` kann eingestellt werden über

`\setcapwidth[`*Ausrichtung*`]{`*Breite*`}`

wobei das optionale Argument die horizontale Ausrichtung festlegt. Man kann wählen zwischen `l` (linksbündig), `r` (rechtsbündig), `c` (zentriert), `i` (innen) und `o` (außen). Der Rand kann mithilfe von

`\setcapmargin[`*Rand links*`]{`*Rand rechts*`}`
`\setcapmargin*[`*Rand innen*`]{`*Rand außen*`}`

gesetzt werden. Soll der linke Rand vom rechten Rand abweichen so ist das optionale Argumente nötig. Für den doppelseitigen Druck gibt es die gesternte Variante.

Zusammensetzung von Label und Trenner

Das Label ist standardmäßig aufgebaut aus dem Namen des Gleitobjektes („Abbildung“, „Tabelle“) dem zugehörigen Zähler und eventuell einem Gliederungspunkt. Möchte man an diesem Aufbau was ändern oder hinzufügen so geht das mit `\renewcommand`. Standardmäßig ist das Label für Abbildungen so definiert: [2]

```
\newcommand*{\figureformat}{\figurename~\thefigure\autodot}
```

Bei Tabellen wird `\tableformat` anstatt `\figureformat` benutzt.

Das Trennzeichen ist normalerweise ein Doppelpunkt und ein Leerzeichen. Möchte man stattdessen einen Gedankenstrich einschließlich der notwendigen Leerzeichen, so ginge dies mit:

```
\renewcommand*{\captionformat}{~---~}
```

12.3 Objekte von Text umfließen lassen – Das wrapfig-Paket

Grafiken und Tabellen sollten im Allgemeinen vom Text abgehoben dargestellt werden. Eine Ausnahme hierbei sind besonders schmale Objekte. Hier kann es durchaus sinnvoll sein, diese vom Text umfließen zu lassen, um Platz einzusparen. Das Paket wrapfig von Donald Arseneau definiert zwei Umgebungen, die es ermöglichen ein Objekt vom Text *eines* Absatzes umfließen zu lassen

`\begin{wrapfigure}[`*Zeilen*`]{`*Position*`}[`*Randüberhang*`]{`*Breite*`}`...
`\begin{wraptable}[`*Zeilen*`]{`*Position*`}[`*Randüberhang*`]{`*Breite*`}`...

[2] `\autodot` setzt an seiner Stelle, das entsprechende Trennzeichen. Im Fall der `caption` das, welches in `captionformat` hinterlegt ist. Normalerweise handelt es sich dabei um einen Doppelpunkt gefolgt von einem Leerzeichen.

Standardversion der `caption`:

Gleitobjekt 1: Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcapindent{2cm}` – *Einzug* = 2 cm:

2 cm

Gleitobjekt 2: Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcapindent*{1cm}`– gesternte Version mit *XEinzug* = 1 cm:

1 cm

Gleitobjekt 3:
Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcapindent{-2cm}` – negativer Wert bei `\setcapindent`:

2 cm

Gleitobjekt 4:
Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcaphanging` – Zurückschalten zur Standardversion:

Gleitobjekt 5: Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

Beispiel 9: Verschiedene Möglichkeiten zur Ausrichtung der `\caption`

13 Verzeichnisse

13.1 Abbildungs- und Tabellenverzeichnis

Analog zum Inhaltsverzeichnis werden Abbildungs- und Tabellenverzeichnis mit den Makros

```
\listoffigures
\listoftables
```

erstellt. Die zugehörigen Hilfsdateien haben die Endung `.lof` (Abbildungen) und `.lot` (Tabellen). Ebenfalls wie beim Inhaltsverzeichnis wird somit wieder zweifaches Kompilieren benötigt.

13.2 Verzeichnisse manipulieren

Zusätzlich zu den automatischen Einträgen ist es möglich, manuelle Eintragungen in die Verzeichnisse zu erstellen:

`\addcontentsline{`*Verzeichniskürzel*`}{`*Gliederungstyp*`}{`*Verzeichniseintrag*`}`

Das *Verzeichniskürzel* ist in der Regel entweder `toc`, `lof` oder `lot`. Der *Gliederungstyp* ist z. B. `section` im Inhaltsverzeichnis, so dass der Eintragtext wie eine weitere `section` dargestellt wird. Als Seitenzahl wird, wie bei den Überschriften, die aktuelle automatisch eingetragen. Die Überschriften der einzelnen Verzeichnisse und auch die Ausgabe zu Beginn einer Bildunterschrift werden oft durch eingebundene sprachbezogene Ergänzungspakete bestimmt (babel). Sie können jedoch ebenfalls geändert werden:

`\renewcommand{`*Befehl der Überschrift/Bezeichnung*`}{`*neuer Verzeichnistitel*`}`

Die möglichen Befehle für Überschriften und Bezeichner lautet:

`\bibname`	„Literaturverzeichnis“ bei scrbook und scrreprt[1]
`\refname`	„Literatur“ bei scrartcl
`\listtablename`	„Tabellenverzeichnis“
`\listfigurename`	„Abbildungsverzeichnis“
`\tablename`	„Tabelle“
`\figurename`	„Abbildung“

Verzeichnisse entsprechen immer der höchsten hierarchischen Gliederungsebene (`\chapter` bei scrbook & scrreprt, sowie `\section` bei scrartcl). Dies ist insbesondere für den Satz von Vakatseiten (Dokumentenklassenoption `open=left/right`, siehe auch Tabelle 3.1, ab Seite 15) von Bedeutung, sowie für den Seitenstil der ersten und folgenden Inhaltsverzeichnisseiten (`\chapterpagestyle`, siehe Kapitel 7).

[1] mit biblatex-Paket nur „Literatur“

13.3 Literaturverzeichnis

13.3.1 Manuelle Erstellung des Literaturverzeichnisses

Ein Literaturverzeichnis kann manuell mit der Umgebung

```
\begin{thebibliography}{Mustermarke}
\bibitem[Marke]{Bezug1} Text1
\bibitem[Marke]{Bezug2} Text2
\bibitem[Marke]{Bezug3} Text3
...
\end{thebibliography}
```

erzeugt werden. Die Mustermarke ist dabei eine Reihe von beliebigen Zeichen, die LaTeX die Größe der Marke mitteilt. Sie sollte demnach mindestens so groß sein wie die längste `Marke`. Mit Hilfe des optionalen Parameters `Marke` kann eine beliebige Markierung für jeden einzelnen Literaturverweis erstellt werden (Standard: laufende Zahl in eckigen Klammern). Bezug ist ein zwingender Parameter bzw. ein Wort mit dem die Zuordnung stattfindet (darf keine Kommata enthalten). An der Stelle im Text an der man den Literaturverweis haben möchte schreibt man entsprechend

```
\cite[Informationen]{Bezug1,Bezug2,...}
```

```
Dieses Skript baut in Teilen auf dem Vorgängerskript von \cite{roedl}
    und dem Buch von Helmut Kopka \cite[3. Auflage]{kopka} auf.
```

Dieses Skript baut in Teilen auf dem Vorgängerskript von [4] und dem Buch von Helmut Kopka [12, 3. Auflage] auf.

Eine so erstellte Bibliographie ist allerdings an die Reihenfolge der Einträge im Literaturverzeichnis geknüpft. Die Einträge werden fortlaufend nummeriert. Mit den optionalen Informationen kann noch beliebiges weiteres Material wie Seitenzahl oder Kapitel zur Verfügung gestellt werden.

Oft werden beim Erstellen eines Dokuments entsprechende Literaturverweise ergänzt aber auch wieder entfernt. So stellt sich die Notwendigkeit nach einem Literaturverzeichnis, das nur die im Dokument verwendeten Einträge ins Ausgabefile übernimmt und auch die Nummerierung der Querverweise aktualisiert.

13.3.2 Automatisches Literaturverzeichnis mit Biber und dem biblatex-Paket

LaTeX bietet eine bequeme Möglichkeit die verwendeten Literaturzitate mit einer Literaturdatenbank zu synchronisieren, die der Autor LaTeX in einem `.bib`-File mitteilt. Hierfür kommt die TeX-Variante BibTeX zum Einsatz. Sie liest nur die Zitate aus der Literaturdatenbank aus, die im Dokument wirklich Verwendung finden und sortiert sie entsprechend. Diese Variante des Literaturverzeichnisses hat den Vorteil, dass man, zum Beispiel bei Google Books, direkt BibTeX-Dateien der betreffenden Bücher herunterladen kann. Außerdem unterstützt das Literaturverwaltungssystem „Citavi“, das an der Universität Regensburg über eine Campus-Lizenz zur Verfügung steht, den Export der *kompletten Citavi-internen Datenbank* in ein BibTeX-file.

Um mit Biber und dem Paket biblatex zu arbeiten legt man eine oder auch mehrere Dateien mit der Erweiterung `.bib` an. Diese enthalten die Literaturangaben. Zunächst werden jedoch die Befehle innerhalb des eigentlichen Dokumentes erläutert.

biblatex und biber

```
\usepackage[Optionen]{biblatex}
\bibliography{.bib-File}
```

Neben der Benutzung von Biber und biblatex steht auch noch das Programm BibTEX zur Verfügung. BibTEX ist zudem weiter verbreitet, weil es das ältere von beiden ist. Biber und das Paket biblatex haben jedoch wesentliche Vorteile, weswegen hier auch ihre Verwendung empfohlen wird. Zum einen ist Biber vollständig unicodefähig, kann also Umlaute und bestimmte Sonderzeichen verarbeiten, die bei BibTEX zu Problemen führen. Außerdem ist es mithilfe von Biber und biblatex sehr viel einfacher persönliche Anpassungen vorzunehmen als mit BibTEX.

Zusätzlich zum Paket selbst muss jedoch auch noch die Literaturdatenbank geladen werden, was durch Angabe des Dateipfades (falls sich die Datei im selben Ordner befindet genügt der Name) mit dem `\bibliography` geschieht. Die Dateiendung `.bib` muss dabei nicht angegeben werden. Allerdings ist zu beachten, dass für den Fall, dass es sich um mehrere Dateien handelt, die Namen durch Kommata, jedoch ohne nachfolgendes Leerzeichen getrennt werden.

Die Paketoptionen bestimmen die Formatierung der Zitate und legen den Stil des Literaturverzeichnisses fest.

```
\printbibliography
```

Dieser Befehl wird an der Stelle im Dokument positioniert, an der das Literaturverzeichnis erscheinen soll. Dort werden jedoch nur die Literaturangaben abgedruckt, auf die im Text entweder mit einem gedrucktem oder unterdrücktem Bezug verwiesen wird.

```
\cite[vorher][nachher]{Bezug}
\nocite{Bezug}
```

Das Makro `\cite` setzt dabei einen Bezug zum betreffenden Eintrag des Literaturverzeichnisses. *Vorher* und *nachhher* sind dabei Textbausteine, die z. B. noch zusätzlich auf einen bestimmten Teil der Literatur, also eine Seite oder ein Kapitel verweisen.

`\cite[vgl.][S. 345]{einfuehrung}`	[vgl. 23, S. 345]

```
\nocide{Verweisschlüssel}
```

setzt dahingegen einen stummen Bezug, also einen Bezug der gar nicht erscheint. Dies ermöglicht es Einträge im Literaturverzeichnis auszugeben, auf welche man gar nicht verwiesen hat. Möchte man alle Einträge ausgegeben haben, welche die `.bib`-Datei enthält, so funktioniert dies durch einen stummen Bezug auf das gesamte Verzeichnis:

```
\nocite{*}
```

Das Ausführen von Biber beginnt ganz normal mit dem Starten des LaTeX-Interpreters F6, damit die `.aux`- und `.bcf`-Dateien aktualisiert werden. Anschließend wird Biber F8 gestartet (erzeugt eine `.bbl`-Datei) und zum Schluss noch zweimal Kompilieren F6, damit neben den Literaturverweisen auch noch alle übrigen Verzeichnisse wieder auf dem aktuellen Stand sind. Je nach Editor und Konfiguration kann dies auch automatisch erfolgen.

.bib-Datei

In der `.bib`-Datei stehen alle Informationen über das zitierte Werk. Die Einträge einer solchen Datei sehen wie folgt aus:

```
@Typ{Bezug1,
    AUTHOR = {Autor1 and Autor2 and Autor3...},
    TITLE = {Titel},
    JOURNAL = {Journal oder Verlag},
    YEAR = {Erscheinungsjahr},
    VOLUME = {Auflage},
    NUMBER = {issue bei papers},
    PAGES = {entsprechender Ausschnitt vonbis},
    MONTH = {Erscheinungsmonat}
}
@Typ{Bezug2,
    AUTHOR = {...
```

Die Angabe der Einträge ist dabei nicht zwingend, jedoch erwartet LaTeX je nach *Typ* bestimmte Angaben. Die wichtigsten Typen samt der erwarteten Angaben finden sich in Tabelle 13.1, eine vollständige Liste gibt es in der Paketdokumentation von biblatex [18]. Die Groß- und Kleinschreibung spielt bei den Bezeichnern „AUTHOR“, „TITLE“ usw. keine Rolle. Als Beispiel dient wieder [23].

```
@book{einfuehrung,
    author = {Voß, Herbert},
    year = {2012},
    title = {Einführung in \LaTeX2e: Unter Berücksichtigung von pdf\LaTeX,
        Xe\LaTeX und Lua\LaTeX},
    keywords = {Empfehlung},
    edition = {1},
    publisher = {Lehmanns Media},
    isbn = {978-3-86541-462-5}
}
```

Stil- und Sortieroptionen

Man kann in den Paketoptionen zudem einen Stil festlegen. Diesen setzt man entweder sowohl für die Zitate als auch für das Verzeichnis, oder für beide einzeln.

`style=`*Stil*	Stil für Zitate und Literaturverzeichnis
`citestyle=`*Stil*	Stil für Zitate
`bibstyle=`*Stil*	Stil für Literaturverzeichnis

Die Voreinstellung ist `numeric`. Es gibt jedoch auch Stile für Zitate unter Nennung von Autor und Jahr oder speziell für Fußnoten. Die biblatex-Anleitung [18] zeigt die Standard-Stile, jedoch gibt es auch darüber hinaus noch viele sehr spezielle Varianten.

Eine Liste der Standard-Stile findet sich in der biblatex-Dokumentation [18]. Darüber hinaus kann man über das `texdoc`-Programm oder CTAN `https://www.ctan.org/topic/biblatex` noch zusätzliche Stile und Ergänzungen finden.

Viele Stile unterstützen unterschiedliche Varianten, die alle in der Dokumentation gut verständlich erklärt sind. Hier soll nur die `-comp` Variante vorgestellt werden:

```
\usepackage[style=numeric-comp]{biblatex}
```

Tabelle 13.1: Eintragstypen für die Literaturdatenbank. Nicht genannte Felder werden nicht verarbeitet.

Typ	*Empfohlen/Notwendig*	*Optional*
article	author, title, journaltitle, year/date	translator, annotator, commentator, subtitle, titleaddon, editor, editora, editorb, editorc, journalsubtitle, issuetitle, issuesubtitle, language, origlanguage, series, volume, number, eid, issue, month, pages, version, note, issn, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
book	author, title, year/date	editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
online	author/editor, title, year/date, url	subtitle, titleaddon, language, version, note, organization, date, month, year, addendum, pubstate, urldate
thesis	author, title, type, institution, year/date	subtitle, titleaddon, language, note, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
unpublished	author, title, year/date	subtitle, titleaddon, language, howpublished, note, location, isbn, date, month, year, addendum, pubstate, url, urldate
Falls der Typ nicht zuzuordnen ist:		
mics	author/editor, title, year/date	subtitle, titleaddon, language, howpublished, type, version, note, organization, location, date, month, year, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
Es gibt auch Typen, die die Dokumentenhierarchie erhalten. Typen, die mit in beginnen, beziehen sich somit auf ein größeres Dokument		
proceedings	title, year/date	editor, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, eventtitle, eventtitleaddon, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
inproceedings	author, title, booktitle, year/date	editor, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, eventtitle, eventtitleaddon, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

Tabelle 13.2: Sortieroptionen für das Literaturverzeichnis mit biblatex. Die Sortierschemata beziehen sich dabei auf die Reihenfolge der Einträge im Literaturverzeichnis. Diese Optionen bestimmen nicht die Abfolge der Daten innerhalb der Einträge.

`none`	nicht sortiert, Auflistung nach Zitierreihenfolge
`nty`	Name, Titel, Jahr
`nyt`	Name, Jahr, Titel
`nyvt`	Name, Jahr, Volume, Titel
`anyt`	alphabetischem Label, Name Jahr, Titel
`anyvt`	alphabetischem Label, Name Jahr, Volume, Titel
`ynt`	Jahr, Name, Titel
`ydnt`	absteigendem Jahr, Name, Titel

Hier bekommt man wegen dem Style `numeric` eine Zahl in eckigen Klammern. Die zusätzliche (optionale) Variante `-comp` sorgt dafür dass bei mehreren Angaben die Zahlen sortiert und zusammengefasst werden. Statt [8,3,1,7,2] erhält man also [1–3,7,8].

Für die Sortierung der Einträge im Literaturverzeichnis gibt es vordefinierte Schemata, zu finden in Tabelle 13.2, man kann sich aber auch sein eigenes Schema basteln. Hier sei wieder auf die Paketdokumentation verwiesen.

`sorting = `*Option*

In Tabelle 13.2 findet man die möglichen *Sortieroptionen*.

Zitierbefehle

Mit dem biblatex-Paket gibt es mehrere Zitierbefehle, welche immer die Struktur

`\`*befehl*`[`*vorher*`][`*nachher*`]{`*Bezug*`}`

haben. *Vorher* steht immer vor dem Zitat (z. B. *siehe*), *nachher* immer nach dem Zitat (oft Seitenangabe). Bei nur einer Eingabe wird diese als *nachher* behandelt. Möchte man nur *vorher*, so muss man ein leeres zweites Argument benutzen. Mögliche Zitierbefehle siehe Tabelle 13.3.

Man kann auch auf mehere Werke auf einmal verweisen:

`\cites[`*vorher*`][`*nachher*`]{`*Bezug*`}[`*vorher*`][`*nachher*`]{`*Bezug*`}...`

Weitere nützliche Paketoptionen findet man in Tabelle 13.4.

Man kann auch auf mehrere Datenbanken verweisen:

`\addbibresource{`*Dateiname*`.bib}`

```
\addbibresource[location=remote]{http://www.bibtex.org/bib/6}
\addbibresource[location=remote,label=lan]{ftp://bib/file.bib}
```

Wie man kleine Formatierungen der Zitate und der Einträge im Literaturverzeichnis vornimmt soll anhand eines kleinen Beispiels gezeigt werden:

Tabelle 13.3: Zitierbefehle für das Paket biblatex

`\cite[`*vorher*`][`*nachher*`]{`*Bezug*`}`	Standard Zitierbefehl zitiert nach dem gesetzten Zitierstil
`\parencite[`*vorher*`][`*nachher*`]{`*Bezug*`}`	umklammertes Zitat (rund bzw. eckig)
`\footcite[`*vorher*`][`*nachher*`]{`*Bezug*`}`	Erzeugt Literaturangabe und Fußnote
`\autocite[`*vorher*`][`*nachher*`]{`*Bezug*`}`	je nach Zitierstil unterschiedliche Ausgabe, siehe Paketoption autocite
`\textcite[`*vorher*`][`*nachher*`]{`*Bezug*`}`	in allen nicht-verbose Stilen verfügbar, gibt Autoren gefolgt vom Zitierschlüssel aus
`\supercite{`*Bezug*`}`	in numerischen Stilen verfügbar, ergibt hochgestellte Zitatnummer ohne Klammern
`\nocite{`*Bezug*`/*}`	fügt `key`/alle Einträge der Datenbank zu Bibliographie ohne Zitat hinzu
`\fullcite[`*vorher*`][`*nachher*`]{`*Bezug*`}`	fügt kompletten Literaturverweis wie aus Bibliographie ein
`\citeauthor[`*vorher*`][`*nachher*`]{`*Bezug*`}`	Ausgabe Autoren, Editoren oder Übersetzer
`\citetitle[`*vorher*`][`*nachher*`]{`*Bezug*`}`	Ausgabe wenn vorhanden `shorttitle` ansonsten `title`
`\citeyear[`*vorher*`][`*nachher*`]{`*Bezug*`}`	Ausgabe Erscheinungsjahr
`\citedate[`*vorher*`][`*nachher*`]{`*Bezug*`}`	Ausgabe volles Datum
`\citeurl[`*vorher*`][`*nachher*`]{`*Bezug*`}`	Ausgabe url-Feld

Tabelle 13.4: Weitere Paketoptionen für das Paket biblatex

`autocite`	`plain`	verhält sich wie `\cite`
	`inline`	verhält sich wie `\parencite`
	`footnote`	verhält sich wie `\footcite`
	`superscript`	verhält sich wie `\supercite`
		Standardwert hängt vom Zitierstil ab
`backref`	`true/false`	fügt Seitenzahlen der Zitate an die Bibliographieeinträge an
`backend`	`biber`	unterstützt von ASCII bis UTF-8, on-the-fly rencoding, Sortierungsregeln über sortlocale, sortcase, sortupper einstellbar
	`bibtex`	traditionelles BibTeX, unterstützt nur ASCII, Sortieren ist immer abhängig von Groß/Kleinschreibung
`heading`	`label`	eigene Bibliographieüberschriften einsetzen, definieren mit `\defbibheading{label}{code}`

```
\usepackage[style=authortitle]{biblatex}
\DeclareFieldFormat[book]{title}{$\clubsuit$ #1 $\spadesuit$}
\DeclareFieldFormat[book]{citetitle}{$\ast$ \textsc{#1} $\ast$}
\DeclareFieldFormat[article]{citetitle}{[ #1 ]}
\renewcommand*{\multinamedelim}{ + }
\renewcommand*{\finalnamedelim}{ und manchmal }
\begin{document}
\cite{companion} \cite{murray} \textcite{companion}
\printbibliography
```

Hier werden die Einträge im Literaturverzeichnis primär nach Autorennamen sortiert. Vor jedem Buchtitel steht das `clubsuit`-Symbol, danach immer ein `spadesuit`-Symbol. Bei den Zitaten im Text werden die Buchtitel mit Sternchen umschlossen. Die Titel der Artikel stehen in den Zitaten in eckigen Klammern. Gibt es mehrere Autoren werden sie mit einem Pluszeichen getrennt, außer vor dem letzten wo hier „und manchmal“ steht.

Eine komplette Liste der formatierbaren Attribute findet man in der Paketdokumentation.

13.4 Index

Die Indexerstellung bei LaTeX funktioniert analog zum Literaturverzeichnis. Es gibt eine manuelle Variante, die jedoch sehr aufwendig ist, und die automatische Indexerstellung über das Zusatzprogramm `makeindex`.

`\makeindex`	Nur einmal in der Präambel ausführbar
`\index{`*Argument*`}`	

Das Makro `\makeindex` aktiviert die Indexerstellung und schreibt die zugehörigen Befehle in die `*.idx`-Datei. Das `\index`-Makro wird überall da positioniert, wohin die Einträge verweisen sollen.

Zum Ausdrucken und zur Formatierung des Index benutzt man üblicherweise das Paket makeidx. Dies liefert das Makro

`\printindex`

und ermöglicht so eine einfache Ausgabe, siehe Beispiel 10.

Analog zur Bibliografie muss hierbei zunächst einmal mit `pdflatex` F6 kompiliert werden, anschließend lässt man `makeindex` (Hierfür gibt es keine F-Taste mehr. Das Programm muss über Tools→Befehle→MakeIndex gestartet werden. Bei alten TeXstudio-Versionen ist hierfür die Taste F11 vorgesehen.) laufen, bevor wiederum zwei `pdflatex`-Läufe nötig sind.

Für Querverweise im Index, z. B. „Index, siehe Verzeichnis“ benötigt das Paket makeidx interne Makros, die auch von babel entsprechend übersetzt werden. Zusätzlich können sie mit `\renewcommand` verändert werden.

`\seename`	Standard: „siehe“
`\alsoname`	Standard: „siehe auch“

Die Verwendung in den Indexeinträgen selbst hat eine etwas gewöhnungsbedürftige Syntax, siehe Tabelle 13.5 [vgl. 23, S. 512].

Im Zusammenhang mit den `\index`-Makro stellen die folgenden Makros Beispiele für die Verwendung der Syntax da. Die Zahlenwerte neben den Makros beziehen sich auf die Nummerierung der Einträge in Beispiel 11.

```
\usepackage{makeidx}
\makeindex
    . . .
\begin{document}
Ein Index\index{Index} wird erstellt.
Die Datei\index{Datei} sammelt die
Einträge\index{Eintrag}\index{sammeln}.
    . . .
\printindex
    . . .
\end{document}
```

Index

Datei, 1

Eintrag, 1

Index, 1

sammeln, 1

Beispiel 10: Einfache Indexerstellung mit makeidx. In Anlehnung an [23, Beispiel 10-7-02]

Tabelle 13.5: Syntaxerläuterungen für den Satz von Indexeinträgen mit `makeindex`.

Syntax	*Bedeutung*
`!`	Es folgt ein Unterverzeichniseintrag, der dem vorangehenden untergeordnet wird. Im Index wird er entsprechend eingerückt. Es sind maximal 2 Ebenen an Untereinträgen möglich.
`@`	Die Angabe vor `@` entspricht dem Sortierkriterium, der Eintrag dahinter stellt den tatsächlichen Indexeintrag dar.
`|`	Das Encap-Zeichen interpretiert den folgenden Namen als Makro, z. B. `|bfseries` oder `|see{`*Eintrag*`}`.
`|(`	Die Seitenzahlen des Indexeintrages werden bis zu einem Eintrag mit `|)` zusammengefasst.
`|)`	Gegenstück zu `|(`

```
\usepackage{makeidx}\makeindex

\printindex          \newpage
1) \index{Index}     \newpage
2) \index{Eintrag|(}\newpage
3) \index{idx-Datei@\texttt{idx}-Datei}
4) \index{Zirkus!Roncalli}   \newpage
5) \index{Manege|see{Zirkus}}\newpage
6) \index{Eintrag}\index{ABC-Schütze}
7) \index{Eintrag|)}
8) \index{Show|seealso{Zirkus}}
9) \index{Gemüse!Erbsen}     \newpage
10)\index{Gemüse!Bohnen}
11)\index{shutdown@\texttt{shutdown}}
12)\index{Wichtig@\textbf{Wichtig}}
13)\index{Mathematik}
14)\index{Clown!Pic}\newpage
15)\index{alpha@$\alpha$}
16)\index{$\beta$}  \newpage
17)\index{123}      \newpage
18)\index{\&}       \newpage
19)\index{Zirkus!Pic@\textsc{Pic}|see{Clown}}
```

Index

β, 7
&, 9

123, 8

ABC-Schütze, 5
α, 7

Clown
  Pic, 6

Eintrag, 2–5

Gemüse
  Bohnen, 6
  Erbsen, 5

idx-Datei, 3
Index, 1
Manege, *siehe* Zirkus
Mathematik, 6

Show, *siehe auch* Zirkus
shutdown, 6

Wichtig, 6

Zirkus
  Pic, *siehe* Clown
  Roncalli, 3

Beispiel 11: Spezielle Syntax für Syntaxeinträge

\index{*Eintrag*}	Einfacher Indexeintrag
\index{*Haupteintrag*!*Untereintrag*}	siehe 1)
\index{*Eintrag*|(}	Untereintrag
\index{*Eintrag*|)}	siehe 9)
\index{*Sortiereintrag*@*Eintrag*}	10)
\index{*Eintrag*|see{*Eintrag*}}	…
\index{*Eintrag*|seealso{*Eintrag*}}	Anfang eines „von–bis“-Verweises

14 Formeln und Einheiten

LaTeX wurde ursprünglich entwickelt um unter anderem einen ordentlichen Formelsatz zu ermöglichen. Somit ist es selbstverständlich, dass es eine riesige Menge an Anpassungsmöglichkeiten für Formeln gilt.

14.1 Typografische Regeln

Variablen oder physikalische Größen werden durch einzelne lateinische oder griechische Buchstaben dargestellt. Nach internationalen Konventionen werden diese grundsätzlich kursiv gesetzt:

- Einfache Variablen x, y, z
- Mathematische Funktionen $z = f(x, y)$, $\psi(t) = \int_{t_0}^{t} \psi \, \mathrm{d}t$
- Physikalische Konstanten ϵ_0, μ_0
- Indizes, die Variablen oder physikalischen Konstanten entsprechen $a_{i,j}$, c_v

Die folgenden Größen werden jedoch aufrecht gesetzt:

- Alle Ziffern 123, α_{25}
- Mathematische Operatoren $\mathbf{A}^T = \mathbf{B}$
- Mathematische Funktionen, die einem bestimmten Typ angehören $a = \arccos(x)$, $\Gamma(x)$
- Einheiten samt Vorsatz $\lambda = 0{,}4\,\mu\mathrm{m}$, $12\,\mathrm{kg}$
- Indizes die zur Kennzeichnung nach etwas benannt sind $x_{\max}$, μ_{B}
- Chemische Summenformeln $\mathrm{H_2O}$

Physikalische Größen bestehen immer aus Zahl und Einheit, wobei

- Zahl und Einheit werden durch ein halbes geschütztes Leerzeichen (`\,` oder `\thinspace`) voneinander getrennt: `15\,km` (15 km)
- lediglich bei Winkelangaben in Grad entfällt dieser Abstand (nicht jedoch bei Temperaturangaben in Celsius) $18^\circ 21' 4''$
- Einheiten können ausgeklammert werden: $P = 50\,\mathrm{kW} \pm 4\,\mathrm{kW} = (50 \pm 4)\,\mathrm{kW}$

Außerdem ist zu beachten:

- Prozentangaben werden wie Einheiten abgesetzt: `13\,\%` (13 %)
- Zahlenkolonnen können zur besseren Lesbarkeit von rechts an in Dreiergruppen unterteilt werden. Dies geschieht ebenfalls durch ein halbes geschütztes Leerzeichen (`\,`): 1 234 567 890 statt 1234567890
- Da LaTeX amerikanischen Standards unterliegt, ist ein Punkt als Dezimaltrennzeichen definiert. Schreibt man anstelle des Punktes einfach ein Komma stimmen jedoch die Abstände nicht mehr (1, 11 statt 1.11). Dies kann man korrigieren indem man das Komma einklammert `$1{,}11$` (1,11) oder Komma und Punkt in der Präambel entsprechend umdeklariert:

```
\DeclareMathSymbol{,}{\mathord}{letters}{"3B}
\DeclareMathSymbol{.}{\mathpunct}{letters}{"3A}
```

- Der Buchstabe „d“ bei Differentialoperatoren und Integralen ist in deutschsprachigen Texten aufrecht zu setzen, zudem wird er bei Integralen generell durch einen Abstand (`\,`) vom Integranden getrennt:

$$\frac{\mathrm{d}}{\mathrm{d}x}f(x) \text{ anstatt } \frac{d}{dx}f(x); \qquad \int_0^y f(x)\,\mathrm{d}x \text{ anstatt } \int_0^y f(x)\mathrm{d}x$$

- Der „Doppelpunkt“ bei der Definition von Funktionen ist für richtige Abstände mit dem Makro `\colon` zu setzen: $f : D \longrightarrow Z$ anstatt $f : D \longrightarrow Z$
- Folgt auf den Doppelpunkt ein Gleichheitszeichen, so kann man das mit dem Symbol `:=` (`\coloneqq`) aus dem mathtools-Paket setzen. Bei der Zeichenfolge `:=` ist der Doppelpunkt nicht ganz zur Achse des Gleichheitszeichens zentriert. Benutzt man das mathtools-Paket, kann man allerdings auch einfach den Doppelpunkt durch das Makro `\mathtoolsset{centercolon}` generell zentrieren lassen, um dasselbe Ergebnis mit der Kurzschreibweise zu erreichen.

Allgemeine Anmerkung:
Viele Zeichen existieren sowohl im Text als auch im Mathemodus. Hier sollte jedoch die nach Inhalt richtige Variante gewählt werden (vgl. Hinweise zum Minuszeichen in Abschnitt 3.3.4). Es sind kleine Unterschiede mit großer Wirkung:

```
Falsch: a-b+2*c=D \hfill$\leftrightarrow$ Richtig: $a-b+2\cdot c=D$
```

Falsch: a-b+2*c=D ↔ Richtig: $a - b + 2 \cdot c = D$

14.2 Schriftattribute

Standardmäßig geht LaTeX davon aus, dass es sich bei Buchstaben um Variablen handelt, sie werden somit automatisch kursiv gesetzt. Zahlen erscheinen jedoch aufrecht. Um die Darstellung der Schrift zu verändern gibt es folgende Befehle:

`\mathrm{Zeichenfolge}`	Roman ABCabc
`\mathtt{Zeichenfolge}`	Typewriter `ABCabc`
`\mathsf{Zeichenfolge}`	Sans Serif ABCabc
`\mathit{Zeichenfolge}`	kursiv $ABCabc$
`\mathcal{Zeichenfolge}`	Kaligraphie $\mathcal{ABC}$
`\mathnormal{Zeichenfolge}`	Standardschrift
`\mathbf{Zeichenfolge}`	fett außer griechische Kleinbuchstaben und Symbole
`\mathbb{Zeichenfolge}`	benötigt amsfonts-Paket, Mengensymbole $\mathbb{N}, \mathbb{C}$
`\mathfrak{Zeichenfolge}`	benötigt amsfonts; Frakturschrift $\mathfrak{ABCabc}$
`\boldsymbol{Zeichenfolge}`	benötigt amsmath; setzt auch griechische Buchstaben und Symbole fett, erzeugt jedoch unschöne Abstände.
`\boldmath`	Schalterbefehl. Muss vor der Matheumgebung angewendet werden; (fast) alle Zeichen fett

```
\newcommand*{\MStyleExample}{
    \mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}}
    \int_{-\infty}^{+\infty}f(t)\mathrm{e}^{-\mathrm{i}\omega t}
    \,\mathrm{d}t
}
```

```
$\displaystyle\MStyleExample$\\
$\textstyle\MStyleExample$\\
$\scriptstyle\MStyleExample$\\
$\scriptscriptstyle\MStyleExample$
```

$$\mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$
$$\mathrm{FT}(f) = \tfrac{1}{\sqrt{2\pi}} \textstyle\int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$
$$\mathrm{FT}(f) = \tfrac{1}{\sqrt{2\pi}} \textstyle\int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$
$$\mathrm{FT}(f) = \tfrac{1}{\sqrt{2\pi}} \textstyle\int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$

Beispiel 12: Mathematische Schriftstile im Vergleich.
`\MStyleExample` dient als Abkürzung für den Beispielcode.

`\unboldmath`	Ausschalten von `\boldmath`
`\bm{Zeichenfolge}`	benötigt das bm-Paket; setzt mathematische Ausdrücke fett (beeinflusst jedoch nicht wie `\boldsymbol` die Abstände)

14.3 Schriftstile

Neben den Schriftattributen kennt LaTeX vier mathematische Schriftstile, siehe Beispiel 12. Man nennt dies Befehle bewusst „styles“ und nicht „Größen“, da sich auch dir Form der Zeichen unterscheidet. Beispielsweise ist in Textformeln die Größe von Zähler und Nenner kleiner als wenn sie nicht auf bzw. unter dem Bruchstrich stünden, da dies für den Zeilenabstand günstiger ist. Den Unterschied zwischen Größe und Style sieht man auch an den Beispielen, da die Symbolgrößen nicht gleichstark skaliert werden.

14.4 Zeilenmodus vs. abgesetzter Modus

Bei LaTeX wird zwischen zwei grundlegend verschiedenen Arten des Formelsatzes unterschieden: Dem mathematischen Zeilenmodus (inline math mode) und dem abgesetzten Modus.

Der Zeilenmodus wird dazu verwendet mathematische Elemente in laufende Zeilen, wie zum Beispiel $\mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$, zu setzen. Theoretisch gibt es keinerlei Beschränkung von Größe und Inhalt, allerdings werden die beiden äußeren Zeilen so weit auseinandergezogen, dass der Inhalt dazwischen passt. Dies kann dazu führen, dass ein sehr unschönes Layout entsteht. Ordnet man zum Beispiel eine Matrix in einer Zeile an lässt sich dies eigentlich nicht vermeiden: $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ Allerdings gibt es Befehle die die Umgebungen kleiner erscheinen lassen und somit das Schriftbild weniger stören. Für den Fall der Matrix bietet sich die `smallmatrix`-Umgebung aus dem amsmath-Paket an $A = \left(\begin{smallmatrix} a & b & c \\ d & e & f \\ g & h & i \end{smallmatrix}\right)$. Jedoch empfiehlt es sich solche Formeln eher im abgesetzten Modus (Abschnitt 14.4.2) zu setzen um das Layout nicht zu stören.

14.4.1 Der Zeilenmodus

Der Zeilenmodus kann durch drei verschiedene Umgebungen aktiviert werden:

`$`*Formelcode*`$`	Standard TEX-Syntax für den Mathemodus
`\(`*Formelcode*`\)`	LATEX-Variante des Mathe-Modus. Fehlermeldungen sind hiermit leichter nachvollziehbar
`\begin{math}`*Formelcode*`\end{math}`	Text-Makro-Variante der LATEX-Version

Die Dollarzeichen sind die weiter verbreitete Variante, da sie die ältere ist. Darüber hinaus war es bis vor einigen Jahren nicht möglich die LATEX-Variante innerhalb von Verzeichnissen oder Überschriften zu verwenden. Mittlerweile ist es für Anfänger häufig sinnvoller die LATEX-Version zu nutzen, da die Fehlermeldungen in diesem Fall leichter verständlich sind.

Neben dem manuellen Mathemodus gibt es noch ein Makro um den Mathemodus zu erzwingen:

`\ensuremath{`*Code*`}`

Dieses Makro testet zunächst, ob man sich gerade im Mathemodus befindet. Ist dies der Fall wird das Argument eins zu eins übernommen. Sollte das jedoch nicht zutreffen, wird vorher in den Mathemodus und anschließend wieder zurück gewechselt.

`\ensuremath` wird vor allem innerhalb von Makrodefinitionen benötigt, die in beiden Modi verwendbar sein sollen, wie zum Beispiel `\SI` (vgl. Abschnitt 14.17).

14.4.2 Der abgesetzte Modus

Einzeilige Umgebungen:

`\[`*Formelcode*`\]`	nicht nummeriert
`\begin{displaymath}`*Formelcode*`\end{displaymath}`	wie `\[...\]`
`\begin{equation}`*Formelcode*`\end{equation}`	(rechtsbündig) nummeriert (siehe auch `leqno` in Tabelle 3.1)
`\begin{equation*}`*Formelcode*`\end{equation*}`	nur mit amsmath-Paket verfügbar; nicht nummeriert

Zusätzlich existiert noch eine für TEX gültige Makrokombination `$$..$$`*. Dies Syntax ist jedoch veraltet und aus verschiedensten Gründen zu vermeiden!*

Mehrzeilige Umgebungen:
Standard-LATEX liefert lediglich die `eqnarray`-Umgebung. Sie sollte jedoch aufgrund erheblicher Mängel nicht mehr verwendet werden. Mehrzeilige Umgebungen erfordern somit das amsmath-Paket.

```
\begin{align}Formelcode\end{align}
\begin{align*}Formelcode\end{align*}
\begin{flalign}Formelcode\end{flalign}
\begin{flalign*}Formelcode\end{flalign*}
\begin{alignat}{Anzahl der Blöcke}Formelcode\end{alignat}
\begin{alignat*}{Anzahl der Blöcke}Formelcode\end{alignat*}
```

Es gibt somit drei verschiedene Arten von `align`-Umgebungen. Die Sternchenformen unterdrücken jeweils wie üblich die Nummerierung. Will man die Nummer lediglich für einzelne Zeilen unterdrücken, so setzt man vor den Zeilenumbruch den Befehl

```
\[
\left.
\begin{aligned}
2x+3 &=7&       2x+5-5&=7-3\\
2x   &=4& \frac{2x}{2}&=2\\
x    &=2
\end{aligned}
\right\}
\]
```

$$\left.\begin{aligned} 2x+3 &= 7 & 2x+5-5 &= 7-3 \\ 2x &= 4 & \frac{2x}{2} &= 2 \\ x &= 2 \end{aligned}\right\}$$

Beispiel 13: Die aligned-Umgebung. Auf die Struktur der Klammern sowie der Makros \left und \right wird in Abschnitt 14.12 genau eingegangen.

`\nonumber`

Er unterdrückt schlicht und einfach die Nummerierung der aktuellen Formelzeile.

Die Darstellung der align-Umgebungen entspricht folgendem Prinzip:

align	=	x		x	=	x	
align	=	x		x	=	x	
alignat mit 2 Blöcken	=	x	x	=	x		
alignat	=	x	x	=	x		
flalign	=	x		x	=	x	
flalign	=	x		x	=	x	

Die Syntax funktioniert ähnlich zu Tabellen, wobei das Argument bei alignat die Anzahl der Gleichungsblöcke benötigt.

```
\begin{align}
   align&=x&x&=x\\
   align&=x&x&=x
\end{align}
```

Das &-Zeichen sollte grundsätzlich *vor* dem Gleichheitszeichen, bzw. dem Relationssymbol gesetzt werden. Somit erreicht man, dass die Ordnungssymbole alle direkt untereinander gesetzt werden. Prinzipiell haben alle align-Umgebungen eine Spaltenstruktur von Rechts-Links-Anordnungen ({rlrlrl...}), und unterscheiden sich lediglich durch die Positionierung (s. o.).

`\begin{aligned}`*Formelcode*`\end{aligned}`

Die aligned-Umgebung ermöglicht prinzipiell dasselbe wie die align-Umgebung. Allerdings kann sie geschachtelt werden und somit innerhalb einer beliebigen anderen Matheumgebung angewendet werden. Man kann mit ihrer Hilfe erreichen, dass zum Beispiel drei Gleichungen nur eine einzelne, gemeinsame Nummer erhalten, siehe Beispiel 13.

`\begin{gather}`*Formelcode*`\end{gather}`
`\begin{gather*}`*Formelcode*`\end{gather*}`

```
\(|a| =
\begin{cases}
    \phantom{-}a;&a\geq 0\\
    -a;&a<0
\end{cases}
\)
```

$$|a| = \begin{cases} a; & a \geq 0 \\ -a; & a < 0 \end{cases}$$

anstatt

$$|a| = \begin{cases} a; & a \geq 0 \\ -a; & a < 0 \end{cases}$$

Beispiel 14: Manipulierte Ausrichtung mithilfe von `\phantom`

`\begin{gathered}`*Formelcode*`\end{gathered}`

Die `gather`-Umgebung zentriert Formeln einfach nur horizontal. Es existiert hierfür ebenfalls eine Sternchenform, die - wie üblich - die Nummerierung unterdrückt.

Die `gathered`-Umgebung hat die gleiche Verwendungsweise wie `aligned`, jedoch ohne Positionierung. Hier wird wie wie bei `gather` einfach nur zentriert.

`\begin{multline}`*Formelcode*`\end{multline}`

Die `multline`-Umgebung ist ebenfalls eine mehrzeilige Umgebung, jedoch wird sie insbesondere für sehr lange Gleichungen verwendet:

links
zentriert
zentriert
zentriert
rechts (4.1)

Hierbei sind keine &-Zeichen nötig. Es wird lediglich ein einfacher Zeilenumbruch (`\\`) nach jeder Zeile gesetzt. Die erste Zeile wird automatisch linksbündig, die mittleren zentriert und die letzte Zeile rechtsbündig gesetzt.

Anmerkung: Um einfacher einen ordentlichen Formelsatz zu erreichen existiert der Befehl

`\phantom{`*Phantomtext*`}`

Er setzt eine leere Box in der Größe des „Phantomtextes“. Somit kann man einen Abstand der Breite eines Objektes einfügen, siehe Beispiel 14.

14.5 Referenz und Bezug

Querverweise innerhalb von Formeln kann, wie auch im laufenden Text mit `\label` und `\ref` gesetzt werden. Zusätzlich liefert jedoch das amsmath-Paket den Befehl

`\eqref{`*Markername*`}`

dieser setzt Klammern (vgl. (1.1) statt vgl. 1.1) um die Zahl und kennzeichnet den Bezug somit unverwechselbar als Formel. Zusätzlich bietet sich noch die Möglichkeit die Beschriftung mit dem Befehl

`\tag{`*Beschriftung*`}`

```
\begin{equation}
a^2+b^2=c^2 \tag{Satz des Pythagoras}\label{pythagoras}\\
\end{equation}
Der \ref{pythagoras} ist einer der fundamentalen Sätze der euklidischen
    Geometrie.
```

$$a^2 + b^2 = c^2 \qquad \text{(Satz des Pythagoras)}$$

Der Satz des Pythagoras ist einer der fundamentalen Sätze der euklidischen Geometrie.

Beispiel 15: Manuelle Formelbeschriftung mit `\tag`

Tabelle 14.1: Mathematische Akzente

$\hat{a}$	`\hat{a}`	$\acute{a}$	`\acute{a}`	$\bar{a}$	`\bar{a}`	$\dot{a}$	`\dot{a}`
$\check{a}$	`\check{a}`	$\grave{a}$	`\grave{a}`	$\vec{a}$	`\vec{a}`	$\ddot{a}$	`\ddot{a}`
$\breve{a}$	`\breve{a}`	$\tilde{a}$	`\tilde{a}`				

manuell zu generieren, vgl. Beispiel 15. Zudem existiert eine Sternchenform `\tag*`, bei der die Klammern um die *Beschriftung* entfallen.

14.6 Mathematische Akzente

Die in Abschnitt 3.3.5 beschrieben Akzentbefehle sind für den Gebrauch in normalem Text bestimmt. Ihre Verwendung ist im Mathemodus nicht zulässig. Die Akzente für den Mathemodus sind in Tabelle 14.1 gezeigt. Zu Demonstrationszwecken dient hier der Buchstabe „a“, sie können jedoch auf allen Buchstaben plaziert werden.

Zusätzlich zu den einfachen Befehlen `\hat` und `\tilde` gibt es auch breitere Versionen, die über mehrere Buchstaben gesetzt werden können. So produziert zum Beispiel das Makro `\widehat{1-x}` die Zeichenkombination $\widehat{1-x}$. `\widetilde` funktioniert analog.

14.7 Exponenten und Indizes

Hochstellen geschieht mittels ^und Tiefstellen mit _ innerhalb der Matheumgebungen. Auch mehrfache Indizes mit beliebiger Tiefe sind möglich. Sollen mehr als ein Zeichen hochgestellt werden, muss die Zeichenfolge in geschweifte Klammern gesetzt werden, sonst ist die Reihenfolge des Hoch- und Tiefstellens mehrdeutig definiert.

```
\(\displaystyle
    x^2\hfill a_n+\hfill x_i^n+\hfill x^{2n}\hfill x_{2y}\hfill
    A_{i,j,k}^{-n+2} \hfill x^{y^{z^2}}\hfill x^{y_1}\hfill
    A^{x_i^2}_{j^{2n}_{n,m}}
\)
```

$$x^2 \qquad a_n + \qquad x_i^n + \qquad x^{2n} \qquad x_{2y} \qquad A_{i,j,k}^{-n+2} \qquad x^{y^{z^2}} \qquad x^{y_1} \qquad A^{x_i^2}_{j^{2n}_{n,m}}$$

Darüber hinaus ist es mit dem mathtools-Paket auch möglich Indizes auf der linken Seite von Zeichen oder ausdrücken zu setzen.

```
\begin{gather*}
\frac{1}{x+y}\\
\frac{a^2+b^2}{a+b}\\
\frac{\frac{a}{x-y}+\frac{b}{x+y}}{
    1+\frac{a-b}{a+\frac{b}{a^2}}}
\end{gather*}
```

$$\frac{1}{x+y}$$
$$\frac{a^2+b^2}{a+b}$$
$$\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$$

Beispiel 16: Brüche

`\prescript{`*Index oben links*`}{`*Index unten links*`}{`*Objekt zu dem die Indizes gehören*`}`

Häufig sieht man auch Varianten wie `$^{y}_{xz}A$` um Indizes auf der linken Seite zu platzieren. Die Ausrichtung und die Abstände sind in diesem Fall jedoch formal nicht korrekt. Daher sollte eher die mathtools-Variante genutzt werden.

Hat man in einem Ausdruck verschiedene Indizes und Exponenten, so muss bei den Variablen ohne Exponent ein leerer Exponent gesetzt werden, damit alle Indizes sich auf gleicher Höhe befinden:

```
$a_l^2 b_m^{} c_n^2$\qquad
$a_l^2 b_m c_n^2$
```

$a_l^2 b_m^{} c_n^2$ $\qquad$ $a_l^2 b_m c_n^2$

14.8 Brüche und Binomialkoeffizienten

Für Brüche wird folgender Befehl genutzt.

`\frac{`*Zähler*`}{`*Nenner*`}`

Der Befehl setzt den Bruch und erzeugt einen Bruchstrich der Länge des Zählers oder Nenners, je nachdem, welcher breiter ist, siehe Beispiel 16.

Das amsmath-Paket liefert das Makro `\genfrac` („generalized fraction“). Es benötigt sechs Parameter, die teilweise leer bleiben können, allerdings sind alle notwendig, und dürfen nicht wegfallen. Die Argumente werden im Anschluss an die Syntax erläutert.

`\genfrac{`*Links*`}{`*Rechts*`}{`*Liniendicke*`}{`*Mathe-Stil*`}{`*Dividend*`}{`*Divisor*`}`

Links/Rechts Linkes bzw. rechtes Begrenzer-Symbol. Bei einem Bruch ist dieses Argument leer, bei einem Binomialkoeffizienten setzt man hier Runde Klammern.

Liniendicke Dicke des Bruchstriches. Ein leerer Parameter liefert hier die Standardstärke. Für einen Binomialkoeffizienten *muss* dieser Parameter auf `0pt` gesetzt werden.

Mathe-Stil Die mathematischen Schriftstile werden hier durch eine Ziffer kodiert ausgewählt. `0` = `\displaystyle`; `1` = `\textstyle`; `2` = `\scriptstyle`; `3` = `\scriptscriptstyle` Bleibt dieses Feld leer, erzwingt das eine Beachtung des aktuellen mathematischen Stils.

Dividend Der Zähler des Bruches bzw. der obere Teil des Binomialkoeffizienten.

Divisor Nenner des Bruches bzw. der untere Teil des Binomialkoeffizienten.

\textstyle $\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$ äußerster Bruch im \displaystyle $\dfrac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$

gesamter Bruch im \displaystyle $$\dfrac{\dfrac{a}{x-y}+\dfrac{b}{x+y}}{1+\dfrac{a-b}{a+\dfrac{b}{a^2}}}$$

Beispiel 17: Größere Brüche mit \dfrac

amsmath definiert intern noch ein paar Kurzformen für die alltägliche Anwendung von \genfrac:

\binom{*oberer Teil*}{*unterer Teil*}
\dfrac{*Zähler*}{*Nenner*}

Binomialkoeffizient, Bruch im \displaystyle (vgl. Abschnitt 14.3) Im normalen Textsatz werden die Formeln dann größer dargestellt, was eine bessere Lesbarkeit mit sich bringt, jedoch die Zeilenabstände unschön beeinflusst. Aber das Makro bietet eine Möglichkeit Doppelbrüche auch im Zeilenmodus lesbar darzustellen, wenn es denn sein muss, vgl. Beispiel 17.

14.9 Wurzeln

Wurzelausdrücke werden erzeugt mit

\sqrt[*Ordnung*]{*Formelcode*}

```
\begin{gather*}
\sqrt{a}\\
\sqrt[3]{8}\\
\sqrt[n+3]{\frac{-q+\sqrt{q+p}}{
    1+q^3}}
\end{gather*}
```

$$\sqrt{a}$$
$$\sqrt[3]{8}$$
$$\sqrt[n+3]{\frac{-q+\sqrt{q+p}}{1+q^3}}$$

14.10 Operatoren

Operatoren können sowohl durch einzelne Zeichen, als auch Namen ausgedrückt werden. Im allgemeinen ist die Wahl der Schreibweise willkürlich, jedoch werden grundsätzlich alle Operatoren aufrecht gesetzt. Zudem werden andere Abstände sowohl vor als auch nach dem Ausdruck gesetzt. LaTeX hat bereits einige Standardbefehle für die häufigsten Operatoren definiert. Dazu gehören unter anderem die Befehle für Summen, Produkte, Limites und Integrale:

\sum_{*Laufindex=Startwert*}^{*Endwert*}
\prod_{*Laufindex=Startwert*}^{*Endwert*}
\lim_{*Variable* \to *Grenzwert*}
\int_{*untere Grenze*}^{*obere Grenze*}

Für eine vollständige Auflistung der Standardmäßigen Symbol-Operatoren, siehe Tabelle 14.2.

Tabelle 14.2: In Standard-LaTeX verfügbare Symbol-Operatoren mit `limits`

`\int`	$\int$	`\sum`	$\sum$	`\prod`	$\prod$	`\bigvee`	$\bigvee$	`\smallint`	$\smallint$
`\bigwedge`	$\bigwedge$	`\biguplus`	$\biguplus$	`\bigcap`	$\bigcap$	`\bigcup`	$\bigcup$	`\bigotimes`	$\bigotimes$
`\bigoplus`	$\bigoplus$	`\bigodot`	$\bigodot$	`\oint`	$\oint$	`\bigsqcup`	$\bigsqcup$		

```
\begin{gather*}
% displaystyle is aktiv
2\sum_{i=1}^na_i\int_a^bf_i(x)\,dx\\
\textstyle
2\sum_{i=1}^na_i\int_a^bf_i(x)\,dx\\
\sum_{i=1}^na_i\qquad
\sum_{\substack{i=1\\j=1\\k=1}}^{
    \substack{\infty\\k\\j}}
\end{gather*}
```

$$2\sum_{i=1}^{n} a_i \int_a^b f_i(x)\,dx$$

$$2\textstyle\sum_{i=1}^{n} a_i \int_a^b f_i(x)\,dx$$

$$\sum_{i=1}^{n} a_i \qquad \sum_{\substack{i=1\\j=1\\k=1}}^{\substack{\infty\\k\\j}}$$

Beispiel 18: Operatoren mit und ohne `\limits`

Diese Symbol-Operatoren erscheinen in drei Größen, je nachdem in welcher Schriftstil aktiv ist (vgl. Abschnitt 14.3). Für die Position der Indizes gibt es ebenfalls eine Unterscheidung nach Schriftstil, es gibt jedoch eine Möglichkeit diese Einstellung lokal umzuschalten:

`\nolimits`	Setzt die Indizes nicht als „Grenzen“.
`\limits`	Setzt, sofern dies für den aktuellen Operator möglich ist, die Indizes als Grenzen

Diese Befehle müssen zwischen dem Operator und der Angabe für die Indizes stehen, siehe auch Beispiel 18. Falls ein Operator das `\limits`-Makro nicht verarbeiten kann, wird es automatisch ignoriert und bewirkt nichts.

Der Befehl `\substack` aus dem amsmath-Paket erlaubt es mehrzeilige Indizes (oder Exponenten) zu setzen.

`\substack{`*Code*`\\`*Code*`\\...}`

Zusätzlich zu den Symboloperatoren gibt es noch eine Reihe Operatornamen, z. B. die trigonometrischen Funktionen, siehe Tabelle 14.3. Viele von Ihnen, wie zum Beispiel `\sin` können das `\limits`-Makro nicht verarbeiten. Bei Ihnen werden immer Indizes und niemals Grenzen gesetzt.

Tabelle 14.3: Weitere in Standard-LaTeX definierte Operatoren, viele davon ohne `\limits`

`\log`	log	`\lg`	lg	`\ln`	ln	`\lim`	lim
`\limsup`	lim sup	`\liminf`	lim inf	`\sin`	sin	`\arcsin`	arcsin
`\sinh`	sinh	`\cos`	cos	`\arccos`	arccos	`\cosh`	cosh
`\tan`	tan	`\arctan`	arctan	`\tanh`	tanh	`\cot`	cot
`\coth`	coth	`\sec`	sec	`\csc`	csc	`\max`	max
`\min`	min	`\sup`	sup	`\inf`	inf	`\arg`	arg
`\ker`	ker	`\dim`	dim	`\hom`	hom	`\det`	det
`\bmod`	mod	`\Pr`	Pr	`\gcd`	gcd	`\deg`	deg
`\exp`	exp						

```
\DeclareMathOperator{\foo}{foo}%in der Präambel
\DeclareMathOperator*{\baz}{baz}%in der Präambel

\[\foo_1^2 = \baz\nolimits_1^2 = \foo\limits_1^2 = \baz_1^2\]
```

$$\mathrm{foo}_1^2 = \mathrm{baz}_1^2 = \mathrm{foo}_1^2 = \mathop{\mathrm{baz}}_1^2$$

Beispiel 19: Eigene Operatoren definieren

Eigene Operatoren definieren

Die Definition eigener Operatoren ist zwar auch mit Standard-LaTeX relativ einfach, jedoch wird sie durch das Paket amsopn von Michael Downes, welches automatisch mit amsmath geladen wird, noch weiter vereinfacht.

`\DeclareMathOperator*{`*Makro*`}{`*Operatorname*`}`	Kann `\limits` benutzen.
`\DeclareMathOperator{`*Makro*`}{`*Operatorname*`}`	

Beide Makros dürfen lediglich in der Präambel verwendet werden. Die dadurch erzeugten Markos können dann analog zu den vordefinierten Operatoren benutzt werden, siehe Beispiel 19.

Möchte man einen bestimmten Operator lediglich in einem Einzelfall verwenden, so liefert amsmath zwei Makros zum Satz von Operatornamen. Die Unterscheidung ist hierbei dieselbe, wie bei `\DeclareMathOperator`:

`\operatorname*{`*Operatorname*`}`
`\operatorname{`*Operatorname*`}`

14.11 Spezielle Buchstaben und Zeichen

Griechische und hebräische Zeichen können innerhalb des Mathemodus oft direkt entsprechend ihrem Wortlaut eingegeben werden, z. B. `\alpha` α. Der entsprechende Großbuchstabe wird dabei durch Großschreibung des ersten Zeichens des Makronames erreicht: `\Phi` (Φ). Da diese Großbuchstaben oft als Operatoren verwendet werden, werden Sie nach Voreinstellung aufrecht gesetzt. Viele der griechischen Großbuchstaben entsprechen jedoch den Zeichen des lateinischen Alphabets (z. B. $\alpha \longrightarrow \mathrm{A}$). Daher existiert für diese Buchstaben kein entsprechendes Makro.

Aufgrund der riesigen Anzahl an mathematischen Symbolen, ist es kaum sinnvoll hier alle aufzulisten. Viele Editoren bieten jedoch intern Symbolleisten oder Listen, die das fertige Symbol enthalten und bei Aktivierung den zugehörigen Quellcode an der Cursorposition schreiben. Darüber hinaus existieren auch andere Ansätze um den Nutzer bei der Suche nach dem richtigen Makronamen zu unterstützen.

Die wohl flexibelste Variante stelt dabei das Onlinetool „Detexify“ ([1]: `http://detexify.kirelabs.org`) dar. Damit ist es möglich die Form des gesuchten Zeichens in ein Kästchen zu „malen“. Anschließend wird die eingezeichnete Form mit den Daten der Symbole verglichen und der Benutzer erhält eine Liste von Vorschlägen, aus der er das gesuchte Zeichen auswählen kann. Darüber hinaus wird auch die genaue Verwendung und ein ggf. notwendige Zusatzpaket angezeigt.

Für die Suche außerhalb des Internets ist „The Comprehensive LaTeX Symbol list“ [17] eine gute, wenn auch nicht ganz so komfortable Alternative. Die dazugehörige Datei `symbols-a4.pdf`

Tabelle 14.4: Die in Standard-LaTeX verfügbaren Klammern. Die Klammernbefehle, welchen bereits ein `\big` vorangestellt wurde, existieren nicht in der kleinsten Größe.

`()`	$()$	`[]`	$[]$
`\{\}`	$\{\}$	`\langle\rangle`	$\langle\ \rangle$
`/|\backslash`	$/\ \vert\ \backslash$	`\big\lmoustache` `\big\rmoustache`	$\big\lmoustache\ \big\rmoustache$
`\arrowvert\vert`	$\arrowvert\ \vert$	`\Vert\Arrowvert`	$\Vert\ \Arrowvert$
`\uparrow\downarrow` `\updownarrow`	$\uparrow\ \downarrow\ \updownarrow$	`\Uparrow\Downarrow` `\Updownarrow`	$\Uparrow\ \Downarrow\ \Updownarrow$
`\lceil\rceil`	$\lceil\ \rceil$	`\lfloor\rfloor`	$\lfloor\ \rfloor$
`\big\lgroup\big\rgroup`	$\big\lgroup\ \big\rgroup$	`\big\bracevert`	$\big\bracevert$

ist Bestandteil der Basis jeder TeX-Distribution und über das Programm `texdoc` (vgl. auch Abschnitt 3.2) abrufbar:

Kommandozeile

```
texdoc symbols-a4
```

14.12 Klammern

Die Größe einzelner mathematischer Klammern kann manuell mit

`\big`*Klammerzeichen/-makro*
`\Big`*Klammerzeichen/-makro*
`\bigg`*Klammerzeichen/-makro*
`\Bigg`*Klammerzeichen/-makro*

eingestellt werden. Die verwendbaren Klammerzeichen und -makros sind in Tabelle 14.4 aufgelistet.

Da Klammern meistens paarweise auftreten, stellt LaTeX auch einen Mechanismus zur Verfügung, mit dem sich die Größe automatisch an den Inhalt anpasst.

`\left(` ... `\right)`

Für die korrekte Begrenzung der Höhe ist es jedoch zwingend erforderlich sowohl eine linke als auch rechte Begrenzung zu setzen. Allerdings ist es über die Syntax `\left.` bzw. `\right.` möglich, ein Klammernsymbol zu unterdrücken, siehe Beispiel 20.

Für Fallunterscheidungen, wie der letzte Teil von Beispiel 20 stellt amsmath eine eigene Umgebung `cases` zur Verfügung. Ihre Verwendung ist zusätzlich in Beispiel 14 auf Seite 122 gezeigt.

`\overbrace` und `\underbrace`

Neben den normalen vertikalen Klammern, existieren auch horizontale geschweifte Klammern um damit Formelsegmente beschriften zu können:

```
\[
\left(\begin{array}{c}9 \\ 2 \\ 4 \end{array}\right)\qquad
2\pi\hbar\left\{a+\frac{\frac{3}{2}\sum x_i^2}{m^*}\right\}\qquad
y=\left\{
    \begin{array}{r@{\quad:\quad}l}
        -1 & x<0 \\0 & x=0 \\1 & x>0
    \end{array}
\right.
\]
```

$$\left(\begin{array}{c}9 \\ 2 \\ 4 \end{array}\right)\qquad 2\pi\hbar\left\{a+\frac{\frac{3}{2}\sum x_i^2}{m^*}\right\}\qquad y=\left\{\begin{array}{r@{\quad:\quad}l} -1 & x<0 \\0 & x=0 \\1 & x>0 \end{array}\right.$$

Beispiel 20: Automatische Größenanpassung von Klammern

```
\[
\left(
\begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{array}
\right)
\]
```

$$\left(\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array}\right)$$

Beispiel 21: Aufbau einer Matrix

```
\overbrace{Formelsegment}^{Beschriftung}
\underbrace{Formelsegment}_{Beschriftung}
```

Die Verwendung erscheint dann in folgender Form:

$$\overbrace{Formelsegment}^{\text{mit overbrace}} \qquad \underbrace{Formelsegment}_{\text{mit underbrace}}$$

14.13 Matrizen

Matrizen können auf dieselbe Weise wie Textmodus-Tabellen erzeugt werden. Anstatt der `tabular`-Umgebung kommt dann `array` zum Einsatz, vgl. Beispiel 21.

Da man bei Matrizen meistens keine links- oder rechtsbündige Ausrichtung benötigt und die Angabe der Spalten somit relativ mühselig ist, liefert amsmath auch hier spezielle Umgebungen. Die Syntax innerhalb der Umgebungen entspricht der Verwendung in Beispiel 21. Man kann somit die Klammern zusammen mit der `array`-Umgebung durch eine `pmatrix`-Umgebung ohne Argumente ersetzen.

$$\texttt{Vmatrix}\begin{Vmatrix} a & b \\ c & d \end{Vmatrix} \quad \texttt{Bmatrix}\begin{Bmatrix} a & b \\ c & d \end{Bmatrix} \quad \texttt{matrix}\begin{matrix} a & b \\ c & d \end{matrix}$$

$$\texttt{vmatrix}\begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad \texttt{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \texttt{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

14.14 Overset und Underset

`\overset{`*Ausdruck drüber*`}{`*Formeltext*`}`
`\underset{`*Ausdruck drunter*`}{`*Formeltext*`}`

amsmath bietet mit diesen Befehlen die Möglichkeit Objekte direkt über oder unter Formelfragmenten anzuordnen.

$$\overset{\text{Ausdruck drüber}}{Formel} \qquad \underset{\text{Ausdruck drunter}}{Formel}$$

Zum Beispiel wird dieser Befehl benutzt um Beschriftungen zu erzeugen oder chemische Formeln mit Oxidationszahlen zu versehen:

$$\overset{\text{+VII}}{\mathrm{Mn}}\mathrm{O_4^-} + 8\,\mathrm{H_3O^+} + 5\,\underset{\uparrow\text{freies Elektron}}{\mathrm{e^-}} \longrightarrow \overset{\text{+II}}{\mathrm{Mn}}^{2+} + 12\,\mathrm{H_2O}$$

Zusätzlich existiert das Makro

`\sideset{_{`*links unten*`}^{`*links oben*`}}{_{`*rechts unten*`}^{`*rechts oben*`}}{`*Operator*`}`

um Indizes bei Operatoren (vgl. Abschnitt 14.10) auch linksseitig platzieren zu können.

$$\sideset{_{\text{links unten}}^{\text{links oben}}}{_{\text{rechts unten}}^{\text{rechts oben}}}{\prod_{\text{unten}}^{\text{oben}}}$$

14.15 Text im Mathemodus

Mit dem amsmath-Paket gibt es den Befehl

`\text{`*Text*`}`

Hier wird der Text entsprechend des mathematischen Schriftstiles gewählt und als richtige Textbox formatiert. Das Makro `\mathrm` setzt dahingegen keine Textbox, sondern lediglich Buchstaben aufrecht. Leerzeichen innerhalb des Argumentes werden (wie auch sonst im Mathe-Modus) ignoriert. Zudem enthält dieser Schrifttyp als Mathematikschrift keine Ligaturen. Das Makro `\mathrm` sollte somit nicht zum Satz von Worten, sondern lediglich zum Satz einzelner Zeichenfolgen dienen.

Soll Text zwischen die Zeilen einer mehrzeiligen Formel eingefügt werden, so geschieht dies mittels

`\intertext{`*Text*`}`

Der Text wird dann so gesetzt, als würde die Matheumgebung beendet und anschließend neu begonnen. Dies ermöglicht jedoch die Ausrichtung fortzusetzen, obwohl ein Text eingefügt wird.

Tabelle 14.5: Von amsthm vordefinierte Theorem-Stile

Stilname	*Beschreibung*	*Beispiel*
`plain`	Standard. Für Theoreme, Lemmata, Propositionen, usw.	**Theorem 1.** *Text*
`definition`	Für Definitionen und Beispiele	**Definition 2.** Text
`remark`	Für Bemerkungen	*Bemerkung 3.* Text

Andere Möglichkeiten zum Setzen von Text im Mathemodus existieren zwar, sollten jedoch aufgrund der verminderten Anpassungsfähigkeiten im Gegensatz zu `\text` vermieden werden.

14.16 Theoreme

Der Autor kann auch eigene Strukturen bzw. Umgebungen mit eigenen Zähler deklarieren. So werden z. B. in den Naturwissenschaften manche Dokumente von einer Reihe von Axiomen oder Definitionen durchzogen. Der Autor kann für diese Strukturen eine einheitliche Formatierung festlegen.

`\newtheorem{`*Strukturname*`}[`*Zählung*`]{`*Strukturbegriff*`}[`*Gliederung*`]`

```
\newtheorem{Def}{Definition}[section]
\begin{Def}[Fermion]
Teilchen mit halbzahligen Spin.
\end{Def}
```

Definition 14.16.1 (Fermion)**.** Teilchen mit halbzahligen Spin.

Das amsthm-Paket ermöglicht es, den Theorem-Stil mit

`\theoremstyle{`*Stilname*`}`

zu ändern. Es gibt drei vordefinierte Stile, die den üblichen Layouts, die in der Mathematik verwendet werden entsprechen, siehe Tabelle 14.5.

Beweise
Zusätzlich definiert amsthm die `proof`-Umgebung für Beweise.

`\begin{proof}[`*Beweistitel*`] ... \end{proof}`

```
\begin{proof}[Beweisname]
Hier steht ein Beweis.
\end{proof}
```

Beweisname. Hier steht ein Beweis. □

14.17 Werte und Einheiten mit siunitx

Bei der manuellen Angabe von Werten und Einheiten (oder auch der Kombination aus beidem) müssen mehrere Aspekte beachtet werden (vgl. auch Abschnitt 14.1). Die wichtigsten Punkte sind hierbei:

- Unterscheidung zwischen Variablen, physikalischen Konstanten (beide kursiv) und Einheiten (aufrecht)

Tabelle 14.6: Wichtigste Optionen zu siunitx

Option=Wert	*Beispiel*	*Beschreibung*
`output-decimal-marker={,}`	1,23	`{,}` als Dezimaltrennzeichen anstatt des Punktes
`exponent-product=\cdot`	$5 \cdot 10^{20}$	Malpunkt anstatt des Kreuzes
`per-mode=reciprocal`	$\mathrm{m\,s^{-1}}$	Negative Potenzen
`per-mode=fraction`	$\frac{\mathrm{m}}{\mathrm{s}}$	Echter Bruch
`per-mode=symbol`	m/s	Schrägstrich als Bruch

- Abstand eines halben Leerzeichens zwischen Wert und Einheit
- Einheitliche Struktur der Angaben (insbesondere sprachliche Besonderheiten: Punkt/Komma, Malpunkt/Produktkreuz, …)

Das Paket siunitx liefert hierfür bequeme Möglichkeiten für die Umsetzung. Hierbei wird das Markup effektiv genutzt, sodass es für die Angabe innerhalb des Dokumentes nicht von Bedeutung ist, welche Sprache genutzt wird und welche Formalitäten einzuhalten sind. Diese Aspekte sind lediglich für die Paketeinstellungen wichtig.

`\usepackage[`*globale Optionen*`]{siunitx}`	
`\sisetup{lokale Optionen}`	Schalter für nachträgliche Änderungen

Eine Auswahl der wichtigsten Optionen findet sich in Tabelle 14.6.

14.17.1 Werte & Einheiten

siunitx liefert für die Angabe von Werten oder Einheiten verschiedene Makros, um Werte getrennt von den Einheiten an die Vorgaben anzupassen.

`\num[`*Optionen*`]{`*Wert*`}`	Wert
`\si[`*Optionen*`]{`*Einheiten*`}`	Einheit

Die Einheiten werden dafür als Textmakros eingegeben. Dies ermöglicht eine flexiblere Formatierung mithilfe der Einstellungen und verbessert die Lesbarkeit des Codes erheblich, z. B. `\si{\kilogram\metre\per\square\second}` ($\mathrm{kg\,m\,s^{-2}}$)

Da das Paket mehr als nur die SI-Einheiten vordefiniert und zusätzlich noch einen einfachen Mechanismus zur Definition eigener Einheiten bereitstellt, sprengt eine komplette Übersicht über alle Möglichkeiten den hiesigen Rahmen. Die Einheiten lassen sich jedoch relativ einfach in der Paketanleitung [24] finden.

Für die Angabe der einer Wert-Einheiten-Kombination steht ebenfalls ein gesondertes Makro zur Verfügung, um den richtigen Abstand gewährleisten zu können:

`\SI[`*Optionen*`]{`*Wert*`}[`*Vor-Einheit*`]{`*Einheit*`}`

Das Zusätzliche Argument für die *Vor-Einheit* ermöglicht es eine Einheit vor den *Wert* zu setzen. Dies ist je nach Zusammenhang z. B. bei Währungen üblich:

```
\SI[per-mode=symbol]{1.99}[\$]{\per\kilogram}
```

$1,99/kg

Es ist auch möglich Einheiten und die zugehörigen Werte speziell zu formatieren. Da es sich hierbei jedoch um Spezialanwendungen handelt, sei wiederum auf die Anleitung [24] verwiesen.

```
\begin{tabular}{@{}S[table-format=2.3]cl@{}}
    \toprule
    {Werte}&\bef{c}-Spalte&\bef{l-Spalte}\\
    \midrule
    5.495&Text&mehr Text\\
    83.56\textsuperscript{x}&\multicolumn{2}{c@{}}{\tablenum{6.4}}\\
    {\$}
    4.567&\multicolumn{2}{c@{}}{\tablenum{6.44}}\\
    3.4&\multicolumn{2}{c@{}}{\tablenum{26.4}}\\
    \bottomrule
\end{tabular}
```

Werte	c-Spalte	l-Spalte
5,495	Text	mehr Text
83,56[x]	6,4	
$4,567	6,44	
3,4	26,4	

Beispiel 22: Verwendung der von siunitx definierten S-Spalte, sowie des `\tablenum`-Makros.

14.17.2 Wertetabellen

Das siunitx-Paket liefert neben den Möglichkeiten für den Satz von Werten und Einheiten auch Möglichkeiten für Wertetabellen.

Zum Beispiel ist es damit möglich, Werte innerhalb einer Tabelle am Dezimaltrennzeichen auszurichten. Der hierfür verwendete Spaltentyp ist die S-Spalte. Sie nimmt zunächst an, dass ihr Inhalt eine Dezimalzahl ist. Möchte man, dass der Inhalt anders behandelt wird, beispielsweise in einer Spaltenüberschrift, so kann man das durch Gruppierung erreichen. Wird in einer Spalte zusätzlich zu einer Dezimalzahl ein Element gefunden, so wird es je nach austreten einfach links bzw. rechts von der am Dezimaltrennzeichen ausgerichteten Zahl platziert. Zusätzlich ist es auch möglich als Option das Format der enthaltenen Ziffernfolgen anzugeben, vgl. Beispiel 22.

Zusätzlich zum neuen Spaltentyp existiert noch das Makro

`\tablenum[`*Optionen*`]{`*Wert*`}`

Damit ist es möglich, eine Ausrichtung am Trennzeichen auch für Zellen, die mit `\multicolumn` oder `\multirow` (siehe Kapitel 11) zusammengefasst wurden, zu erreichen. Ein Beispiel für die Verwendung ist in Beispiel 22 gezeigt.

Die wichtigste Option für Wertetabellen ist `table-format`. Sie dient dazu, den Platz, der für die Ausrichtung reserviert wird einzustellen. Sie kann wie alle siunitx-Optionen sowohl global als auch lokal gesetzt werden.

```
\sisetup{table-format = +2.3e+2}
```

Diese Einstellung bedeutet:

- Es soll genug Platz für ein Vorzeichen reserviert werden (erstes +)
- Die Werte haben maximal 2 Vorkomma- und 3 Nachkommastellen (`2.3`)
- Eine Angabe der Größenordnung in Exponentialschreibweise soll berücksichtigt werden (e), dabei soll ebenfalls ein Vorzeichen (zweites +) und bis zu 2 Stellen gesetzt werden.

Nicht vorhandene Teile, können bei der Angabe des `table-format` entfallen. Darüber hinaus gibt es noch Möglichkeiten für die Angabe von Messungenauigkeiten und eine lange Liste

weiterer Optionen für die Feinabstimmung von Tabellen, auch Formatierungen und Ausrichtung anpassen können. Diese Möglichkeiten sind jedoch sehr speziell und können bei Bedarf der Paketanleitung [24] entnommen werden.

14.18 Das mhchem-Paket

```
\usepackage[version=4]{mhchem}
```

Das mhchem-Paket vereinfacht die nötige Befehlsstruktur um chemische Formeln zu setzen. Wichtig ist es beim laden die richtige Version zu wählen, da es bei diesem Paket möglich ist unterschiedliche Versionen zu laden. Wir benutzen Version 4. Außerdem ermöglicht das Paket Indizes auf der linken Seite hinzuzufügen, was sich beim Setzen von Isotopen als besonders praktisch erweist.

```
\ce{Summenformel}
```

Dies ist der hauptsächliche Befehl. Im folgenden finden sich einige Möglichkeiten der Anwendung.

Summenformeln Einfache chemische Summenformeln werden durch Eingabe der enthaltenen Zeichen direkt hintereinander erzeugt.

```
\ce{
    H2O\quad Sb203\quad H+\quad CrO4^2-\quad [AgCl2]-\quad Y^{99}+
}
```

$\ce{H2O}\quad \ce{Sb203}\quad \ce{H+}\quad \ce{CrO4^2-}\quad \ce{[AgCl2]-}\quad \ce{Y^{99}+}$

Mengen Mengen werden direkt vor der restlichen Formel geschrieben:

```
\ce{2H2O}\quad\ce{1/2H2O}
```

$\ce{2H2O}\quad\ce{1/2H2O}$

Isotope Indizes auf der linken Seite

```
\ce{^{227}_{90}Th+}
```

$\ce{^{227}_{90}Th+}$

Bindungen Weitere Varianten finden sich in der Paketanleitung [5].

```
\ce{A\bond{1} B\bond{2} C\bond{3} D}
```

$\ce{A\bond{1} B\bond{2} C\bond{3} D}$

Formeln Beispiele für verschiedene Arten von Reaktionspfeilen (auch mit Beschriftungen):

```
\ce{CO2 + C -> 2CO}\\
\ce{CO2 + C <- 2CO}\\
\ce{CO2 + C <=> 2CO}\\
\ce{H+ + OH- <=>> H2O}\\
\ce{$A$ <-> $A'$}\\
\ce{CO2 + C ->[\alpha][\beta]2CO}
```

$\ce{CO2 + C -> 2CO}$\
$\ce{CO2 + C <- 2CO}$\
$\ce{CO2 + C <=> 2CO}$\
$\ce{H+ + OH- <=>> H2O}$\
$\ce{$A$ <-> A'}$\
$\ce{CO2 + C ->[\alpha][\beta]2CO}$

Im Mathemodus werden sämtliche Elementsymbole immer aufrecht gesetzt. Im Textmodus wird die aktuelle Schriftart auch für die Formeln übernommen.

Das Paket mhchem bietet noch mehr Möglichkeiten zur Formatierung chemischer Formeln. Alle Befehle findet man selbstverständlich in der Dokumentation [5].

14.19 Kommutative Diagramme

In der Mathematik werden kommutative Diagramme dafür verwendet zu demonstrieren, dass unterschiedliche Verkettungen von Abbildungen das gleiche Ergebnis liefern. Häufig werden kommutative Diagramme mit dem Paket xy gesetzt (Für die zugehörige Anleitung siehe [21]), jedoch gibt es ein neueres Paket, das noch mehr Flexibilität liefert und eine einfachere Syntax benutzt. Daher werden wir uns in diesem Kurs auf kommutative Diagramme mit dem Paket tikz-cd beschränken.

Das Paket tikz-cd basiert, wie der Name schon sagt auf dem Paket Ti*k*Z, welches für die Erstellung von Grafiken jeglicher Art mit pgf („portable graphics format“) benutzt wird. Viele Features von Ti*k*Z können auch in Kombination mit der Bibliothek cd, welche dem Paket tikz-cd entspricht, benutzt werden. Der Funktionsumfang von Ti*k*Z übersteigt jedoch den Umfang dieses Kurses. Jedoch wird ein Teil der Funktionalität im Fortgeschrittenenkurs behandelt. Für weiterführende Informationen, zum Beispiel um den Stil einzelner Diagramme zu verändern, sei somit auf den Fortgeschrittenenkurs oder die Anleitung [22] verwiesen.

14.19.1 Die Struktur der Diagramme

Das Laden des Paketes funktioniert wir üblich, allerdings muss bei Verwendung des babel-Pakets noch eine Bibliothek zusätzlich geladen werden. Das beruht auf der Tatsache, dass babel die Bedeutung verschiedener Sonderzeichen, zum Beispiel ", verändert. Die Bibliothek babel deaktiviert diese Wirkung für den Ti*k*Z-spezifischen Code. Insgesamt wird das Paket und die Bibliothek dann folgendermaßen geladen:

```
\usepackage{tikz-cd}
\usetikzlibrary{babel}
```

Optionen beim Laden sind nicht notwendig. Einzelne Diagramme werden dann in der Umgebung tikzcd gesetzt, die allgemeine Optionen für den Stil des Diagramms verarbeiten kann.

```
\begin{tikzcd}[Optionen]
Diagrammcode
\end{tikzcd}
```

Die Struktur des Diagrammcodes ähnelt dabei dem einer Tabelle. Spalten werden durch & und Zeilen durch \\ getrennt. Alle Elemente innerhalb dieser Tabelle werden im Mathemodus gesetzt. Dennoch empfiehlt es sich häufig, das Diagramm innerhalb einer abgesetzten Mathe-Umgebung zu setzen um es zu zentrieren und gleiche Abstände, wie bei Formeln zu erhalten. Hier ein Beispiel für die Struktur:

```
\begin{tikzcd}
    A&B\\
    C&D
\end{tikzcd}
```

$$\begin{tikzcd} A & B \\ C & D \end{tikzcd}$$

14.19.2 Pfeile

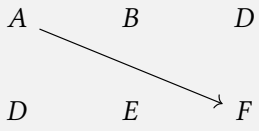
Innerhalb der `tikzcd`-Umgebung können die folgenden beiden Makros äquivalent zur Erzeugung von Pfeilen genutzt werden.

`\arrow[`*Optionen*`]`
`\ar[`*Optionen*`]`

Ein Pfeil beginnt immer in dem Feld der „Tabelle“ in dem der zugehörige Befehl aufgerufen wird. Die Optionen dienen dazu zu spezifizieren, in welche Richtung ein Pfeil zeigt, Beschriftungen hinzuzufügen oder Änderungen am Pfeil selbst vorzunehmen. Die einzelnen Optionen werden als Komma-separierte Liste angegeben.

Das Ziel eines Pfeiles wird durch Richtungsparameter angegeben. Der Richtungsparameter ist eine Liste aus den Buchstaben `r` („right“), `l` („left“), `d` („down“) und `u` („up“). So zeigt ein Pfeil mit dem Richtungsparameter `rrd`, zwei Felder nach rechts und eines nach unten.

```
\begin{tikzcd}
    A\arrow[rrd]&B&D\\
    D&E&F
\end{tikzcd}
```



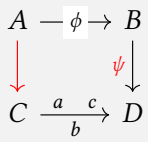
Pfeilbeschriftungen

Beschriftungen für Pfeile werden mithilfe der in [22, Abschnitt 17.10.4] beschriebenen Syntax:

"Beschriftung" Optionen

Durch die *Optionen* kann angegeben werden, wo genau sich der Beschriftungstext befinden oder welche Farbe verwendet werden soll. Eine einzelne Option kann ohne Gruppierung angegeben werden. Für mehrere ist diese jedoch notwendig und die Optionen sind durch Komma zu trennen. Ohne Angabe von Optionen werden die Beschriftungen immer darüber (horizontale Pfeile) oder rechts daneben (vertikale Pfeile) gesetzt. Die Positionierung kann in Form einer Richtung angegeben werden (`left`, `right`, `above`, `below`), mithilfe einer relativen Positionierung ((`very`) `near start`, (`very`) `near end`) oder mit der Option `description`. `description` setzt dabei die Beschriftung direkt auf den Pfeil, siehe auch folgendes Beispiel:

```
\begin{tikzcd}
    A \arrow[r, "\phi" description] \arrow[d, red]
    & B \arrow[d, "\psi"{red,left}] \\
    C \arrow[r, "a" near start, "b"', "c" near end]
    & D
\end{tikzcd}
```



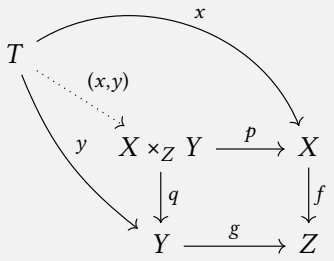
Im obigen Beispiel wird zusätzlich gezeigt, dass ein Apostroph als Option die Position an der Pfeilachse spiegelt.

Gebogene Pfeile

Pfeile können einfach durch Angabe der Optionen `bend left` oder `bend right` gebogen werden. Optional kann als Wert für beide Optionen ein Winkel angegeben werden[1]. Der Winkel gibt an in welcher Richtung der Pfeil das Feld verlässt bzw. unter welchem Winkel er in das Zielfeld eintritt. Die Wirkung wird im folgenden Beispiel verdeutlicht.

[1] Ohne Angabe des Winkels wird der zuletzt angegebene Winkelwert verwendet. Wurde nie ein Winkel angegeben, so wird der Wert 30 benutzt.

```
\begin{tikzcd}
    T
    \arrow[drr, bend left=50,"x"]
    \arrow[ddr, bend right=15,"y"]
    \arrow[dr, dotted,"{(x,y)}"]\\
    &X\times_Z Y\arrow[r,"p"]\arrow[d,"q"]
    &X\arrow[d,"f"] \\
    &Y\arrow[r,"g"]
    &Z
\end{tikzcd}
```



Standardpfeile

```
to head
rightarrow
leftarrow
leftrightarrow
Rightarrow
Leftarrow
Leftrightarrow
```

Mapsto

```
maps to
mapsto
mapsfrom
Mapsto
Mapsfrom
```

Pfeile mit Haken

```
hook
hook'
hookrightarrow
hookleftarrow
```

Pfeile mit Schwanz

```
tail
rightarrowtail
leftarrowtail
```

Zweiköpfige Pfeile

```
two heads
twoheadrightarrow
twoheadleftarrow
```

Harpoonen

```
harpoon
harpoon'
rightharpoonup
rightharpoondown
leftharpoonup
leftharpoondown
```

Gestrichelte Pfeile

```
dashed
dashrightarrow
dashleftarrow
```

Schlängelnde Pfeile

```
squiggly
rightsquigarrow
leftsquigarrow
leftrightsquigarrow
```

Pfeile ohne Spitze

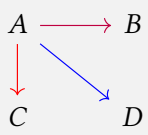
```
no head
no tail
dash
equal
```

Ein graues Kreuz in den obigen Beispielen bedeutet, dass das entsprechende Ende des Pfeiles von der Einstellung nicht geändert wird. Das erlaubt Pfeile nach eigenen Wünschen zusammenzubauen.

Absolute Positionierung von Pfeilen

Üblicherweise beginnt ein Pfeil immer in dem Feld der Tabellenstruktur, in dem das zugehörige Makro aufgerufen wird. Mit den Optionen `to` und `from` ist es jedoch möglich dieses Verhalten zu umgehen. Damit ist es auch möglich Pfeile zu setzen, die von anderen Pfeilen anstatt von einem Feld aus weg zeigen.

```
\begin{tikzcd}
    A \arrow[to=Z, red] \arrow[to=2-2, blue]
    & B \\
    |[alias=Z]| C
    & D
    \arrow[from=ul, to=1-2, purple]
\end{tikzcd}
```



Das Beispiel zeigt, wie Felder benannt werden können. Entweder über die Syntax

`|[alias=`*Name*`]|`

oder über die Zahl der Spalten und Zeilen, die sie vom obersten linken Element entfernt sind. So meint die Bezeichnung `1-2` das Feld in Zeile 1 und Spalte 2. `2-1` bezeichnet dahingegen die zweite Zeile der ersten Spalte, usw.

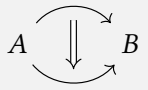
Da es auch möglich ist relative Ortsangaben mithilfe der Richtungsparameter zu machen, sollte es vermieden Werden Felder mit Buchstabenkombinationen zu benennen, die nur aus `l`, `r`, `u` und `d` bestehen.

Wie bereits erwähnt ist es mit dieser absoluten Positionierung auch möglich Pfeile von Pfeilen ausgehen zu lassen. Streng genommen, geht der Pfeil jedoch nicht vom Pfeil aus sondern von der Beschriftung. Diese ist eine Node (Knotenpunkt) und Knotenpunkte können in Ti*k*Z wie Koordinaten benannt werden. Das funktioniert mithilfe der Option `name`:

`\arrow[`*Richtungsparameter*`, "`*Beschriftung*`"{name=`*Name*`,` *weitere Optionen*`}]`

In der Praxis sieht das dann folgendermaßen aus:

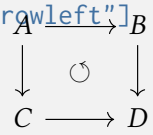
```
\begin{tikzcd}
    A \arrow[r, bend left=50, ""{name=U, below}]
    \arrow[r, bend right=50, ""{name=D}]
    & B
    \arrow[Rightarrow, from=U, to=D]
\end{tikzcd}
```



Unsichtbare Pfeile

Mithilfe der Option `phantom` ist es möglich Pfeile unsichtbar zu machen. Die Beschriftung wird jedoch keinesfalls unsichtbar. Sie wird zentriert auf der unsichtbaren „Linie“ ausgegeben.

```
\begin{tikzcd}
    A \arrow[r] \arrow[d] \arrow[dr, phantom, "\circlearrowleft"] B
    & B \arrow[d] \\
    C \arrow[r]
    & D
\end{tikzcd}
```

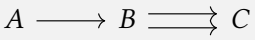


Pfeile verschieben

Die Optionen `shift=`*Koordinate*, `shift left=`*Länge*, `shift right=`*Länge*, `xshift=`*Länge* und `yshift=`*Länge* ermöglichen es Pfeile in geringem Maß zu verschieben. Somit ist es Beispielsweise möglich mehrere Pfeile in die gleiche Richtung zu setzen.

Die Standardwerte von `shift left` und `shift right` sind auch genau darauf ausgelegt;

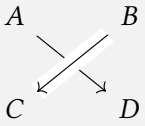
```
\begin{tikzcd}
    A \arrow[r]
    & B \arrow[r, shift left]
    \arrow[r, shift right]
    & C
\end{tikzcd}
```



Pfeile übereinander legen

Manchmal ist es notwendig, dass Pfeile sich kreuzen. Um den Eindruck zu zerstören, dass sich die Pfeile tatsächlich schneiden kann man sie übereinander legen. Dies geht mithilfe der Option `crossing over`. Setzt man diese Option geschickt ein, ist es auch möglich „dreidimensionale“ Diagramme zu erzeugen.

```
\begin{tikzcd}
    A \arrow[dr] & B \arrow[dl, crossing over] \\
    C
    & D
\end{tikzcd}
```



Literaturverzeichnis

[1] URL: http://detexify.kirelabs.org.

[2] Robert Bringhurst. *The elements of typographic style. Version 3.2.* 3. ed., expanded and rev. Point Roberts u.a.: Hartley & Marks, 2008. 382 S.

[3] David Carlisle. *The longtable package.* URL: http://truefunny.com/m/ctan/macros/latex/required/tools/longtable.pdf.

[4] Florian Rödl. *Einführung in LaTeX 2ε*. 2012.

[5] Martin Hensel. *The mhchem Bundle.* 16. Jan. 2017. URL: http://www.ctan.org/pkg/mhchem.

[6] Javier Bezos. *Customizing lists with the enumitem package.* URL: http://mirror.informatik.uni-mannheim.de/pub/mirrors/tex-archive/macros/latex/contrib/enumitem/enumitem.pdf.

[7] Donald Ervin Knuth. *The TeXbook.* [Repr.] Bd. A. Computers and typesetting. Reading, Mass: Addison Wesley und Addison-Wesley, 2006.

[8] Markus Kohm. *KOMA-Script: Eine Sammlung von Klassen und Paketen für LATEX 2e; Anleitung zu Version 3.13.* 5., überarb. und erw. Aufl. für Koma-Script 3. Berlin: Lehmanns, 2014.

[9] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script. Die Anleitung.* 2016. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/doc/scrguide.pdf (besucht am 16. 11. 2016).

[10] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script. The Guide.* 2016. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/doc/scrguien.pdf (besucht am 16. 10. 2016).

[11] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script: Eine Sammlung von Klassen und Paketen für LaTeX 2ε*. 4., überarb. und erw. Auflage für KOMA-Script 3. Lehmann, 2012.

[12] Helmut Kopka. *Einführung.* 1. Aufl., [unveränd. Nachdr.] Bd. 1. LaTeX. Bonn [u.a.]: Addison-Wesley, 1994.

[13] Helmut Kopka und Patrick W. Daly. *Guide to LaTeX: tools and techniques for computer typesetting.* 4. ed., 5. print. Addison-Wesley series on tools and techniques for computer typesetting. Boston, Mass. [u.a.]: Addison-Wesley, 2005.

[14] Martin Schröder. *The ragged2e-package.* URL: http://truefunny.com/m/ctan/macros/latex/contrib/ms/ragged2e.pdf.

[15] Frank Mittelbach und Michel Goossens. *The LaTeX companion.* 2nd ed. Boston [etc.]: Addison-Wesley, 2004.

[16] Marion Neubauer. „Feinheiten bei wissenschaftlichen Publikationen – Mikrotypographie-Regeln, Teil I“. In: *Die TeXnische Komödie* 1996.4 (1996), S. 23–40.

[17] Scott Pakin. *The Comprehensive LaTeX Symbol List.* 19. Jan. 2017. URL: http://www.ctan.org/tex-archive/info/symbols/comprehensive/#symbols-a4.pdf.

[18] Philipp Lehman. *The biblatex Package: Programmable Bibliographies and Citations*. URL: http://ftp.fau.de/ctan/macros/latex/contrib/biblatex/doc/biblatex.pdf.

[19] Sebastian Rahtz und Heiko Oberdiek. *Hypertext marks in LaTeX. a manual for hyperref*. 2012. URL: http://mirrors.ctan.org/macros/latex/contrib/hyperref/doc/manual.pdf (besucht am 10. 11. 2016).

[20] Robin Fairbairns. *footmisc: a portmanteau package for customising footnotes in LaTeX*. URL: http://ftp.fau.de/ctan/macros/latex/contrib/footmisc/footmisc.pdf.

[21] Kristoffer H. Rose. *Xy-pic User's Guide*. 6. Okt. 2013. URL: http://texdoc.net/texmf-dist/doc/generic/xypic/xyguide.pdf.

[22] Till Tantau. *The TikZ and PGF Packages: Manual for Version 3.0.0*. 2013. URL: http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf (besucht am 03. 08. 2015).

[23] Herbert Voß. *Einführung in LaTeX 2_ε: Unter Berücksichtigung von pdfLaTeX, XeLaTeX und LuaLaTeX*. 1. Aufl. Lehmanns Media, 2012.

[24] Joseph Wright. *siunitx: A comprehensive (SI) units package*. URL: http://ctan.space-pro.be/tex-archive/macros/latex/contrib/siunitx/siunitx.pdf.

Index

Sonderzeichen

*, Spaltendefinition 93
<{}, Spaltendefinition 95
>{}, Spaltendefinition 95

A

Abbildung . . . 91, *siehe auch* Gleitobjekt
Absatz
    Eingabe 8
    Kennzeichnung 17, **42**
Absatzbox 87
Abschnitt . . . . *siehe* Gliederungsebene
    Nummerierung . . . . . *siehe* Zähler
Abstand . . . . . *siehe* Zwischenraum
Akzent 27
    Mathemodus 123
amsfonts, Paket 118
amsmath, Paket 119–131
amsthm, Paket 131
Anführungszeichen 25
    deutsch 26
    englisch 26
Anhang 33
Argument 6
    optional 7
array, Paket 94
article . . . . . *siehe* scrartcl
Aufzählung 81–84
    Symbol 81
Ausrichtung . . . *siehe* Textausrichtung
.aux, Dateityp 5

B

b, Spaltendefinition 94
babel, Paket 23
Balkenbox 88
.bbl, Dateityp 108
Befehl . . . . . *siehe* Makro
Beweis (Mathematik) . . *siehe* Theorem
.bib, Dateityp 108
Biber 108
biblatex, Paket 108
BibTeX 108
Bild . . . . . *siehe* Abbildung
Bindekorrektur 17
Bindestrich 27, 47
Binomialkoeffizient 124
bm, Paket 119
Body . . . . . *siehe* Textkörper
book, Klasse . . . . *siehe* scrbook, Klasse
booktabs, Paket 96
Bruch (Mathematik) 124

C

C, Spaltendefinition 96
c, Spaltendefinition 93
chemische Gleichungen *siehe* Formelsatz
.cls, Dateityp 5, **13**
color, Paket 48
Counter . . . . . *siehe* Zähler
csquotes, Paket 26

D

Detexify 127
Distribution 3
Dokumentenklasse 13–20
    Optionsliste 15–19
doppelseitiger Druck 17
.dvi, Dateityp 5

E

Editor 3
Eingabekodierung . . . *siehe* Kodierung
Einheit 131
Einrückung 47
Einzug . . . . . *siehe* Absatz
Entwurfsmodus . . . . . 15, *siehe*
enumitem, Paket 83
Ergänzungspaket . . . . . *siehe* Paket
Exponent 123

F

Farbe . . . . . *siehe* color, Paket
fontenc, Paket 23
Formelsatz 117
  Chemie 134
  Referenz 122
  typografische Hinweise 117
Fußnote 58
Fußzeile . . . . . 73, *siehe* Seitenstil
  Anzahl 19
  Formatierung 76
  Trennlinie 19
Funktion (Mathematik) . *siehe* Operator (Mathematik)

G

Gedankenstrich
  deutsch . . . . *siehe* Halbgeviertstrich
  englisch . . . . . *siehe* Geviertstrich
geometry, Paket 70
Geviertstrich 27
Gleitobjekt 99–104
  Beschreibung
    Formatierung 103
    Position 102
  Verzeichnis 107
Gliederungsebene30, *siehe auch* Überschrift
  Nummerierung
    Punktierung 15
globale Option 20
Grafik . . . . . *siehe* Abbildung
griechische Buchstaben 127

H

Halbgeviertstrich 27
Hauptteil 30
Header . . . . . *siehe* Präambel
hebräische Buchstaben 127
Hervorhebung . *siehe* Textauszeichnung
Hilfsdatei 4–6
Hyperlink 60
hyperref, Paket 60

I

Index
  Verzeichnis 114
Index (Mathematik) 123
Inhaltsverzeichnis 33
inputenc, Paket 22

J

J, Spaltendefinition 96

K

Kapitel . . . . . *siehe* Gliederungsebene
Klammer (Mathematik) 128
Klasse . . . . . *siehe* Dokumentenklasse
Kodierung 22
  Schriftkodierung 23
KOMA-Script 13
kommutative Diagramme 135
Kompilieren 4
Kopfzeile . . . . . 73, *siehe* Seitenstil
  Anzahl 18
  Formatierung 76
  Höhe 18, 19
  Trennlinie 19

L

L, Spaltendefinition 96
l, Spaltendefinition 93
Länge 53
  Einheit 54
Leerseite . . . . . *siehe* Vakatseite
Leerzeichen
  geschützt 46
  im Code 8
Leerzeile . . . . . *siehe* Zwischenraum
  im Code 8
letter . . . . . *siehe* scrlettr2
Linie . . . . . *siehe* Balkenbox
linksbündig . . . . *siehe* Textausrichtung
Literaturverzeichnis 108
  manuell 108
  mit biblatex 108
lmodern 37
lmodern, Paket 23
.lof, Dateityp 5
.log, Dateityp 5
lokale Option 20
longtable, Paket 97
.lot, Dateityp 5
LR-Box 86

M

m, Spaltendefinition 94
makeidx, Paket 114
Makro 6
  definieren 56
  Struktur 6–7
  umdefinieren 56
Mathematiksatz . . . . *siehe* Formelsatz
  Struktur . . . . . *siehe* Theorem
  Text 130
Matrix (Mathematik) 129
mehrspaltiger Textsatz 70
  zweispaltig 18
mhchem, Paket 134
microtype, Paket 25
MikTEX . . . . . *siehe* Distribution
multicol, Paket 70
multirow, Paket 94

N

Nachspann 30
Neunerteilung 67
Nummerierung . . . . . *siehe* Zähler

O

Operator (Mathematik) 125
  definieren 127
Optionen
  global vs. lokal 20

P

p, Spaltendefinition 93
Paket 20
  Anleitung 20
Papierformat 15, 70
pdflatex 4
placeins, Paket 101
Präambel 9

Q

Querverweis 58

R

R, Spaltendefinition 96
r, Spaltendefinition 93
ragged2e, Paket 47
Rahmen 86
Rand . . . . . *siehe* Satzspiegel
Randnotiz 58
Reader 3
rechtsbündig . . . *siehe* Textausrichtung
report . . . . . *siehe* scrreprt

S

S, Spaltendefinition 133
Satzspiegel 67–70
Schrift 37–42
  -attribut (Mathemodus) 118
  Schriftart 37
  Schriftattribut 37
  Schriftgröße
    global 17
    lokal 39, 40
  Stil (Mathemodus) 119
Schriftkodierung . . . . *siehe* Kodierung
scrartcl, Klasse 14
scrbook, Klasse 14
scrlettr2 14
scrreprt, Klasse 14
Seitenlayout *siehe auch* Satzspiegel, *siehe* Seitenstil
Seitennummerierung 52
Seitenstil 73
Seitenumbruch 44–45
setspace, Paket 44
siunitx, Paket 131
Sonderzeichen 6, 8
  fremdsprachig 27
Sonderzeichen (Mathematik) 127
Sprachanpassung 22
Sprache . . . . . *siehe* babel
.sty, Dateityp 5

T

Tabelle . . . . 93, *siehe auch* Gleitobjekt
  Linie 93, 96
  Spaltenformatierung 93
  Spaltenzwischenraum 93
tabularx, Paket 95
tabulary, Paket 96
Teilungsverhältnis . . . *siehe* Satzspiegel
.tex, Dateityp 5
TEX Live . . . . . *siehe* Distribution

TeXmaker . . . . . *siehe* Editor
TeXstudio . . . . . *siehe* Editor
Textausrichtung 47
Textauszeichnung 39
Textkörper 9
Theorem 131
tikz-cd, Paket 135
Titelei 15, **28**
Titelseite . . . . . *siehe* Titelei
.toc, Dateityp 5
Trennung 45
typearea, Paket 67

U

Überschrift *siehe auch* Gliederungsebene
  Formatierung ändern 40
  Schriftgröße 15, 16
Umgebung 7
Umlaute 22
Untergliederung *siehe* Gliederungsebene
Untersteichung 40

V

Vakatseiten 18
  Kapitelanfang 16
Verzeichnis 107
Von–bis-Strich . . *siehe* Halbgeviertstrich
Vorspann 30

W

Wert 131
wrapfig, Paket 104
Wurzel (Mathematik) 125

X

X, Spaltendefinition 95

Z

Zähler 51
Zeilenabstand 44
Zeilenumbruch 43–44
  im Code 8
zentriert . . . . . *siehe* Textausrichtung
Zitat 25
Zusammenfassung 16, 32
Zwischenraum 55

Index der Makros, Umgebungen und Optionen

Sonderzeichen

\! 56
"| 46
"" 46
"' 26
"- 46
"< 26
"> 26
"` 26
' 26
'' 26
\, ... 46, 56
- 27, 47
-- 27
--- 27
\/ 39
\: 56
\; 56
<< 26
>> 26
\@ 24
@{} 94
. 9
\# 9
$ 9
$-Umgebung 120
\$ 9
% 9
\% 9
& 9, 93
\& 9
\- 46
« 26
» 26
‹ 26
› 26
„ 26
‚ 26
“ 26
” 26
‘ 26
’ 26
_ 9
_ 9
^ 9
~ 9, 44
\\ 43
\ 9
\[-Umgebung 120
\(-Umgebung 120
"= 47
"~ 47
\{ 9
{ 9
\} 9
} 9
‚ 26
‘‘ 26
~ 46

A

abstract=, Option 16
\addchap 32
\addcontentsline 107
addmargin-Umgebung 47
\addpart 32
\addsec 32
\addtocounter 51
\addtokomafont 40
\addtolength 53
align-Umgebung 120
alignat-Umgebung 120
aligned-Umgebung 121
\Alph 51
\alph 51
\appendix 33
\arabic 51
\areaset 69

array-Umgebung 93
\author 29
\automark 76
\autoref 62

B

\backmatter 30
\baselineskip, Länge 53
BCOR=, Option 17, 68
\bfseries 38
\bibitem 108
\bibliography 109
\bibname 107
\bigskip 56
\binom 125
\bm 119
\boldmath 118
\boldsymbol 118
\bottomfraction 102
bottomnumber, Zähler 102
\bottomrule 96

C

\caption 99
captionbeside-Umgebung 102
captions=, Option 19
cases-Umgebung 128
\ce 134
Center-Umgebung 48
center-Umgebung 48
\Centering 48
\centering 48
\chapter 30
\chapternumdepth 52
\chapterpagestyle 73
\cite 109
\cites 112
\cleardoublepage 44
cleardoublepage=, Option 18
\clearmainofpairofpagestyles . . 75
\clearpage 44
\clearpairofpagestyles 75
\clearplainofpairofpagestyles . 75
\cline 94
clines, Option 19
\cmidrule 96
\cmidrulewidth, Länge 97
\color 48
\colorbox 48

D

\date 29
dbltopnumber, Zähler 102
\DeclareMathOperator 127
\dedication 29
\deffootnote 59
\deffootnotemark 59
\definecolor 48
description-Umgebung 81
\dfrac 125
displaymath-Umgebung 120
DIV=, Option 17, 68
document-Umgebung 9
\documentclass 13
\dotfill 56
\doublespacing 44
draft=, Option 15

E

\emph 39
english, Option . . *siehe* globale Option
\enlargethispage 45
\enquote 26
\ensuremath 120
enumerate-Umgebung 81
enumi, Zähler 51
\eqref 122
equation, Zähler 51
equation-Umgebung 120
\evensidemargin, Länge 53
\extratitle 29

F

\fbox 86
\fboxrule, Länge 86
\fboxsep, Länge 86
\fcolorbox 48
figure, Zähler 51
figure-Umgebung 99
\figurename 107
flalign-Umgebung 120
\FloatBarrier 101
\floatpagefraction 102
\flq 26
\flqq 26

\flushbottom 69
FlushLeft-Umgebung 48
flushleft-Umgebung 48
FlushRight-Umgebung 48
flushright-Umgebung 48
\fnsymbol 51
fontsize=, Option 17
\footheight, Länge 74
footheight=, Option 19
footinclude=, Option 69
footlines=, Option 19
\footnote 58
footnote, Zähler 51
\footnotemark 59
footnotes=, Option 18
\footnotesize 40
\footnotetext 59
\footref 59
footsepline=, Option 19
\footskip, Länge 53
\foreignlanguage 24
\frac 124
\frame 86
\framebox 86
\frenchspacing 24
\frontmatter 30
\frq 26
\frqq 26

G

gather-Umgebung 121
gathered-Umgebung 121
\genfrac 124
\glq 26
\glqq 26
\grq 26
\grqq 26

H

\headheight, Länge 74
headheight=, Option 18
headinclude=, Option 69
headings=, Option 15, 16, 31
headlines=, Option 18
\headmark 76
\headsep, Länge 53
headsepline=, Option 19
\heavyrulewidth, Länge 97
\hfill 55
\hline 93
\hrulefill 56
\hspace 55
\Huge 40
\huge 40
\hyphenation 46

I

ilines, Option 19
\include 34
\includegraphics 91
\includeonly 34
\indent 43
\index 114
\indexpagestyle 73
\input 33
\int 125
\intertext 130
\item 81
itemize-Umgebung 81
\itshape 38

L

\labelenumi 81
labeling-Umgebung 81
\labelitemi 81
\LARGE 40
\Large 40
\large 40
\left 128
\leftmark 75
\lightrulewidth, Länge 97
\lim 125
\limits 126
\linebreak 43
\listfigurename 107
\listoffigures 107
\listoftables 107
\listtablename 107
\lowertitleback 29
lrbox-Umgebung 89

M

\mainmatter 30
\makebox 86
\makeindex 114

\maketitle 29
\manualmark 76
\marginline 60
\marginpar 58, 59
\markboth 73
\markright 73
math-Umgebung 120
\mathbb 118
\mathbf 118
\mathcal 118
\mathfrak 118
\mathit 118
\mathnormal 118
\mathrm 118
\mathsf 118
\mathtt 118
\mbox 86
\mdseries 38
\medskip 56
\midrule 96
minipage-Umgebung 87
\minisec 32
multicols-Umgebung 70
\multicolumn 94
\multirow 94

N

\newcolumntype 95
\newcommand 57
\newcounter 52
\newenvironment 58
\newlength 54
\newline 43
\newpage 44
\newsavebox 88
\newtheorem 131
ngerman, Option . . *siehe* globale Option
\nocide 109
\nocite 109
\noindent 43
\nolimits 126
\nolinebreak 44
\nonfrenchspacing 24
\nopagebreak 45
\normalcolor 48
\normalfont 38
\normalsize 40
\num 132
numbers=, Option 15, 32

O

\oddsidemargin, Länge 53
olines, Option 19
\onecolumn 70
\onehalfspacing 44
open=, Option 16
\operatorname 127
\overbrace 129
\overset 130

P

page, Zähler 51
\pagebreak 44
\pagemark 76
\pagenumbering 52
\pageref 58
\pagestyle 73
paper=, Option 15
\paperheight, Länge 53
\paperwidth, Länge 53
\paragraph 30
\paragraphnumdepth 53
\parbox 87
\parindent, Länge 53
\parskip, Länge 53
parskip=, Option 17, 42
\part 30
\partnumdepth 52
\partpagestyle 73
\phantom 122
\printbibliography 109
\printindex 114
\prod 125
proof-Umgebung 131
\publishers 29

Q

\qquad 56
\quad 56
quotation-Umgebung 47
quote-Umgebung 47

R

\raggedbottom 69
\RaggedLeft 48

\raggedleft 48
\RaggedRight 48
\raggedright 48
\raisebox 86
\recalctypearea 69
\ref 58
\refname 107
\renewcommand 57
\renewenvironment 58
\right 128
\rightmark 75
\rmfamily 38
\Roman 51
\roman 51
\rule 88

S

samepage-Umgebung 45
\savebox 89
\sbox 89
\scriptsize 40
\scshape 38
secnumdepth, Zähler 51
\section 30
\sectionnumdepth 52
\selectlanguage 24
\setcaphanging 103
\setcapindent 103
\setcapmargin 104
\setcapwidth 104
\setcounter 51
\setfootnoterule 59
\setkomafont 40
\setlength 53
\settoheight 54
\settowidth 54
\sffamily 38
\si 132
\singlespacing 44
\sisetup 132
\slshape 38
\small 40
\smallskip 56
\sqrt 125
\stepcounter 51
\subject 29
\subparagraph 30
\subparagraphnumdepth 53
\subsection 30
\subsectionnumdepth 52
\substack 126
\subsubsection 30
\subsubsectionnumdepth 52
\subtitle 29
\sum 125
\suppresfloats 101

T

T1, Option . . . . . *siehe* fontenc, Paket
\tabcolsep, Länge 53
table, Zähler 51
table-Umgebung 99
\tablename 107
\tablenum 133
\tableofcontents 33
tabular-Umgebung 93
\tabularnewline 93
tabularx-Umgebung 95
tabulary-Umgebung 96
\tag 122
\text 130
\textasciicircum 9
\textasciitilde 9
\textbackslash 9
\textbf 38
\textcolor 48
\textfraction 102
\textheight, Länge 53
\textit 38
\textmd 38
\textnormal 38
\textrm 38
\textsc 38
\textsf 38
\textsl 38
\textsubscript 39
\textsuperscript 39
\texttt 38
\textup 38
\textwidth, Länge 53
\thanks 29
\the*Countername* 52
thebibliography-Umgebung 108
\thefootnotemark 59
\theoremstyle 131
\thispagestyle 73

tikzcd-Umgebung 135
\tiny 40
\title 29
\titlehead 29
titlepage=, Option 15
titlepage-Umgebung 29
\titlepagestyle 73
toc=, Option 16, 34
tocdepth, Zähler 51
\topfraction 102
\topmargin, Länge 53
topnumber, Zähler 102
\toprule 96
\topskip, Länge 53
totalnumber, Zähler 102
\ttfamily 38
\twocolumn 70
twocolumn=, Option 18
twoside=, Option 17
\typearea 69

U

\unboldmath 119
\underbrace 129
\underline 40
\underset 130
\uppertitleback 29
\upshape 38
\usebox 89
\useencodingofkomafont 42
\usefamilyofkomafont 42
\usefontofkomafont 42
\usekomafont 40
\usepackage 20
\useseriesofkomafont 42
\useshapeofkomafont 42
\usesizeofkomafont 42
utf8, Option . . . . *siehe* inputenc, Paket

V

\value 51
\verb 49
verbatim-Umgebung 48
\vline 94

W

wrapfigure-Umgebung 104
wraptable-Umgebung 104