PeiTEX TEXnical Solutions
Marei Peischl
Gneisenaustraße 18
20253 Hamburg
https://peitex.de

Einführung in LaTeX 2ε

unter Berücksichtigung von KOMA-Script

Bearbeitungsstand 2023-04-18 (Commit: 3ec0924, kompiliert am 19. April 2023)

Über dieses Dokument

Das vorliegende Dokument basiert auf unterschiedlichen Kursen sowie Schulungen und wird seit mehr als 10 Jahren kontinuierlich verbessert. Es ist historisch gewachsen und wird im aktuellen Status veröffentlicht.

Dieses Projekt wird wahrscheinlich nie eine endgültige Fassung erlangen. Das liegt auch daran, dass LaTeX sowie die behandelten Pakete stetig weiterentwickelt werden.

Bei Unstimmigkeiten bin ich für jede Hilfe dankbar, also meldet euch gerne, falls ihr unschlüssige Punkte oder auch einfach nur Schreibfehler findet. Hierfür bin ich über marei@peitex.de erreichbar. Vielen Dank!

Aufgrund der besseren Anschaulichkeit werden in Beispielen an einigen Stellen Makros verwendet, die im vorherigen Verlauf noch nicht besprochen wurden. Diese sind zum Teil nicht gesondert gekennzeichnet, um das Beispiel nicht noch komplizierter zu gestalten. Die Bedeutung der Makros (auch wenn diese für das Beispiel nicht von Bedeutung sein sollten) findet ihr über den Index am Ende des Skriptes. In Schulungen oder Workshops wird explizit darauf hingewiesen und die Funktion kurz erläutert.

Hamburg, im April 2023
Marei Peischl

Inhaltsverzeichnis

Was ist LaTeX?

TeX[1] (gesprochen „tech“) wurde ab 1977 „zum Satz schöner Bücher und insbesondere Bücher, die viel Mathematik enthalten“ [11] von Donald E. Knuth, einem mittlerweile emeritierten Professor der Stanford University, erschaffen. Da sich die Benutzung als recht schwierig herrausstellte entwickelte der Mathematiker, Informatiker und Programmierer Leslie Lamport in den 1980er Jahren das Makropaket LamportTeX, welches abgekürzt LaTeX genannt wird.

LaTeX ist eine Sammlung von Makros für das Textsatzprogramm TeX. Es greift mithilfe dieser Makros auf die Deklarationen in TeX zurück und vereinfacht damit die Anwendung.

Seit 1993 bearbeitet ein Team namens LaTeX3 Project[2] die jetzige Version und erweitert kontinuierlich den Funktionsumfang auch auf Basis existierender Ergänzungspakete.

Da LaTeX wie auch reines TeX mathematische Notationen unterstützt, ist es im naturwissenschaftlichen und technischen Umfeld am verbreitetsten. Es ist allerdings für die verschiedensten Dokumententypen geeignet und ermöglicht sehr viel Automatisierung und langfristige Arbeitserleichterung bei der Umsetzung von Layouts. Dies kann je nach Fachbereich sehr unterschiedliche Vorteile haben.

Word vs. LaTeX – Die Philosophie

Textverarbeitungsprogramme gliedern sich in verschiedene Sparten:

Wortprozessoren (z. B. Microsoft Word) Der Text wird direkt in einer grafischen Nutzeroberfläche bearbeitet. Eine Formatierungseigenschaft oder Formatvorlage wird für einen Textbereich ausgewählt und angewendet. Oftmals auch Textverarbeitungssysteme genannt, allerdings ist diese Bezeichnung im Vergleich zu den anderen Programmtypen irreführend.

Desktop-Publishing-Systeme (z. B. Adobe InDesign) Hier existieren 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* eigene grafische Oberfläche. Es gibt natürlich auch Systeme die über eine verfügen, allerdings ist diese nicht Teil des Konzeptes. Es wird ein Quelltext (Code) bearbeitet, welcher neben dem Inhalt auch die Dokumentenstruktur enthält. Dieser wird vom Programm interpretiert und erzeugt entsprechend der Layouteinstellungen eine Ausgabedatei (üblicherweise eine `.pdf`-Datei).

[1] Großschreibung von τεχ; Tau Epsilon Chi. Kommt vom altgriechischen τέχνη und bedeutet in etwa Fähigkeit, Kunst, Handwerk.

[2] https://www.latex-project.org/

Die Konzepte haben alle ihre eigenen Vor- und Nachteile, sodass die Wahl der Software letztendlich bei einem selber liegt. Den wesentlichsten Unterschied stellt die Beziehung zwischen Dokumenteninhalt und Formatierung dar. Bei TEX-basierten Systemen erhält der Inhalt ein sogenanntes „Markup“, eine semantische Auszeichnung. Das heißt, dass im Code selbst nicht direkt angegeben wird wie ein Element formatiert werden soll, stattdessen erhält der ausgewählte Textbereich eine Kennzeichnung. Eine Textzeile kann somit beispielsweise als Fließtext, Überschrift, Beschriftung, etc. markiert werden. Das Layout wird beim Übersetzen in eine `.pdf`-Datei den Einstellungen entnommen.

Diese Vorgehensweise ermöglicht ein einheitliches Layout, ohne das man viele Anpassungen vornehmen muss. Elemente des gleichen Typs werden für das gesamte Dokument gleich behandelt und wirken dadurch professionell, verfügen über eine klare Struktur und sind für den Leser leichter nachvollziehbar. Auch Formatierungsänderungen und die Wiederverwendbarkeit einzelner Elemente sind nachträglich jederzeit problemlos möglich.

Das System hat zudem den Vorteil, ein Layout auf verschiedene Dokumente zu übertragen. Durch das Verändern der Einstellungen lässt sich aus demselben Code eine Präsentation, ein gedrucktes Skript oder eine Aufgabenstellung in verschiedenen Varianten erzeugen.

LATEX stellt einige Basislayouts für die geläufigsten Dokumententypen bereit. Diese erfüllen sämtliche Ansprüche für einfache Text-Dokumente und sind durch andere Layouts und Ergänzungspakete erweiterbar. Komplexere Dokumente, die z. B. mathematische Formalismen oder aufwändige Tabellen beinhalten sind mit deutlich weniger Aufwand realisierbar, als dies bei 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 auf vorherige Versionen zurückzuspringen und Vergleiche zwischen diesen durchzuführen.

Bei LATEX wird, wie in der Softwareentwicklung häufig, das Versionsverwaltungssystem git[3] eingesetzt. Die Verwendung solcher Systeme vereinfacht zwar vieles, funktioniert jedoch unabhängig von LATEX und wird daher in diesem Dokument nicht weiter behandelt.

[3] https://git-scm.com/

Wichtige Hinweise bevor es los geht

Verwendung von Vorlagen

Es gibt jede Menge Vorlagen für LaTeX Dokumente. Die Vielfalt ist jedoch so groß, dass sich darunter auch viel Veraltetes oder gar Unbrauchbares befindet. Wenn man mit einer Vorlage arbeiten möchte, sollte man sich immer erstmal die geladenen Pakete in dieser anschauen. Dabei ist wichtig zu gucken, welchen Verwendungszweck diese haben, ob sie wirklich benötigt werden und ob sie aktuell sind.

Außerdem sollte man darauf achten, dass bei den System- und Sprachanpassungen keine anderen Optionen verwendet werden, als die in diesem Skript behandelten. In Abschnitt 3.4 wird noch genauer auf die Sprachanpassungen eingegangen. Vorlagen, die diesen Ansprüchen nicht genügen, können zumeist als veraltet angesehen werden.

Falls bei der Frage, ob die Vorlage brauchbar ist, Unsicherheit besteht hilft peiTeX gerne bei der Einschätzung. Genauso natürlich auch bei der Unterstützung der Aktualisierung oder sonstigen Anpassungen und Fragen/Problematiken rund um das Thema LaTeX.

Umgang mit Fehlermeldungen

In den Standardeinstellungen vieler TeX-Editoren ist meistens der „nonstopmode“ aktiviert. Das bedeutet, dass bei einem Fehler das Programm trotzdem weiterläuft und nicht anhält. Diese Standarteinstellung führt dazu, dass beim Kompiliervorgang trotz auftretender Fehler eine auf den ersten Blick zufriedenstellende Ausgabe erzeugt wird, was wiederum dazu verleiten kann, dass Fehlermeldungen ignoriert werden. *Genau davon sollte in jedem Fall abgesehen werden!*

Fehlermeldungen geben Auskunft, wenn der Programmdurchlauf nicht wie erwartet verlaufen ist und irgendetwas, sei es auch noch so eine Kleinigkeit, nicht sauber funktioniert hat. Tritt zum Beispiel ein Fehler an einer Stelle auf, dann kann das unteranderem zu „komischem“ Verhalten an ganz anderer Stelle im Dokument führen. Im schlimmsten Fall kann sogar Dokumenteninhalt „verschluckt“ werden.

In den Programmeinstellungen kann man diese Einstellung ändern, indem man beim Aufruf von `pdflatex` die Option `-interaction=nonstopmode` entfernt.

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 und 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.

Teil I

TEXnische Grundlagen

1 Die grundlegende Funktionsweise

1.1 Komponenten eines LaTeX-Systems

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

Einen Editor Hier wird der Text mit den dazugehörigen Formatierungsbefehlen eingegeben. Es genügt ein einfacher Texteditor wie z. B. Notepad++, gedit oder ähnliche.

Für Anfänger:innen ist es allerdings einfacher, einen speziellen LaTeX-Editor zu nutzen, da dieser unter Anderem auch bei der Fehlersuche hilft. Beispiele für kostenfreie Editoren sind TeXstudio, TeXmaker und TeXworks. Sie stehen für alle gängigen Betriebssysteme zur Verfügung. Für den Anfang empfiehlt sich TeXstudio[1], da dieser die meisten Unterstützungsfunktionen bietet.

Anmerkung für Leser:innen mit Programmiererfahrung: Wenn ihr eine Lieblings-IDE habt, dann existieren dafür entsprechende Plugins um die LaTeX-Unterstützung zu aktivieren. In diesen Fällen ist kein gesonderter Editor notwendig.

Eine Distribution Die Distribution ist der Kern eines TeX-Systems und enthält das eigentliche Übersetzungsprogramm. Der Übersetzer wandelt den Quellcode unter Berücksichtigung der Formatierungsbefehle und Formatdateien in ein Ausgabedokument (meistens eine `.pdf`-Datei) um. Diesen Prozess nennt man Kompilierung und die durchführenden Programme Kompilierungsprogramme oder Compiler.

Für die Nutzung von LaTeX existieren unterschiedliche Kompilierungsprogramme. Die häufigsten sind pdfLaTeX, XeLaTeX und LuaLaTeX. Der wesentliche Unterschied besteht darin welche Schrifttypen und wie Sonderzeichen verarbeitet werden. Dieses Skript bezieht sich (wenn nicht anders beschrieben) auf LuaLaTeX. Allerdings funktionieren viele Beispiele auch mit den anderen Programmen. Bei der Verwendung von Vorlagen kann es jedoch dazu kommen, dass sich diese explizit auf einen anderen Compiler beziehen. In diesem Fall ist es hilfreich sich der Existenz anderer Varianten bewusst zu sein.

Da LaTeX für sehr viele und auch sehr unterschiedliche Bereiche verwendet wird, kann es passieren, dass eine Vorlage nicht genau auf die Bedürfnisse der Nutzer:innen abgestimmt ist. In diesem Fall gibt es in LaTeX unterschiedlichste Zusatzpakete und auch kleinere Zusatzprogramme, welche die Funktionalität erweitern und nutzerfreundliche Schnittstellen für nahezu jedes Anwendungsgebiet ermöglichen.

[1] https://www.texstudio.org/

Die beiden üblichsten Distributionen (die für alle gängigen Betriebssysteme verfügbar sind) heißen TEX Live (für Mac OS: MacTEX) und MiKTEX. Die Basisfunktionität ist bei beiden gleich, allerdings gibt es geringfügige Unterschiede in den Lizenzen und dadurch enthaltenen Paketen.

Viele Linux-Distributionen verfügen über eigene Pakete für TEX Live. Diese sind häufig jedoch nicht auf dem neuesten Stand. Es ist daher abzuwägen, ob einem diese Pakete genügen oder ob man aktuellere nutzen möchte. In letzterem Fall besteht immer die Möglichkeit eine Installation aus den CTAN[2]-Resourcen durchzuführen [26, siehe].

Einen Betrachter Er wird benötigt, um das erzeugte Dokument ansehen bzw. ausdrucken zu können. Bei `.pdf`-Dateien stehen da zum Beispiel der Adobe Acrobat Reader oder Okular zur Verfügung. Bei einigen Editoren, wie TEXstudio oder TEXmaker ist ein solches Programm bereits integriert. Diese eingebetteten Versionen eignen sich jedoch meistens nur für eine grobe Kontrolle, denn es fehlen häufig Details in der Anzeige, wie beispielsweise dünne Linien. Für eine abschließende Endkontrolle sollten sie nicht verwendet werden.

Allgemein verfügt das `.pdf`-Format über deutlich mehr Funktionalität, als nur die bloße Dokumentendarstellung. Es ist sogar möglich Videos und Animationen einzubinden oder interaktive Formulare in diesem Format zu erzeugen. Nicht alle Anzeigeprogamme unterstützen allerdings sämtliche Features.

1.2 Vom Quellcode zum Dokument

Diese erwähnten drei Programme werden nun genutzt, um ein Dokument zu erstellen. Zuerst wird der Code im Editor geschrieben und gespeichert. Danach übernimmt das Kompilierungsprogramm und erstellt, wenn keine Fehlermeldung auftritt, eine im Betrachter sichtbare PDF-Ausgabe.

Quellcode Beginnen tut alles mit dem Quellcode. Dieser ist in den Quelldateien enthalten und hat bei LATEX üblicherweise die Dateiendung `.tex`. Zu ihm gehört alles, was man zuvor im Editor verfasst hat. Der einfachste Quellcode, um ein funktionierendes Dokument zu erstellen, sieht in LATEX etwa so aus:

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

Je größer und komplexer das Dokument wird, umso größer und komplexer wird natürlich auch der Quellcode. Aus welchen Komponenten der Code genau bestehen muss und wie er entsteht wird im nächsten Teil des Skriptes erklärt.

[2] Das Comprehensive TEX Archive Network (CTAN) ist eine Plattform für die Veröffentlichung von Materialien rund um TEX/LATEX.

PDF-Ausgabe erzeugen Wurde dieser Quellcode gespeichert, übernimmt das Satzprogramm und interpretiert diesen. Es ruft gegebenenfalls auch weitere externe Programme auf, die zum Beispiel bei der Erstellung einer Bibliografie notwendig sind. Abschließend kann es im Betrachter eine PDF-Ausgabe erstellen. Dies bedeutet, dass aus dem eben gezeigten Quellcode nun Folgendes entsteht:

Mein erstes Dokument!

Falls man mit einem speziellen LaTeX-Editor arbeitet, geschieht dies meistens über einen einzigen Tastendruck. Allerdings ist es nach wie vor möglich, direkt über eine Kommandozeile zu kompilieren. Die notwendigen Schritte dafür 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 `lualatex`: In der Kommandozeile[3] an den Speicherort der Datei navigieren und dann `lualatex` ausführen:

   Kommandozeile

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

   Dabei werden das `.pdf`-Dokument und einige zusätzliche Hilfsdateien erzeugt.
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 3.8).
4. *Optional:*
   Je nachdem, welche Zusatzelemente verwendet werden (Bibliografie, Stichwortverzeichnis, …) ist es notwendig, mehrere Kompilierungsvorgänge durchzuführen und ggf. auch weitere Programme zu starten (z. B. um die Literaturangaben zu sortieren). Wenn diese Elemente besprochen werden, wird auch auf diesen Punkt genauer eingegangen.
5. Das fertige `.pdf`-Dokument ansehen.

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

[3] Je nach Betriebssystem wird das Programm auch *Eingabeaufforderung* (Windows) oder *Terminal* (Mac OS oder Linux) genannt.

Tabelle 1.1: Die wichtigsten Hilfsdateien und ihre Bedeutung der Dateiendungen (je nach Konfiguration werden ggf. zusätzliche Dateien erstellt)

Endung	*Erläuterung*
`.tex`	LaTeX oder TeX Input-Format und kann kompiliert werden
`.sty`	LaTeX-Style: Ergänzungspaket; erweitert die Funktionalität und kann mit `\usepackage` geladen werden, siehe Abschnitt 3.3
`.cls`	LaTeX-Class: Dokumentenklasse, welche den Dokumententyp festlegt, siehe Abschnitt 3.1
`.dvi`	Device independent file format: Ausgabeformat der ursprünglichen Programmvariante `latex`
`.log`	Protokoll: Warnungen und Fehlermeldungen sowie detaillierte Informationen über den letzten LaTeX-Durchlauf
`.toc`	Speicherung aller 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

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

Hilfsdateien Bei der Umwandlung in das Ausgabeformat, also zum Beispiel in das PDF-Dokument, wird in der Regel mehr als nur die Ausgabedatei erstellt. Diese zusätzlich erzeugten Dateien sind Hilfsdateien. In Tabelle 1.1 findet sich eine Auflistung mit genaueren Erläuterungen zu den Dateitypen. Diese Dateien werden entweder vom Compiler für die Erstellung von Querverweisen und Verzeichnissen oder von zusätzlichen Programmen benötigt. 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.

1.3 Befehlsstruktur

Einzelne Elemente werden durch Befehle als solche gekennzeichnet. Bei LaTeX werden diese Befehle Makros genannt. 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 Befehl (Auflistung in Tabelle 1.2).

Sonderzeichen hinter Backslash Einzelne Sonderzeichen hinter einem Backslash (vgl. ebenfalls Tabelle 1.2). Diese Makros werden meistens zum Ausdruck der Sonderzeichen selber genutzt. Zum Beispiel wird dann mit `\$` das Zeichen $ im Text dargestellt.

Backslash mit Buchstabenfolge Klassische Makros bestehen aus einem Backslash und einer Buchstabenfolge. Das erste Zeichen, welches dem Befehl dann folgt, wird als Befehlsende interpretiert.

`\Befehl`

Dies ist die einfachste Form von einem Befehl. Dieser wird mit \ eingeleitet und benötigt keine weiteren Argumente. Die Struktur und Verwendung wird im Folgenden genauer erläutert. Ein Beispiel hierfür ist das LaTeX-Logo: `\LaTeX` wird zu LaTeX

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

1.3.1 Makros

Befehle mit Argumenten ermöglichen es, Makros an die genaue Verwendung anzupassen oder die Wirkung des Befehls auf einen kurzen Textabschnitt einzugrenzen.

`\Befehl{`*Argument*`}`

Das Argument wird durch geschweifte Klammern begrenzt und darf nicht weggelassen werden. Bei den meisten Makros mit Argumenten ist das auch nachvollziehbar, da diese ohne das Argument keinen Sinn ergeben.
Für das folgende Beispiel wird das Makro `\color{Farbe}` verwendet. Zusätzlich wird noch das Paket color oder xcolor benötigt. Genaueres dazu siehe auch Abschnitt 5.2.

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

Dieser Text ist grau.

Ohne eine Angabe der Farbe wäre der Befehl `\color` sinnlos. Außerdem erkennt man, dass der Befehl `\color{gray}` wie ein Schalter wirkt und alles was danach folgt grau geschrieben wird.

Um die Wirkung eines solchen Schalters einzugrenzen können geschweifte Klammern, die eine sogenannte Gruppe bilden, verwendet werden:

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

Dieser Text ist grau.
Dieser Text ist schwarz.

Weitere Informationen zu Gruppen finden sich am Ende dieses Kapitels in Abschnitt 1.4.

`\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[4]:

[4] 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).

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

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

`$\sqrt{2}$` liefert hierbei das gleiche Ergebnis wie `$\sqrt[]{2}$`. Allgemein hängt das Verhalten jedoch vom bei der Definition hinterlegten Säumniswert ab (vgl. 4.3). Dieser wird verwendet, wenn kein Argument übergeben wird. Bei `\sqrt` ist der voreingestellte Wert leer.

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

Neben den bisherigen Argumententypen verfügen einige Makros noch über Sternchen-Versionen. Das Sternchen agiert ebenso als Schalter. Es aktiviert zusätzliche Funktionen oder deaktiviert Teile der Funktionalität. Was der Schalter bewirkt hängt vom jeweiligen Befehl ab. In diesem Dokument wird bei der Erläuterung der Funktionalität eines Befehls mit Sternchenargument auch zusätzlich erklärt, was das Sternchen für Auswirkungen hat.

Das häufigste Beispiel sind hierbei die Makros der Gliederungsebenen (siehe Abschnitt 3.7):

```
\section[Verzeichniseintrag]{Überschrift}
\section*{Überschrift}
```

Bei den Erläuterungen der Makros wird nun unterschieden, ob beide Varianten unterschiedliche Argumente verarbeiten oder die Konfiguration gleich bleibt.

\MakroEins*{*Argument*}

in diesem Fall hätten sowohl `\MarkoEins` als auch `\MakroEins*` nur ein notwendiges Argument.

\MakroZwei[*optionales Argument*]{*Argument*}
\MakroZwei*{*Argument*}

In diesem Fall sind die beiden Varianten zwei unterschiedliche Befehle, die auch unterschiedliche Argumente verarbeiten können. `\MakroZwei` verfügt über ein optionales Argument, dass `\MakroZwei*` nicht verarbeiten kann.

1.3.2 Umgebungen

Umgebungen funktionieren wie Befehle und können ebenfalls notwendige sowie optionale Argumente besitzen.

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

Umgebungen werden entweder zur Abgrenzung spezieller Inhalte vom Fließtext oder zur Eingrenzung von Einstellungen genutzt. Dies ist auch mithilfe von Gruppierungen möglich (siehe auch Abschnitt 1.4).

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.

Umgebungen haben ähnlich zu den Sternchenversionen bei Makros auch Sternchenvarianten. In diesem Fall wird das Sternchen allerdings nicht als Argument, sondern als Teil des Umgebungsnamens übergeben.

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

1.4 Eigenheiten von TEX

Manche Eingaben werden im Code (also auch bei LATEX) etwas anders interpretiert, als man es oft von Textverarbeitungssoftware gewohnt ist. Das liegt unteranderem auch daran, dass bestimmte Sonderzeichen als Steuerzeichen genutzt werden. Eine frühzeitige Betrachtung ist hier hilfreich, da dies bei Ignorierung oft zu Verwunderung und der Suche nach Fehlern führt.

Falls also die Software während der ersten Schritte anders reagiert als erwartet, hilft es sich die Bedeutung der folgenden Steuerzeichen und Mechanismen wieder ins Gedächtnis zu rufen:

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

Leer- oder Tabulatorzeichen am 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 abgebildet und können bei Nichtbeachtung zu fehlenden Leerzeichen führen z. B.:

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

Dieser Satz wurde mit LATEXgesetzt.

Abhilfe kann man schaffen in dem man entweder das Makro durch eine leere Gruppe ({}) beendet oder das Leerzeichen explizit eingibt (\):

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`
%	Einleitung von Kommentar	`\%`

```
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 (siehe vorheriges Beispiel). 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 hintereinander auf, so werden diese wie eine Einzelne behandelt.

Die Methode der Absatzkennzeichnung sowie gegebenenfalls auch Absatzabstände werden global eingestellt. Die Möglichkeiten und Unterschiede zwischen Absatz- und Zeilenumbrüchen werden in Unterabschnitt 5.4.1 erläutert.

Sonderzeichen Alle Zeichen die eine spezielle Eigenschaft haben (wie z.B. % als Einleitung eines Kommentars) müssen, wenn sie als Text ausgegeben werden sollen, über einen Befehl (in diesem Falle \%) angesprochen werden.

Tabelle 1.2 zeigt die Sonderzeichen, welche als Befehl interpretiert werden samt ihren Bedeutungen und den zugehörigen Befehlen für die Textausgabe.

Gruppen Wie bereits in Abschnitt 1.3 gezeigt, ist es möglich durch geschweifte Klammern die Wirkung eines Makros zu beschränken. Diesen beschränkten Bereich nennt man bei TeX auch „Gruppe“. Er kann, wie schon erwähnt aus geschweiften Klammern, aber auch aus Umgebungen gebildet werden.

Wie bei den meisten Programmiersprachen wird bei TEX zwischen lokalen und globalen Definitionen unterschieden. Die meisten Befehle sind lokal und wirken nur innerhalb der aktuellen Gruppe. Das bedeutet, dass sie zum Ende der Gruppe wieder auf den Zustand der zu Beginn der Gruppe existierte zurückgesetzt werden.

Da Klammern über größere Bereiche unübersichtlich werden gibt es diese Makros:

```
\begingroup
\endgroup
```

Sie wirken wie eine Umgebung die keine zusätzlichen Aktionen vornimmt.

Kommentare Da das Konzept von LATEX sich damit befasst, dem Text eine logische Struktur zu geben, kann es 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 müssen. Das Prozentzeichen (%) ist das Kommentarzeichen bei TEX und LATEX. Das Programm ignoriert alles, was in der Zeile hinter dem Prozentzeichen steht. Somit kann man Beschriftungen vornehmen, die nicht in der Ausgabe erscheinen sollen.

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

1.5 Struktur des Quellcodes

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

```
\documentclass[Optionen]{Name}
...
\begin{document}
...
\end{document}
```

Präambel

Dokumentenkörper

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

Dokumentenkörper Der Teil der von der `document`-Umgebung eingeschlossen wird heißt Text-, oder Dokumentenkörper und wird oft auch als Body bezeichnet. Er enthält den eigentlichen Text mit lokalen Formatierungsbefehlen.

`\begin{document}` startet die Erzeugung der Ausgabe und `\end{document}` beendet das Compiler-Programm. Dies bedeutet, dass alles, was hinter dem Ende der `document`-Umgebung

steht, von LaTeX nicht mehr beachtet wird. Falls das Programm keinen Befehl zum Abschluss des Dokuments finden sollte, wird eine Fehlermeldung ausgegeben.

Es gibt Befehle, die nur innerhalb der Präambel bzw. nur im Dokumentenkörper 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, die in der Präambel verwendet werden müssen.

Bei allen anderen genügt oft die Überlegung, ob die Makros selbst eine Ausgabe erzeugen oder nicht. Alles, was eine Ausgabe erzeugt gehört in den Dokumentenkörper und alle Einstellungen in die Präambel.

2 Das erste LaTeX-Dokument

Nun geht es zum praktischen Teil und zwar dem Erstellen des ersten eigenen Dokuments.

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

3 Der Aufbau eines LaTeX-Dokuments

3.1 Dokumentenklassen

Die Dokumentenklasse bestimmt den Dokumententyp und nimmt die grundlegenden Einstellungen vor:

```
\documentclass[Option1, Option2,...]{Klassenname}
```

Da alle weiteren Einstellungen von dieser Auswahl abhängen, sollte die Dokumentenklasse immer als allererstes gewählt werden. Üblicherweise macht man dies 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
Die sogenannten Standardklassen sind einfach strukturiert und ein Basisbaustein von LaTeX. Sie richten sich jedoch mit ihren Einstellungen nach US-amerikanischen Konventionen für Typografie und Papierformat und verfügen zudem über deutlich weniger vorgefertigte Anpassungsmöglichkeiten.

KOMA-Script dagegen ist eine Sammlung von Klassen und Paketen für LaTeX, die seit mehr als 20 Jahren kontinuierlich weiterentwickelt und erweitert wird. Sie verbessert nicht nur die Funktionalität und Nutzerfreundlichkeit, sie ist bei weitem flexibler und bieten deutlich mehr Bedienkomfort als die Standardklassen. Die Benutzung ist allein aufgrund der möglichen typografischen Feinanpassungen und deutlichen Erweiterung der Auszeichnungssprache LaTeX sinnvoll.

Die KOMA-Klassen sind sehr gut dokumentiert (sogar zweisprachig: [12] & [13] und zusätzlich in Buchform [14]), sodass ein Blick in die Dokumentation oft hilfreich ist!

Die KOMA-Klassen (Dokumentenklassen aus dem KOMA-Script-Bundle) ermöglichen viele sonst sehr komplexe Anpassungen durch vorgefertigte Elemente. Wenn die gewünschte Anpassung mit den Elementen von KOMA-Script durchgeführt werden kann, so ist dies in jedem Fall zu priorisieren, 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).

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 grundlegenden Eigenschaften stimmen im Wesentlichen überein.

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

- ▷ Titelei auf eigener Seite (oder mehreren)

- ▷ Doppelseitiges Layout (Unterscheidung innerer/äußerer Rand anstatt links/rechts)
- ▷ Beinhaltet alle Ebenen[1]
- ▷ Kapitel beginnen immer auf neuer rechter Seite
- ▷ Seitenzählung mit römischen Ziffern (Vorspann) und arabischen Ziffern (Hauptteil & Nachspann), siehe auch Abschnitt 3.6 `\part`, `\chapter`, `\section`, ...
- ▷ Nummerierung von Abbildungen, Tabellen und Gleichungen mit vorangestellter Kapitelnummer (z. B. 1.1)
- ▷ Nummerierung der Fußnoten (bei jedem neuen Kapitel wird wieder bei 1 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)
- ▷ Beinhaltet alle Ebenen (wie scrbook)
- ▷ Kapitel beginnen immer auf neuer (jedoch nicht gezwungenermaßen rechter) Seite
- ▷ Keine Grobgliederung möglich, daher durchlaufende Seitennummerierung (wie scrartcl)
- ▷ Nummerierung von Elementen mit vorangestellter Kapitelnummer (wie scrbook)

scrletter (letter) Für Briefe. Die Klasse ist allerdings der Vorgänger des mittlerweile existierenden scrletter-Paketes und sollte deshalb nicht mehr verwendet werden. Das scrletter-Paket ermöglicht nämlich die Erzeugung eines Textdokumentes zusammen mit einem Brief in einem Dokument. Dazu gibt es spezielle Briefelemente, wie zum Beispiel Absender, Anschrift, Betreff, Anlagen, etc... Für die Benutzung wird auf das zugehörige Kapitel der KOMA-Script-Dokumentation verwiesen [12].

beamer Für Bildschirmpräsentationen. Hierfür gibt es etliche angepasste Styles. Die grundlegende Verwendung wird jedoch in diesem Dokument derzeit nicht weiter behandelt. Informationen finden sich in [25].

[1] Die einzelnen Ebenen und die dazugehörigen Befehle `\section`, `\part` usw. werden in Abschnitt 3.7 erläutert.

3.2 Dokumentenklassenoptionen

Die meisten Dokumentenklassen liefern zusätzlich zum Standardlayout weitere Optionen, um grundlegende Anpassungen am Layout vorzunehmen. Typische Beispiele hierfür sind das Papierformat oder die Basisschriftgröße. Diese werden (wie in Abschnitt 3.1 bereits erwähnt) bei der Bestimmung der Klasse mit angegeben.

```
\documentclass[Option1, Option2,...]{Klassenname}
```

Die Optionen hängen von der Klasse selber ab. Deswegen gibt es auch keine Listen an immer verfügbaren Optionen. Früher wurden dafür nur Schlüsselworte verwendet wie z. B. `a4paper` oder `a5paper`. Mittlerweile gibt es aber schönere Möglichkeiten von Schlüssel-Wert- oder sogenannten KeyVal-Strukturen (engl. Key-Value). Mit diesen muss nicht für jede Option ein eigenes Schlüsselwort registriert werden, sondern ein Schlüssel kann unterschiedliche Werte annehmen wie z. B. `paper=a4` oder `paper=a5`.

KOMA-Script unterstützt in vielen Fällen noch die alten Varianten, allerdings sollte, wenn möglich, immer die neueste verwendet werden. Diese finden sich entweder in der Paketdokumentation [12] oder teilweise auch in diesem Dokument.
Im Folgenden wird ein kurzer Überblick über die möglichen Optionen der KOMA-Klassen gegeben. Stellenweise sind diese mit einem späteren Kapitel verknüpft (mit Verweis auf die genauere Beschreibung).

Entwurfsmodus `draft=true/false`

Um die Kompilierungszeit zu verkürzen kann der Entwurfsmodus bei der Arbeit am Dokument genutzt werden. Dabei werden allerdings einige Mechanismen deaktiviert. Welche und wie viele das sind hängt von der Dokumentenklasse und den verwendeten Ergänzungspaketen ab.

Es kann also dazu kommen, dass zum Beispiel Bilder nicht eingebunden (vgl. Kapitel 10) oder Hyperlinks (Unterabschnitt 4.4.3) deaktiviert werden. Bei fast allen Dokumentenklassen erzeugt der Entwurfsmodus zusätzlich kleine, schwarze Kästchen am Ende von überlangen Zeilen. Dies erleichtert dem ungeübten Auge zu erkennen, wo eine manuelle Nachbearbeitung der Umbrüche nötig ist.

Papierformat `paper=`*Format*

Das Standardformat ist a4. Die Benennung der ISO-Formate ist identisch zu den Namen des Standards. Der Buchstabe wird klein geschrieben, z. B. a5, b6, c8, d0.

Daneben werden auch die amerikanischen Formate letter, legal und executive unterstützt.

Seitenausrichtung `paper=portrait/landscape`

Zusätzlich zum Format kann auch die Ausrichtung angegeben werden. Hierbei entspricht portrait dem Hochformat (Standard), landscape dem Querformat.

Titelei `titlepage=true/false/firstiscover`

Schaltet die Struktur der Titelei zwischen Titelseite(n) und Titelkopf um. Details siehe Abschnitt 3.5.

Grundschriftgröße `fontsize=10pt/11pt/12pt`
Einstellung der Basisschriftgröße. Andere Werte sind möglich, allerdings existieren dann keine angepassten Voreinstellungen, siehe auch Unterabschnitt 5.1.2.

Absatzkennzeichnung `parskip=`*Einstellung*
Wählt die Methode der Absatzkennzeichnung wie in Unterabschnitt 5.4.1 beschrieben aus.

Doppelseitiger Druck `twoside=true/false`
Schaltet die Unterscheidung zwischen linken und rechten Seiten ein bzw. aus. Ränder werden dann nach innerem und äußeren Rand unterschieden anstatt nur zwischen links und rechts. Die Voreinstellung ist abhängig von der Dokumentenklasse (vgl. Abschnitt 3.1).

Zweispaltiger Satz `twocolumn`
Aktiviert den zweispaltigen Satz für das gesamte Dokument. Für Details oder andere Möglichkeiten des mehrspaltigen Satzes, siehe Abschnitt 6.3.

KOMA-Script bietet zusätzlich die Möglichkeit Optionswerte zu ändern, nachdem die Dokumenteklasse geladen wurde. Diese Anpassungen werden jedoch nicht von allen Optionen unterstützt. Allerdings kann dies beispielsweise bei lokalen Anpassungen der Absatzkennzeichnung sinnvoll sein. Details der Unterschiede zwischen „Frühe[r] oder späte[r] Optionenwahl“ sind ausführlich in [12] beschrieben.

Falls eine spätere Änderung notwendig ist geschieht dies mithilfe von:

`\KOMAoptions{`*Optionenliste*`}`

Die Optionenliste hat in diesem Fall die gleiche Struktur, wie sie es bei einer Angabe als Option zu `\documentclass` oder `\usepackage` notwendig wäre.

3.3 Ergänzungspakete

LaTeX möchte unterschiedlichen Bedürfnissen gerecht werden und stellt deshalb riesige Mengen an Ergänzungspaketen zur Verfügung. Sie liefern Befehle und/oder verändern das Dokumentenlayout auf verschiedenste Weise. Normalerweise installiert man zusammen mit der Distribution automatisch die Pakete mit der höchsten Verbreitung. Sie müssen somit lediglich geladen werden:

`\usepackage[`*Option1, Option2, ...*`]{`*Paketname*`}` (in der Präambel)

Pakete können nur innerhalb der Präambel geladen werden. `\usepackage` muss somit zwischen `\documentclass` und `\begin{document}` stehen (siehe auch Beispiel 1).

Im Laufe dieses Dokuments werden einige sehr wichtige Pakete genauer betrachtet. Tabelle 3.1 zeigt 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 Fehlerquellen die in der Wechselwirkung von Paketen entstehen können und verringert außerdem die Kompilierungszeit.

Tabelle 3.1: Ergänzungspakete, die in diesem Dokument vorkommen

ams…	Ergänzungspakete für Formelsatz; Kapitel 14
array	Weitere Spaltenformatierungen für Tabellen; Abschnitt 11.2
babel	Sprachunterstützung, Trennregeln,…; Unterabschnitt 3.4.2
booktabs	Verbessertes Spacing und Linien bei Tabellen; Abschnitt 11.6
fontenc	Ausgabekodierung und Vektorschrift; Unterunterabschnitt 3.4.1
color	Ermöglicht die Verwendung von Farben für LaTeX-Elemente; Abschnitt 5.2
csquotes	Kontextsensitiver Satz von Anführungszeichen; Unterunterabschnitt 5.6.1
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; Unterabschnitt 4.4.3
inputenc	Eingabekodierung; Unterunterabschnitt 3.4.1
lmodern	Verbesserte Fassung der Schriftart Computer Modern; Unterunterabschnitt 3.4.1
longtable	Mehrseitige Tabellen; Unterabschnitt 11.7.2
mhchem	Chemische Formeln; Abschnitt 14.18
microtype	Verbessertes Trennverhalten durch Berücksichtigung der Mikrotypografie; Unterabschnitt 5.5.1
multicol	Mehrspaltiger Textsatz; Abschnitt 6.3
multirow	Tabellenzellen über mehrere Tabellenzeilen; Unterabschnitt 11.7.1
placeins	Einfache Möglichkeit, um den Bewegungsbereich von Gleitobjekten einzuschränken; Abschnitt 12.1
ragged2e	Modifizierter Flattersatz mit Worttrennungen; Tabelle 5.7
scrlayer-scrpage	Modifikationen des Seitenstils; Abschnitt 7.2
setspace	Zeilenabstand ändern; Unterabschnitt 5.4.4
siunitx	Zahlen und Einheiten typographisch korrekt setzen; Abschnitt 14.17
tabularx	Tabellen mit fester Breite über die dehnbare X-Spalte; Abschnitt 11.4
tabularray	Alternative Tabellenimplementierung auf Basis von expl3; Abschnitt 11.3
typearea	Professionelle Satzspiegelkonstruktion; Abschnitt 6.1

3.3.1 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* Pakete die geladen werden. Man kann und sollte Optionen die von mehreren Paketen verarbeitet werden können, als Dokumentenklassenoption angeben. Insbesondere gilt dies für die Optionen zur Sprache (`german`) und den Entwurfsmodus (`draft`).

Paketanleitungen finden
Für speziellere Anwendungen oder zum Nachschlagen bestimmter Makros empfiehlt es sich einen Blick in die jeweilige Paketanleitung zu werfen. Die richtige Paketanleitung zur installierten Version findet man über das Programm `texdoc`, welches Bestandteil jeder LaTeX-Distribution ist. Dies geht z. B. über das Ausführen in der Kommandozeile.

Kommandozeile

```
texdoc Paketname
```

Dieser Befehl sucht nach einer Dokumentation zum angegebenen Paketnamen und öffnet sie gegebenenfalls. Sollte keine Anleitung vorhanden sein, wird eine entsprechende Meldung angezeigt.

Wenn man mit grafischen Anwendungen leichter vertraut ist als mit Kommandozeilenprogrammen, dann 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 installiert. Auf anderen Betriebssystemen sind ggf. zusätzliche Installationen notwendig.

Einige Editoren (z. B. TeXstudio) ermöglichen zudem die Anzeige der Paketdokumentation beim Rechtsklick auf die Zeile, in der ein Paket geladen wird.

3.4 Sprachanpassung

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. Heutzutage 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.

Seit Anfang 2018 ist UTF-8 nun auch bei pdfLaTeX die voreingestellte Kodierung. Allerdings ist es durch die Implementierung[2] nicht möglich, diese in Makronamen oder Labels zu verwenden. Im Fließtext stellt sie jedoch kein Problem mehr da.

Das folgende Beispiel zeigt die grundlegenden Einstellungen:

[2] Umlaute werden intern als Makros verarbeitet.

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 zwei Schritte. Da pdfLaTeX jedoch nach wie vor weit verbreitet ist und häufig von Vorlagen verwendet wird, werden diese Anpassungen in den folgenden Abschnitten kurz besprochen. Falls kein Interesse an der Nutzung von pdfLaTeX besteht, kann dieser Abschnitt auch übersprungen werden.

3.4.1 Unicode Anpassungen für pdfLaTeX

Für die Verwendung von UTF-8 unter pdfLaTeX sind grundsätzlich zwei Schritte notwendig. Bei Versionen nach dem Frühjahr 2018 wird der erste (Unterunterabschnitt 3.4.1) automatisch ausgeführt, jedoch ist der zweite nach wie vor unabdingbar.

Eingabekodierung

Der Begriff „Eingabekodierung“ beschreibt, wie Zeichen im Quellcode abgespeichert werden. Für den Text ist die Eingabekodierung wichtig sobald man über den 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 der Standard für Textkodierung und enthält alle im Alltag notwendigen Zeichen:

```
\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 in älteren Vorlagen findet man öfter noch 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. Hier muss die Quelldatei selbst geändert werden. Am einfachsten ist dies möglich, indem im Editor eine neue `.tex`-Datei angelegt unter der Inhalt aus einer in die andere Datei kopiert wird.

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 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 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 als `T1` vorliegt (siehe auch Abschnitt 5.1).

3.4.2 Sprachanpassung (Das babel-Paket)

TeX benutzt einen sehr effektiven Algorithmus, um den Satz eines Absatzes möglichst perfekt zu gestalten. Dieser verteilt die Wörter gleichmäßig, was unschöne Löcher im Blocksatz verhindert und beschäftigt sich mit der Trennung von Wörtern am Zeilenende. Da jedoch in den unterschiedlichen Sprachen verschiedene Regeln gelten, muss man vorher eingestellen, welche davon genutzt werden sollen.

Eine Sprachanpassung ermöglicht außerdem, 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 noch das Paket polyglossia. Dieses ist jedoch auf die beiden Systeme beschränkt und da die Entwicklung bei babel stark vorangeschritten ist, wird in diesem Dokument nicht weiter darauf eingegangen.

Babel verfügt über viele Sprachen. Verschiedenste Mechanismen können deshalb auch innerhalb eines Dokumentes die Sprache 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 sind dies Pakete die mit den Verzeichnissen (insbesondere dem Literaturverzeichnis oder dem Index) arbeiten. Sinnvoll ist es daher, die Option `ngerman` nicht als Paketoption zu übergeben, sondern als globale Dokumentenoption (vgl. Abschnitt 3.3). Somit kann jedes Paket, das 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 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, Unterabschnitt 3.5.1).

Leerzeichen nach Satzzeichen

Neben den Trennregeln unterscheiden sich die beiden Sprachen Deutsch und Englisch auch durch die Leerzeichen nach einem Punkt am Satzende. Mit der Standardeinstellung 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 in 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 dieselbe Weise wie bei einem Punkt verhindert (\) oder ermöglicht (`\@`) werden.

Anführungszeichen und Umlaute

Eine weitere Eigenschaft des babel-Pakets ist der Satz von Anführungszeichen und Umlauten. Ein Überbleibsel aus der Zeit vor UTF-8 (als man Umlaute und Anführungszeichen noch nicht direkt eingeben konnte) ist, dass Zeichenfolgen wie „"U“ als „Ü“ ausgegeben werden. Diese Funktion ist aus historischen Gründen für alle deutschsprachigen Varianten voreingestellt.

Da es in deutschen Texten jedoch keinen Verwendungszweck für das Zeichen " gibt, sollte die aktivierte Funktion keine Probleme machen. Das Setzen von richtigen Anführungszeichen funktioniert teilweise ebenfalls über babel. Dieser Funktionalität wird sich jedoch später in einem eigenen Abschnitt gewidmet in Unterabschnitt 5.6.1.

Es gibt auch noch ähnliche Kombinationen für das Setzen von deutschen Anführungszeichen. Da es hier jedoch mittlerweile deutlich bequemere Lösungen gibt, werden diese in Unterabschnitt 5.6.1 erläutert.

3.5 Titelei

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

Titelseiten zeigen den Dokumententitel zusammen mit weiteren Informationen, wie beispielsweise dem 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 aber immer *nur eine* davon benutzt werden, auch wenn theoretisch Kombinationen möglich sind).

3.5.1 Automatische Titeleierzeugung

Man übergibt hier mit Hilfe von anderen Makros (wie z. B. `\author{Autor}`) die Informationen an LaTeX und lässt 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.

```
... 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* in die Präambel zu setzen, auch wenn es grundsätzlich möglich ist.

Optisch entspricht diese Variante dem ersten Beispieldokument (Beispiel 1 auf Seite 12).

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. Hier werden allerdings andere Seitenränder verwendet, als für den Rest des Dokumentes[3].

KOMA-Script bietet zudem weitere Makros zum Erzeugen von Widmungen, Schmutztiteln[4], Verlagsinformationen, ...

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

mit Symbolen gekennzeichnet

Die Ausgabe mit `\maketitle` hängt von der Dokumentenklasse ab. Teilweise erweitern Templates die Funktionalität noch um zusätzliche Elemente. Soweit eine Anpassung vorgesehen ist, ist dies üblicherweise in der Dokumentation der Vorlagen geschildert. Allerdings ist es aufgrund der Komplexität schwieriger hier einzelne Anpassungen vorzunehmen. Daher existiert zusätzlich die

[3] Die Ränder können über Makros eingestellt werden. Details dazu finden sich durch Nachschlagen des Optionswertes `firstiscover` in der KOMA-Doku [12].

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

Möglichkeit die Gestaltung komplett selbst zu übernehmen.

3.5.2 Frei gestaltbare Titelseite

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

Diese Variante bietet deutlich mehr Gestaltungsfreiraum. Sie 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. Hilfreiche Makros für die Gestaltung finden sich im Abschnitt zu Textformatierungen oder Platzierungen.

3.6 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*:	Seitennummerierung in kleinen römischen Ziffern (i, ii, ...), nicht nummerierte Kapitelüberschriften – Feinere Untergliederungen in Abschnitte nicht sinnvoll, Abschnitt geeignet für Titelei, diverse Verzeichnisse sowie Vorwort
`\mainmatter`	*Hauptteil*:	arabische Seitenzahlen beginnend mit 1, ...
`\backmatter`	*Nachspann*:	Untergliederung wie beim Vorspann, Seitennummerierung wird aus dem Hauptteil fortgesetzt

3.7 Gliederungsebenen

Um die logische Struktur eines Dokuments abbilden zu können nutzt LaTeX Gliederungsebenen. Diese erzeugen die Überschriften und unterteilen zusätzlich das Dokument in Abschnitte. Die Hierarchie ist hierbei abhängig von der Dokumentenklasse (vgl. Tabelle 3.2).

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

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

Für den aktuellen Abschnitt entspricht diese Syntax:

Tabelle 3.2: 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 scrreprt
`\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

```
\section{Gliederungsebenen}
```

Diese Makros erledigen nicht nur die automatische Nummerierung und Formatierung der Überschriften, sondern tragen zusätzlich die Überschriften ins Inhaltsverzeichnis und in die Kolumnentitel[5] ein. Für lange Überschriften kann es hilfreich sein, eine Kurzversion für den Inhaltsverzeichniseintrag zu erstellen. Dies wird mithilfe des optionalen Arguments der Gliederungsmakros erreicht.

Die gesternte Version unterdrückt sämtliche Automatismen. Es wird lediglich eine nicht nummerierte Überschrift gesetzt. Außerdem besteht kein Einfluss auf die Kolumnentitel, denn diese bleiben bei den vorherigen Werten (vgl. Unterunterabschnitt 3.7.1).

3.7.1 Erweiterung der Gliederungsbefehle durch KOMA-Script

Bei KOMA-Script gibt es mittlerweile sehr viele ergänzende Mechanismen zur Anpassung der Gliederungsebenen. Hier werden lediglich die mit der häufigsten Verbreitung besprochen. Für eine vollständige Übersicht steht die Dokumentation zur Verfügung [12].

Unterschiedliche Einträge für Inhaltsverzeichnis und Kolumnentitel

Wie das optionale Argument der Gliederungsebenen verarbeitet werden soll, kann konfiguriert werden. Diese erweiterte Funktionalität wird über die Option `headings` mit einem der drei Werte `optiontotoc`, `optiontohead` oder `optiontoheadandtoc` aktiviert.

Die Werte setzen entsprechend ihrer Bezeichnung die Standardfunktion des optionalen Arguments. Zusätzlich schaltet KOMA-Script dann auf die erweiterte Verarbeitung um. Dies ermöglicht die Angabe mehrerer Werte:

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

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

KOMA-Script prüft dabei, ob die optionale Angabe ein Gleichheitszeichen enthält. Falls dies der Fall ist, ändert es die Vorgehensweise. Benötigt man dennoch ein Komma oder Gleichheitszeichen als Bestandteil des Verzeichniseintrages, so kann der Inhalt durch Gruppierung entsprechend strukturiert werden:

```
\section[{Eine Überschrift mit = im optionalen Argument}]{Überschrift}
```

Indem man einen leeren Eintrag angibt (`head={}`), ist es außerdem möglich den Inhaltsverzeichniseintrag zu unterdrücken. Echte leere Einträge kann man mithilfe leerer Boxen (Kapitel 9) erzeugen.

Nicht nummerierte Überschriften

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

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

Es existiert jeweils auch eine Sternchen-Variante (z. B. `\addsec*`). Diese entspricht überwiegend der gesternten Version (z. B. `\section*`), allerdings wird der Kolumnentitel im Gegensatz zum klassischen Makro geleert.

Zusätzlich zu den klassischen Gliederungsebenen bietet KOMA-Script einen weiteren Überschriftentyp:

```
\minisec{Überschrift}
```

Diese Art 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.

3.7.2 Zusammenfassung

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

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

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

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

3.7.3 Anhang

Im Gegensatz zur Grobaufteilung (Abschnitt 3.6) 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, denn 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`, Unterabschnitt 4.1.3).

Entgegen einer weit verbreiteten Ansicht, dass die erste Seite des 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 und dies reicht in der Regel als Kennzeichnung vollkommen aus. In Spezialfällen sollte für diese Entscheidung jedoch die Konzeption des Anhangs berücksichtigt werden.

3.8 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 4.1).

`\tableofcontents`

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

Einfache Formatierungsänderungen geschehen mithilfe der Klassenoption `toc`. Tabelle 3.3 zeigt die möglichen Werte mitsamt einer kurzen Erläuterung. Für weitere Informationen zur Formatierung von Verzeichnissen wird auf Kapitel 13 verwiesen.

3.9 Dokumente aufteilen

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

Tabelle 3.3: Werte für die KOMA-Dokumentklassenoption `toc`

`bib`	Literaturverzeichnis erscheint im Inhaltsverzeichnis
`bibnumbered`	Literaturverzeichnis erscheint im Inhaltsverzeichnis und erhält zusätzlich eine Nummer
`flat`	Inhaltsverzeichnis ist tabellarisch; in der ersten Spalte stehen die Gliederungsnummern, in der zweiten die Überschriften und in der letzten die Seitenzahlen
`graduated`	Inhaltsverzeichnis ist hierarchisch aufgebaut mit begrenztem Platz für die Gliederungsnummern
`idx`	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

`\input{`*Dateiname*`}`
`\InputIfFileExists{`*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 Dokuments des gleichen Typs einzubinden. Eine Dateiendung muss hierbei lediglich dann angegeben werden, wenn sie sich von `.tex` unterscheidet.

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

Der zweite Befehl `\InputIfFileExists` wird dafür benötigt Fehler zu vermeiden, falls eine Datei ggf. nicht existiert.

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. Diese ermöglicht es 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 auch die 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. Kapitel 4 erhält ebenso weiterhin die Nummer 4, auch wenn Kapitel 2 und 3 im aktuellen Ausgabedokument nicht auftauchen. Selbiges gilt für die Seitenzahlen. Das alles bleibt solange erhalten, bis man an der Struktur der Datei Änderungen vornimmt.

4 Strukturautomatisierung

4.1 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 intuitiv möglich ist. In Tabelle 4.1 werden die wichtigsten Zähler, die in Standarddokumenten benutzt werden, aufgelistet.

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 ganze Zahl zum Wert des Zählers
`\stepcounter{`*Zähler*`}`	Inkrement von 1

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

`value{`*Zähler*`}`	Wert; kann überall dort als Argument angegeben werden, wo LaTeX eine Wertangabe erwartet, erzeugt keine Ausgabe
`arabic{`*Zähler*`}`	Arabische Ziffern
`Roman{`*Zähler*`}`	Große römische Ziffern
`roman{`*Zähler*`}`	Kleine römische Ziffern
`Alph{`*Zähler*`}`	Großbuchstaben A–Z (Werte von 1–26)
`alph{`*Zähler*`}`	Kleinbuchstaben a–z (Werte von 1–26)
`fnsymbol{`*Zähler*`}`	Symbole (Werte von 1–9); gedacht für die Fußnoten

Die interne Abfrage z. B. zur Ausgabe der Seitenzahl oder der Kapitelnummerierung geschieht jedoch mit:

`\the`*Zählername*

Für die Seitenzahl bedeutet dies:

```
\thepage
```

30

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 4.3) `\thepage` erreichen:

Tabelle 4.1: Auflistung wichtigster Zähler und ihre 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
`mpfootnote`	Fußnotenzähler innerhalb von `minipage`
`equation`	Formel-Zähler
`figure`	Bilder-Zähler
`table`	Tabellen-Zähler
`secnumdepth`	Nummerierungstiefe bei Überschriften
`tocdepth`	Nummerierungstiefe im Inhaltsverzeichnis

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

XXX

Bei der Seitennummerierung 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 mit 1. Die Seitenzahl kann jederzeit mit dem `\setcounter`-Befehl neu gesetzt und mit `\thepage` abgefragt werden.

4.1.1 Eigene Zähler definieren

Ein beliebiger neuer Zähler wird definiert mit:

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

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

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

```
\newcounter{mycounter}[section]
```

4.1.2 Besondere Makros für die Manipulation von der Verzeichnis-/Nummerierungstiefe

Die Zähler secnumdepth und tocdepth regeln wie weit Gliederungsebenen nummeriert bzw. ins Inhaltsverzeichnis eingetragen werden.

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 einzuschalten mit:

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

Dies kann man auch auf weitere Gliederungsebenen anwenden:

```
\partnumdepth
\chapternumdepth
\sectionnumdepth
\subsectionnumdepth
\subsubsectionnumdepth
\paragraphnumdepth
\subparagraphnumdepth
```

4.1.3 Punkt am Ende der Nummer

KOMA-Script setzt in seiner Standardeinstellung einen Abschlusspunkt ans Ende aller Gliederungsnummern, wenn die Objektnummern[1] nicht nur aus arabischen Ziffern bestehen. 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.

Dieses Verhalten kann auch für eigene Zähler genutzt werden. In diesem Fall muss in der Definition der Ausgabe \the*Zähler* am Ende ein

```
\autodot
```

ergänzt werden.

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

4.2 Längen und Zwischenräume

4.2.1 Längen

LaTeX speichert Längenmaße (wie z. B. die Breite des Textes) für die Erzeugung der Seiten in entsprechenden Variablen. Diese Variablen sind bezeichnend benannt. Ein wichtiger Unterschied zu den Zählern ist, dass Längen die Struktur eines Makros haben, also mit Backslash genutzt werden (z. B. `\linewidth`). Einige Beispiele für wichtige Längen sind in Tabelle 4.2 zu finden.

Tabelle 4.2: Wichtige 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*`}`	Länge auf neues Maß setzen
`\setlength{`*Länge*`}{`*Maß plusMaß1 minusMaß2*`}`	Elastische Definition einer Länge
`\addtolength{`*Länge*`}{`*Maß*`}`	Addition des Maßes zur Länge

Allgemein sollte man mit der direkten Änderung von Längen sehr vorsichtig sein, da Längenvariablen oft auch für weitere Funktionen als nur ihre Hauptfunktion genutzt werden. Änderungen an Maßen die den Satzspiegel beeinflussen sollten weitestgehend unterlassen beziehungsweise nur lokal gesetzt werden.

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

Tabelle 4.3: Typografische Einheiten die von LaTeX verstanden werden (nach Größe sortiert)

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

```
\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. Diese kann entweder direkt angegeben werden oder man benutzt eine bereits existierende Länge als Referenz. Somit ist es möglich einen Bruchteil einer Länge zu benutzen (z. B. `.5\linewidth`). Für die direkte Angabe stehen Zoll (in), metrische (cm/mm) und einige typografische Einheiten zur Verfügung. Diese werden in Tabelle 4.3 gezeigt.

Eine wichtige Abweichung ist der Unterschied zwischen „big point“ und „point“, da diese beiden Einheiten sehr nah beieinander liegen. Der „big point“ heißt in der gesprochenen Sprache auch DTP-Punkt oder PostScript-Punkt und wird dort oft mit dem Kürzel „pt“ angegeben. Das führt dazu, dass eine Angabe von „11 pt“ in TeX eine andere Bedeutung hat, als in anderen Textverarbeitungen wie z.B. MS Word. „11 pt“ bei Word sind für TeX „11 bp“.

Bei manchen Befehlen wird eine eigene Längenangabe gefordert. Es kann von Vorteil sein, 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 definieren:

```
\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.

```
[…] mit nachfolgendem normalen Ansatzumbruch.\par
[…] nachfolgendem Absatzabstand von 1cm.
\newlength{\parindentsaved}% Am besten in die Präambel setzen
\setlength{\parindentsaved}{\parindent}
\setlength{\parindent}{1cm}\par
[…] 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
Durch das TEX-Makro `\the` ist es möglich die Maße von Längen explizit auslesen zu können. Man gibt einfach das Längenmakro im Anschluss an `\the` an:

```
\the\linewidth
```

420.55125pt

Die Ausgabe auf diese Weise erfolgt immer in der Einheit pt. Dieser Wert kann dann entsprechend in die benötigte Einheit umgerechnet werden.

4.2.2 Zwischenräume

`\hspace{`*Längenmaß*`}`
`\hspace*{`*Längenmaß*`}`

Das `\hspace`-Makro setzt an die aktuelle Position eine horizontale Lücke mit 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 noch zu gleichen Abständen:

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

Herr Müller Brief Regensburg, den 19. April 2023

Zwei weitere Varianten, die den Zwischenraum mit Punkten oder einer Linie auffüllen, sind:

`\dotfill`
`\hrulefill`

Außerdem existieren ein paar vorgefertigte Abstände mit fester Größe, um das gesamte Dokument über ein einheitliches Layout zu erhalten:

`\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 genutzt werden (siehe Kapitel 14).

4.2.3 Vertikale Abstände

Analog zu den horizontalen Zwischenräumen existieren diese Makros:

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

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

Vorgefertigte Abstände entsprechend den horizontalen Abständen sind auch hier vorhanden. 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
```

4.3 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. Genutzt werden sollte dies aber nur, wenn man genau weiß, wofür das zu überschreibende Makro verwendet wird. Bei intern verwendeten Makros kann dies sonst schnell zu unvorhersehbarem Verhalten führen.
Um das Dokument möglichst einheitlich zu gestalten, sollten Neu- und Umdefinitionen grundsätzlich *global* vorgenommen werden. Die Makros wirken jedoch lokal und somit gelten Änderungen nur innerhalb der aktuellen Gruppe oder Umgebung und werden nach Beendigung wieder in den Ursprungszustand zurückgesetzt.

4.3.1 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 hier 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 dem Platz, an dem die Argumente eingefügt werden sollen, wird der jeweilige Platzhalter `#1` für das erste Argument, `#2` für das Zweite, usw. eingefügt.

```
\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: 38 - Nach der Änderung: XXXVIII

4.3.2 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 genutzt werden, wobei die Argumente immer nur beim Umgebungsanfang angegeben werden.

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

4.4 Querverweise

4.4.1 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 4.4.1 auf Seite 39.

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

4.4.2 Fußnoten und Randnotizen

Eine Randnotiz bzw. Fußnote[2], kann mit

Das ist eine Randnotiz.

`\marginpar[Notiz falls innen]{Notiz falls außen}`
`\footnote{Fußnotentext}`

erstellt werden.

Fußnoten
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 andere Boxen zu setzen. Ist eine Fußnote darin jedoch erforderlich, so muss sie manuell eingefügt werden:

`\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. So ist es möglich Fußnoten innerhalb von LR-Boxen, Tabellen und im Mathemodus zu erzeugen.

[2] Dies ist eine Fußnote.

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).

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*`}`

Dies ermöglicht die Erzeugung von Randnotizen die sich darin unterscheiden, ob sie auf dem rechten oder linken Rand gedruckt werden. Insbesondere im zweiseitigen Satz ist dies wichtig, da hier die Fußnoten immer außen gedruckt werden, also auf linken Seiten links und auf rechten Seiten rechts. Als Beispiel wurden die beiden Pfeile als Randnotizen auf dieser Seite und Seite 41 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{`*Text*`}`

Natürlich könnte das auch mit `\marginpar` erzeugt werden, allerdings stellt dies einen erhöhten Aufwand dar. Letztlich hat `\marginline` aber keine andere Bedeutung als:

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

4.4.3 Hyperlinks – Das hyperref-Paket

Das hyperref-Paket erweitert die Möglichkeiten im Umgang mit Textmarken und -referenzen um ein Vielfaches. Um Hyperlinks zu erzeugen wird hierbei die Einbindung dieses Pakets am meisten genutzt. Wenn man damit nämlich auf einen Link im PDF-Dokument klickt, dann 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.

Tabelle 4.4: 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 Literaturverweise
`filecolor`	cyan	Farbe für Links, die Dateien öffnen
`menucolor`	red	Farbe für Acrobat Menüeinträge
`runcolor`	filecolor	Farbe für Links, die Anmerkungen aufdecken
`urlcolor`	magenta	Farbe für URLs
`allcolors`		Weist allen Farbwerten den gleichen Wert zu

`\hypersetup{`*Optionen*`}`

Für eine saubere Umsetzung müssen die Einstelllungen jedoch innerhalb der Präambel gesetzt werden.

← Diese Randnotiz gehört zu einem Beispiel aus Unterabschnitt 4.4.2

Farbanpassugen

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

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

`colorlinks=true` färbt dabei (anstatt Rahmen zu erzeugen) alle Hyperlinks ein und `linkcolor=`*Farbe* wählt zusätzlich noch die Farbe für dokumenteninterne Verweise (nicht Literaturverweise) aus. Die übrigen Arten von Referenzen (Literaturangaben, URLs, ...) bleiben jedoch in der ihnen zugewiesenen Standardfarbe. Tabelle 4.4 zeigt die Möglichkeiten zur Farbänderung auf. Die Änderung der Farbe (welcher Art auch immer) gehört zum Text. Dies bedeutet, dass z.B. blaue Links später auf der entsprechenden Seite auch in blau gedruckt werden.

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 diese nicht mit gedruckt werden. Es gibt also automatisch eine Unterscheidung zwischen digitaler und Printversion. Tabelle 4.5 zeigt die Möglichkeiten. Wichtig im Gegensatz zu farbigen Links ist, dass man die Farbe als rgb-Farbwert angiebt, da diese Farbe anders verarbeitet wird.

Zusätzlich bietet das hyperref-Paket die Möglichkeit die PDF-Metadaten zu ändern. Die Metadaten sind sozusagen der Buchumschlag 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. In Tabelle 4.6 sind die entsprechenden Funktionen aufgelistet.

Tabelle 4.5: hyperref-Optionen zur Farbumschaltung von Rahmen
(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 4.6: 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

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

```
\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*`}`

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

`\autoref{sec:hyperref}`	Unterabschnitt 4.4.3

Die Namen werden über das babel-System übersetzt. Ihre Makros samt der Standarddefinitionen im Englischen und Deutschen finden sich in Tabelle 4.7.

Neben den hier bereits gezeigten Funktionen gibt es mit hyperref Möglichkeiten die `.pdf`-Funktionalität noch zu erweitern. Somit kann man mithilfe dieses Paketes zum Beispiel auch

Tabelle 4.7: \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

.pdf-Formulare erzeugen. Für diese spezielleren Features sei auf die Paketanleitung [22] verwiesen.

5 Textformatierungen

5.1 Schriftarten und Textauszeichnung

Schriftarten werden nach Sippe, Familie, Form und Serie klassifiziert. Die Standard-Schriftsippe in LaTeX ist „Computer Modern“. Wie bereits in Unterunterabschnitt 3.4.1 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 Schriftfamilien.

Eine sehr gute Übersicht bietet „The LaTeX Font Catalogue“[1]. Dort ist angegeben, welche Schriftarten in den TeX Distributionen enthalten sind und wie sie unter pdfLaTeX verwendet werden können.

In den moderneren Compilern ist es einfacher eine beliebige Schriftart auszuwählen. Zudem ist es möglich die Systemschriftarten des Betriebssystems zu nutzen. Hierfür genügt es den Namen der Schriftart zu wissen. Weitere Hinweise sind deshalb nicht im Katalog eingetragen.

```
\usepackage{fontspec}
```
Benötigt XeLaTeX oder LuaLaTeX

Das Paket fontspec aktiviert die Möglichkeit der Verwendung von Systemschriften. Die grundlegende Verwendung funktioniert über drei Befehle:

```
\setmainfont{Schriftart}[Features/Optionen]
\setsansfont{Schriftart}[Features/Optionen]
\setmonofont{Schriftart}[Features/Optionen]
```

5.1.1 Schriftattribute

Bei TeX und somit auch LaTeX werden die einzelnen Schriftschnitte typografisch klassifiziert. Die größte Gruppe stellt dabei die 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 „Computer Modern“. Bei dieser existieren eine Antiqua (Serifenschrift), eine Grotesk (serifenlose Schrift) sowie eine nichtproportionale Schrift (Monofont, Schreibmaschinenschrift):

[1] http://www.tug.dk/FontCatalogue/

`\rmfamily`	Serifenschrift
`\sffamily`	Serifenlose Schrift
`\ttfamily`	Monofont

Es gibt Schriftsippen, die über deutlich mehr unterschiedliche Schriftschnitte verfügen. Da die Anzahl an im Fließtext verwendeten Schrifttypen so gering wie möglich gehalten werden sollte, 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:

`\textrm{`*Text*`}`	Serifenschrift
`\textsf{`*Text*`}`	Serifenlose Schrift
`\texttt{`*Text*`}`	Monofont

Jede dieser Familien unterteilt sich in verschiedene Formen und Serien. Tabelle 5.1 zeigt die üblicherweise verfügbaren Kombinationen.

Da die aufrechte Form `\upshape` und die Serie Medium `\mdseries` Standard sind, machen diese Befehle natürlich nur Sinn, wenn man die Form und Serie zuvor geändert hat. Um wieder auf die Standardschriftart des Dokumentes zu wechseln, wird lediglich einer der folgenden Befehle benötigt:

```
\normalfont
\textnormal\marg{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[2]. Diese passen automatisch die Schriftgröße an.

`\textsuperscript{`*Text*`}`	Hochgestellter Text
`\textsubscript{`*Text*`}`	Tiefgestellter Text

Kursivkorrektur

Wenn ein aufrechtes Zeichen einem kursiven folgt, so kann – je nach verwendeter Schriftart – der Abstand zwischen beiden zu klein werden, sodass sich 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 27, S. 97]

[2] Bei den Standardklassen für LaTeX 2ε existiert nur die Variante zum Hochstellen von Elementen.

Tabelle 5.1: Verfügbare 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`

Tabelle 5.2: Skalierung der Schriften bei Hauptschrift 11 pt und Angaben der tatsächlichen Schriftgrößen bei unterschiedlichen Basisgrößen

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

```
[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 gedruckt [27, S.97]. Dieses Verhalten wurde im Digitaldruck beibehalten und darum werden heutzutage hauptsächlich die Buchstabenfolgen `fl`, `ff`, `fi`, `ffi` und `ffl` in Ligaturen umgewandelt. Je nach Schriftart gibt es mehrere oder teilweise gar keine Ligaturen.

5.1.2 Schriftgrößen

Die Schriftgrößen skalieren mit der im Befehl `\documentclass` festgelegten Standardgröße für die Schrift. Mit `\normalsize` wird die Standardschriftgröße abgefragt. Alle zur Auswahl stehenden Schriftgrößen mit Beispielen finden sich in Tabelle 5.2.

5.1.3 Textauszeichnung

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

```
\emph{Text}
```

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*

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

Selbstverständlich kann man Hervorhebungen auch durch Änderung der Schriftattribute setzen. Man sollte dabei allerdings immer darauf achten, dass der Textfluss nicht durch zu häufige oder zu auffällige Änderungen unterbrochen wird. 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! [3]

5.1.4 Globale Formatierungsänderungen für Elemente

Wenn man dauerhaft die Textformatierung für verschiedene spezielle Textelemente, wie zum Beispiel die Kopf- und/ oder 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 dabei immer, dass die Reihenfolge der Anwendung der verschiedenen Elemente beachtet wird. 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 unbedingt vermieden werden diese Elemente für die Übergabe von Textausgaben zu nutzen oder Umdefinitionen darin vorzunehmen.
Direkt auf die Schriftart von einzelnen Elementen zugreifen bzw. diese umschalten kann man mithilfe des Befehls:

`\usekomafont{`*Element*`}`

Die wichtigsten Elemente werden im Folgenden in alphabetischer Reihenfolge erläutert (eine Übersicht für alle Elemente findet man in [12]):

`author`	Autor auf der Titelseite bei Verwendung von `\maketitle` (Unterabschnitt 3.5.1)
`caption`	Gleitumgebungsbeschreibung (Abschnitt 12.2)

[3] Mit der letzten Anpassung der DIN 5008 von 2020 rät auch die DIN endlich ausdrücklich von der Verwendung von Unterstreichungen ab [4].

captionlabel	Gleitumgebungslabel (Abschnitt 12.2)
chapter	Kapitelüberschriften (Abschnitt 3.7)
chapterentry	Inhaltsverzeichniseintrag von Kapiteln (Abschnitt 3.8)
chapterentrypagenumber	Zugehörige Seitenzahl (Abschnitt 3.8)
chapterprefix	Präfix von Kapiteln/Anhängen (bei Option headings=twolinechapter/-appendix) (Abschnitt 3.7)
date	Datumsangabe auf der Titelseite bei Verwendung von \maketitle (Unterabschnitt 3.5.1)
descriptionlabel	Label einer description-Umgebung (Kapitel 8)
disposition	Alle Gliederungsüberschriften, sowie die Überschrift der Zusammenfassung, Anwendung vor spezifischen Elementen (Abschnitt 3.7)
footnote	Marke und Text von Fußnoten (Unterabschnitt 4.4.2)
footnotelabel	Marke einer Fußnote, Anwendung nach dem footnote-Element (Unterabschnitt 4.4.2)
footnotereference	Referenzierung der Fußnotenmarke (Unterabschnitt 4.4.2)
footnoterule	Linie über den Fußnoten (Unterabschnitt 4.4.2)
labelinglabel	Label der labeling-Umgebung (Abschnitt 8.2)
labelingseparator	Trennzeichen einer labeling-Umgebung (Abschnitt 8.2)
minisec	Überschrift einer \minisec (Unterunterabschnitt 3.7.1)
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 3.7)
section	Abschnittsüberschrift (Abschnitt 3.7)
subject	Typisierung des Dokumentes (Abschnitt 3.5)
subsection	Überschrift von \subsection (Abschnitt 3.7)
subsubsection	Überschrift von \subsubsection (Abschnitt 3.7)
subtitle	Untertitel auf der Titelseite bei Verwendung von \maketitle (Unterabschnitt 3.5.1)
title	Titel auf der Titelseite bei Verwendung von \maketitle (Unterabschnitt 3.5.1)
titlehead	Kopf über dem Haupttitel auf der Titelseite bei Verwendung von \maketitle (Unterabschnitt 3.5.1)

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, Grundlinienabstand, Kodierung, Familie, Form und Serie
`\useencodingofkomafont{`*Element*`}`	Kodierung
`\usesizeofkomafont{`*Element*`}`	Schriftgröße und Grundlinienabstand
`\usefamilyofkomafont{`*Element*`}`	Schriftfamilie
`\useseriesofkomafont{`*Element*`}`	Schriftserie
`\useshapeofkomafont{`*Element*`}`	Schriftform

5.2 Farben – Das xcolor-Paket

LaTeX ist normalerweise Schwarz-Weiß. Um Farben nutzen zu können sind Erweiterungen nötig. Ursprünglich lief dies über das Paket:

```
\usepackage{color}
```

Allerdings gibt es schon lange eine verbesserte Variante mit dem Paket:

```
\usepackage{xcolor}
```

Bei diesem Paket kann man ein Basisfarbmodell wählen. Die unterschiedlichen Farbmodelle hängen vom Ausgabeformat ab. Beispielsweise verfügen Bildschirme normalerweise über 3 Basisfarben (Red, Green, Blue – RGB) die meisten Farbdrucker nutzen vier Farben (Cyan, Magenta, Yellow, blacK – CMYK)[4]. Eine PDF mit RGB-Farben weist beim Ausdrucken Farbverschiebungen auf. Mit einer automatischen Konvertierung können diese Fehler etwas reduziert werden.

Für ein möglichst gutes Ergebnis sollte `xcolor` mit einer entsprechenden Option geladen werden:

`\usepackage[cmyk]{xcolor}`	Druck
`\usepackage[rgb]{xcolor}`	Bildschirm oder Beamer

Anschließend kann die Farbe umgeschaltet oder für einen bestimmten Textbereich oder eine Box gesetzt werden:

```
\color{Farbname}
\textcolor{Farbname}{Text}
\colorbox{Farbname}{Text}
\fcolorbox{Rahmenfarbe}{Hintergrundfarbe}{Text}
```

Um letzten Endes wieder zur Ausgangsfarbe zurückzukehren, existiert das Makro:

[4] xcolor unterstützt noch weitere Farbmodelle, allerdings sind diese beiden die wichtigsten. Alle Möglichkeiten sind in der Paketdokumentation zu finden [10].

`\normalcolor`

Es wechselt zurück zu der Farbe, die am Ende der Präambel gewählt wurde. Das Paket beinhaltet einige vordefinierte Farben, die ohne weitere Schritte verwendet werden können:

black darkgray gray lightgray white violet magenta pink purple red orange yellow lime green teal cyan blue olive brown

Erzeugt werden diese jeweils mit:

`\textcolor{`*Farbname*`}{`*Farbname*`}`

Es gibt zusätzlich die Möglichkeit auch Farben direkt über Farbwerte ansprechen zu können, allerdings widerspricht dies dem Grundgedanken von LaTeX. Zusätzlich sollten in einem professionellen Dokument Farben mit Bedacht und zueinander passend eingesetzt werden. Für ein einheitliches Layout ist es daher sinnvoll Basisfarben zu definieren und ggf. Schattierungen davon.

Eigene Farbdefinitionen werden bei `xcolor` nach Modell gesetzt, allerdings ist es möglich direkt unterschiedliche Werte je nach Einstellung des Basismodells anzugeben:

`\definecolor{`*Farbname*`}{`*Modellliste*`}{`*Definitionsliste*`}`

Falls nur ein Modell angegeben wird und dieses nicht dem Zielmodell entspricht wird auch hier versucht zu konvertieren.

Beispielsweise wird die peiTeX Logofarbe in diesem Dokument definiert als:

```
\definecolor{peitex}{rgb/cmyk}{1,.6,0/0,.4,1,0}
```

Schattierungen oder Farbmischungen sind nach der Definition einer Farbe über die Nutzung von

Farbname1`!`*Farbname2*`...`

möglich. Falls der zweite Farbname nicht angegeben ist erfolgt eine Mischung mit der „Hintergrund"/„Papierfarbe" (hier weiß). Das folgende Beispiel zeigt eine Folge mit Werten von 0 bis 100 in Zehnerschritten:

Farbangabe als `peitex!`*Zahl*: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100,
Farbangabe als `peitex!`*Zahl*`!black`: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100,

5.3 Code „wörtlich" ausdrucken

Oft ist es nötig - zum Beispiel zum Schreiben eines Skriptes über LaTeX - Code wörtlich im Dokument abdrucken zu können. Hierfür existieren die beiden Umgebungen:

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

Die Sternchenform unterscheidet sich von der ungesternten, indem 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...

\end{document}
\end{verbatim*}
```

```
\documentclass[ngerman]{scrartcl}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{babel}
\begin{document}

Ein␣bisschen␣Text...

\end{document}
```

Zusätzlich gibt es noch zwei Makros für kleinere Codeschnipsel:

`\verb*{Code}`

Da der Code hier ohne Bedeutung gelesen wird, ist es nicht möglich geschweifte Klammern innerhalb des Arguments zu verwenden. Die öffnende Klammer liest alles bis zur nächsten schließenden und verarbeitet den Inhalt als Argument. Die Zuordnung von Anfang und Ende wäre damit fehlerhaft.

Um solche Zuordnungsprobleme zu vermeiden, kann auch einfach ein anderes Gruppierungszeichen verwendet werden, z.B.:

```
\verb|\code{mit einem Argument}|
```

```
\code{mit einem Argument}
```

Die Wahl des Zeichens ist beliebig mit Ausnahme von Leerzeichen oder Zeilenumbrüchen. Allerdings muss am Anfang und Ende das identische Zeichen stehen.

5.4 Umbrüche

5.4.1 Absatzumbruch

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

`\par`

Die Art und Weise, wie Absatzümbrüche gekennzeichnet werden sollen, wird über die Dokumentenklassenoption `parskip=`*Methode* festgelegt. Grundsätzlich wird zwischen zwei Varianten der Absatzkennzeichnug unterschieden:

Einrückung der ersten Zeile In einem neuen Absatz wird die erste Zeile um einen Abstand eingerückt. Ein neuer Sinnabschnitt wird allerdings nur durch einen Abstand gekennzeichnet und nicht eingerückt (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[5].

KOMA-Script verfügt darüber hinaus über Modifizierer, bei einer Absatzkennzeichnung durch Abstand, ebenso die letzte Zeile des Absatzes gesondert zu kennzeichnen. Diese und die zugehörigen Erläuterungen sind in der Dokumentation [14] aufgeführt.

Der Vorteil der Einrückung liegt darin, dass der Anfang und nicht das Ende des Absatzes gekennzeichnet wird. Dies ermöglicht beispielsweise eine Unterscheidung, wenn ein Seiten- mit einem Absatzumbruch zusammenfällt. Auch bei Verwendung abgesetzter mathematischer Formeln (Kapitel 14) hat diese Variante Vorteile. Trotz Abstand kann man so 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.

Um den Einzug der ersten Zeile lokal zu verhindern existiert das Makro:

`\noindent`

Um den Einzug dagegen zu aktivieren nutzt man:

`\indent`

5.4.2 Zeilenumbruch

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

[5] Hier finden die Makros `\smallskip`, `\medskip` und `\bigskip` Anwendung, die in Unterabschnitt 4.2.3 erklärt werden.

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:

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 manuellem Umbruch `\\`*:*
Dies ist eine sehr lange Zeile Text bis zu einem Umbruch, um den
Unterschied zwischen `\linebreak` und `\\` zu demonstrieren.

`\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 einen Umbruch, was zu sehr unschönen riesigen Wortabständen führen kann.

5.4.3 Zeilenumbruch verhindern

`~`	„geschütztes Leerzeichen“; kein Zeilenumbruch zwischen Wörtern; Beispiel `Stufe~5`
`\nolinebreak[`*Priorität*`]`	kein Zeilenumbruch, mit Angabe der Priorität (0=niedrig bis 4=zwingend)

5.4.4 Zeilenabstand ändern

Mit Änderungen am Zeilenabstand sollte man prinzipiell vorsichtig sein. Die Standardeinstellungen von LaTeX sind in der Regel so gut, dass laienhaftes Eingreifen zur Verschlechterung führen kann. Leider gibt es oft Vorgaben für Haus- oder Abschlussarbeiten die unschöne Einstellungen, wie beispielsweise doppelten Zeilenabstand verlangen[6].

Falls eine Vorgabe für den Zeilenabstand besteht, so sollte man beachten, dass bei Verwendung von doppeltem Zeilenabstand lediglich der Fließtext mit eben diesem gesetzt wird. Für die Titelseite oder Verzeichnisse wechselt man auf einfachen Zeilenabstand.

Möchte man solche Vorgaben erfüllen, dann bindet man am besten das Paket setspace ein:

```
\usepackage{setspace}
```

[6] Unschön deshalb, weil der Streifeneffekt bei zu großem Zeilenabstand zunimmt und den gleichmäßigen Graueindruck der Seite stört. Dies trägt zur Verschlechterung der Lesbarkeit bei.

Dazu lautet der Befehl für eineinhalbfachen Zeilenabstand dann:

`\onehalfspacing`

Für doppelten Zeilenabstand nutzt man:

`\doublespacing`

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

`\singlespacing`

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

5.4.5 Seitenumbruch

Um einen Seitenumbruch vornehmen zu können, kann man folgende Makros verwenden:

`\newpage`	Seitenumbruch (Positionierung der Gleitobjekte möglich)
`\pagebreak[`*Priorität*`]`	Seitenumbruch mit Angabe der Priorität (0=niedrig bis 4=zwingend)
`\clearpage`	Sofortige Beendigung der aktuellen Seite (Ausgabe übriger Gleitobjekte im Anschluss)
`\cleardoublepage`	Wie `\clearpage`, allerdings mit Beginn des Textes auf der nächsten ungeraden (rechten) Seite

Der Unterschied zwischen einem Seitenumbruch mit `\newpage` und `\pagebreak` ist derselbe, wie beim Zeilenumbruch zwischen `\newline` und `\linebreak`.
Bei `\pagebreak` wird der Blocksatz eingehalten. Hier wird allerdings nicht der Wortabstand entsprechend vergrößert, sondern ein Seitenumbruch erfolgt erst, nachem der Text bis zum Zeilenende ausgeschrieben wurde. `\pagebreak` innerhalb eines Absatzes erzwingt somit lediglich einen Seitenumbruch am Zeilenende.

`\newpage` beendet die Seite sofort und führt den Text auf einer neuen Seite fort. Ausgegebene Gleitobjekte (siehe Kapitel 12) können allerdings im entstandenen offenen Weißraum am Ende des Textes eingefügt werden.

`\clearpage` beendet die Seite ebenfalls direkt und setzt im Anschluss auf einer oder mehreren Seiten sämtliche noch zu setzende Gleitobjekte.

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 `\clearpage` oder `\cleardoublepage`.

5.4.6 Seitenumbruch verhindern

`\nopagebreak[`*Priorität*`]`	Kein Seitenumbruch, Angabe der Priorität (0=niedrig bis 4=zwingend)

\nopagebreak verhindert, dass zwischen zwei Absätzen ein Seitenumbruch eingefügt wird. Innerhalb eines Absatzes unterbindet es einen Seitenumbruch nach der aktuellen Zeile.

```
\begin{samepage}
  ...
\end{samepage}
```

Die samepage-Umgebung erlaubt für ihren Inhalt nur Seitenumbrüche bei Absatzumbrüchen.

5.4.7 Unsaubere Seitenumbrüche – Seitengröße in Sonderfällen anpassen

In Sonderfällen ist es 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}
```

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. Eine Vergrößerung des Textbereiches um eine einzelne Zeile, kann dann so aussehen:

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

5.5 Silbentrennung

LaTeX kann Wörter entsprechend der Spracheinstellung trennen. Hierfür werden Silben definiert, die als Muster dienen. Diese werden über das zuständige Sprach-Paket geladen und aktiviert (vgl. 3.4.2).

Falls ein Umbruch eines Absatzes nur unschön gelingt, meldet LaTeX dies mit einer „overfull \hbox“ oder „underfull \hbox“ Warnung. „Overfull“ bedeutet, dass LaTeX über die Zeile hinaus in den Rand schreibt. In diesem Fall ist auch angegeben, wie weit der Text in den Rand hinaus ragt.

```
Overfull \hbox (1.0212pt too wide) in paragraph
```

Bei „Underfull“ hingegen sind die Wortabstände innerhalb der Zeile zu groß. Dies kann passieren, wenn die Wortabstände sich zu stark von denen aus der Zeile vorher unterscheiden. Die angegebene „badness“ kann dabei tatsächlich als Faktor genutzt werden, um zu sehen „wie schlimm“ dieser Vorfall ist.

Der maximale Wert für die „badness“ liegt bei 10 000.

```
Underfull \hbox (badness 1248) in paragraph
```

Wenn LaTeX sich nicht besser zu helfen weiß, wird Text ungünstig umgebrochen. Dieses Verhalten tritt in den meisten Fällen der beiden Warnungstypen auf. Entweder kennt LaTeX eine Trennstelle nicht oder potenziell mögliche Trennungen sind verboten.

Am auffälligsten wird dies bei Bindestrich-Wörtern. LaTeX verbietet im Allgemeinen eine Trennung von Wörtern mit Bindestrich abseits der Bindestrichposition. Für sehr lange Wörter kann eine zusätzliche Trennstelle dennoch notwendig sein. Mithilfe von babel funktioniert das über einen „hard hyphen“. Dieser wird über `\babelhyphen{hard}` eingefügt und ersetzt den einfachen Bindestrich falls notwendig.

```
Bindestrich-Wort\\
Bindestrich\babelhyphen{hard}Wort
```

Bindestrich-
Wort
Bindestrich-Wort

Im Fließtext ohne Einfluss von Bindestrichen wird bei TeX eine Trennmöglichkeit manuell angegeben durch:

`\-`

Ähnlich zum einfachen Bindestrich schließt eine solche Angabe andere Trennmöglichkeiten aus. babel liefert daher auch Möglichkeiten eine zusätzliche Trennstelle anzugeben ohne die Trennung in anderen Wortteilen zu unterbinden.

Über weitere Modi, die es ermöglichen die Trennung zu beeinflussen, verfügt das Makro:

`\babelhyphen*{`*Typ/Text*`}`

Mit folgenden Typbezeichnungen kann man die Trennstellen definieren:

`soft` Normale zusätzliche Trennstelle. Bei einer Trennung wird ein Bindestrich eingefügt, sonst ist er unsichtbar.

`hard` Setzt immer einen Bindestrich.

`repeat` In einigen Sprachen wird bei einer Trennung sowohl vor, als auch nach der Trennstelle ein Bindestrich eingefügt (z. B. Polnisch, Portugiesisch oder Spanisch)

`nobreak` Ein Bindestrich, an dem nicht umgebrochen werden darf sogar dann nicht, wenn ein Leerzeichen folgt.

`empty` Trennstelle, an der kein Bindestrich eingefügt wird.

`sonstige` Falls der Inhalt des Arguments von `\babelhyphen` keiner der bisher genannten Zeichenfolgen entspricht, wird der Text direkt eingefügt und wie ein „hard hyphen“ behandelt. Der Text innerhalb des Arguments kann nicht umgebrochen werden. Um nach einem Schrägstrich umzubrechen, kann dies beispielsweise mit `/` gemacht werden.

Außerdem regelt das Sternchen, ob zusätzliche Trennungen unterbunden werden sollen. `\babelhyphen` erlaubt Trennungen innerhalb der übrigen Wortteile, `\babelhyphen*` unterbindet diese.

Die deutsche babel-Anpassung aktiviert zusätzlich Shorthands, die eine sehr ähnliche Funktionalität haben.

`-`	Bindestrich, der andere Trennungen unterdrückt
`"=`	Bindestrich, der andere Trennungen erlaubt (`\babelhyphen{hard}`)
`"~`	Bindestrich, an dem nicht getrennt werden darf (`\babelhyphen{nobreak}`)
`\-`	Trennmöglichkeit, die andere Trennungen ausschließt (`Ur\-instinkt`, `Stau\-becken`)
`"-`	Trennmöglichkeit, die andere Trennungen nicht ausschließt (`\babelhyphen{soft}`)
`""`	Trennmöglichkeit, bei der kein Trennstrich benötigt wird (`\babelhyphen{empty}`)
`"`	Ligaturauflösung und Schaffung einer Trennmöglichkeit (`Auf"|forderung` setzt Aufforderung statt Aufforderung)
`"/`	Schrägstrich mit nachfolgender Trennmöglichkeit (`\babelhyphen{/}`)

Kommt ein Wort öfters vor, dann 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}
```

5.5.1 Mikrotypographie für verbesserte Trennungen

Das microtype-Paket erweitert den Trennalgorithmus um mikrotypographische Möglichkeiten.

```
\usepackage{microtype}
```

Beispielsweise wird damit ein optischer Randausgleich aktiviert. Um ein gleichmäßigeres Bild zu erzeugen, stehen Bindestriche somit an Trennungen leicht über den Rand hinaus.

Außerdem ermöglicht es eine minimale Dehnung einzelner Zeichen. Auch wenn die Anpassungen gering sind, entfallen dadurch viele manuelle Korrekturen von selbst.

Das Paket stellt darüber hinaus noch jede Menge Konfigurationsmöglichkeiten zur Verfügung. Diese finden sich – wie üblich – in der Dokumentation [24].

5.5.2 Geschützte Leerzeichen

Wie bereits in Unterabschnitt 5.4.6 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`

5.6 Sonderzeichen

5.6.1 Anführungszeichen

Anführungszeichen stehen prinzipiell vor und hinter [vgl. 19, S.29]:

- einer wörtlich wiedergegebenen Äußerung (direkte Rede),
- einer wörtlich angeführten Textstelle (Zitat)[7]
- 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“[19, S.30], siehe Abschnitt 5.7.

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.

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 (Unterunterabschnitt 3.4.1) und bei erweitertem Kerning auch für das Paket microtype (Unterabschnitt 5.5.1).

Eingabe von Anführungszeichen
Unter Verwendung der Eingabekodierung UTF-8 ist es möglich Anführungszeichen direkt einzugeben. Hierfür sind auf Standard-Tastaturen (abängig vom Betriebssystem) komplexe Tastenkombinationen notwendig. Um diese Eingabe zu vereinfachen, gibt es in LaTeX Abkürzungen (sog. „shorthands“). Die englischen Varianten sind immer aktiv. Die Kombinationen für deutsche Anführungszeichen aktivieren sich, wenn babel mit der entsprechenden Option geladen wird (vgl. Unterabschnitt 3.4.2).

Deutsche Anführungszeichen
(benötigt wird babel mit deutscher Sprachoption)

	„Gänsefüßchen“:		“Spitze Form„[8]:	
Doppelt öffnend:	`\glqq` oder `"``	„	`\frqq`	»
Doppelt schließend:	`\grqq` oder `"'`	“	`\flqq`	«
Einfach öffnend:	`\glq`	‚	`\frq`	›
Einfach schließend:	`\grq`	‘	`\flq`	‹

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

Die Makros sind – wenn dies vielleicht auch nicht direkt ersichtlich ist – benennend bezeichnet. `\glq` steht dabei für „German Left Quote“. `qq` bezeichnet entsprechend die doppelten Anführungszeichen.

Englische Anführungszeichen

Doppelt öffnend:	``` `` ```	“
Doppelt schließend:	`''`	”
Einfach öffnend:	`` ` ``	‘
Einfach schließend:	`'`	’

Einführungszeichen mit csquotes

Das Paket csquotes vereinfacht die Eingabe der Anführungszeichen erheblich. Es ermöglicht unter anderem kontextabhängige und durch babel an die Sprache angepasste Anführungszeichen mit dem Makro:

`\enquote{`*Zitatinhalt*`}`

Am folgenden Beispiel werden die Unterschiede aufgezeigt. Beide Vorgänge haben dieses Ergebnis:

> 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.“

So sieht es in LaTeX nur mit dem babel-Paket aus:

```
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{}
```

So mit csquotes:

```
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.}
```

Es gibt also mehrere Möglichkeiten die Anführungszeichen zu erzeugen. Beide Versionen kommen zum selben Ergebnis. Es ist allerdings empfehlenswert csquotes zu laden und zu verwenden.

5.6.2 Binde- & Gedankenstrich

Auch wenn der Unterschied (ohne den direkten Vergleich) nicht jedem auffällt, sollte man vor allem in wichtigen Dokumenten darauf achten Binde- und Gedankenstriche richtig zu setzen.

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

Im klassischen Englisch gibt es sogar noch einen Strichtyp mehr, den Streckenstrich („von–bis-Strich“). Dieser wird jedoch nicht mehr durchgehend benutzt. So beschreibt der Typograf Bringhurst den sogenannten Geviertstrich[9] als „zu lang für gute Typografie“ [2].

In der deutschen Typografie gibt es nur zwei unterschiedliche Strichtypen. Sie werden je nach Anwendung von Leerzeichen umgeben oder nicht.

LaTeX stellt um allen Ansprüchen gerecht zu werden drei verschiedene Strichtypen zur Verfügung:

`-`	-	Divis; Bindestrich; Trennstrich
`-- oder –`	–	Halbgeviertstrich; Streckenstrich; Gedankenstrich
`--- oder —`	—	Geviertstrich; klassischer englischer Gedankenstrich

Es 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 und zwar das *Minuszeichen*. Es ist häufig so lang wie ein Halbgeviertstrich und passt damit zu Tabellenziffern. 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 verwendet. Um die unterschiedliche Bedeutung auch über das Markup zu wahren, sollten beide dennoch nicht austauschbar verwendet werden.

5.6.3 Akzente und Sonderzeichen

Die deutsche Sprache gehört zu den Sprachen 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 der Nutzung von XeLaTeX oder LuaLaTeX gibt es hierbei keinerlei Einschränkungen. Für pdfLaTeX ist es derzeit[10] 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 5.3 und 5.4 zeigen die möglichen Sonderzeichen und Akzente in der klassischen LaTeX-Notation.

5.6.4 Auslassungszeichen

Das Auslassungszeichen oder auch Ellipse (…) wird bei LaTeX mit diesem Befehl gesetzt:

```
\ldots
```

Beim Satz von drei einfachen Punkten (...) stimmen die Abstände nicht. Bei Aufzählungen ist es zudem nötig einen kleinen Abstand zwischen die Auslassungspunkte und ein folgendes Komma zu

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

[10] (März 2019)

Tabelle 5.3: Akzente in der klassischen LaTeX-Notation für alle Buchstaben (hier anhand des Beispielbuchstabens „o“)

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 5.4: 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				

setzen:

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

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

Das Makro `\dots` ist eine Abkürzung für `\ldots\,`. Der Unterschied zwischen `\dots` und `\ldots` wird allerdings zu oft nicht beachtet, was fehlerhafte Abstände zur Folge hat.

5.7 Textausrichtung

Standardmäßig setzt LaTeX jeden Text im Blocksatz, also links- und/oder rechtsseitig bündig. Es gibt daher keinen Befehl für Blocksatz. Um ihn trotzdem zu erhalten, beendet man einfach eine andere Form der Ausrichtung. Um Text im Flattersatz zu setzen gibt es die in Tabelle 5.5 dargestellten Befehle und Umgebungen.

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

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

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

Tabelle 5.5: 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`

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. Wenn nur das notwendige Argument für den Einzug angegeben wird, dann 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 (z. B. innerhalb von Tabellen) zu sehr unschönen Ergebnissen führen.

```
\usepackage{ragged2e}
```

Das Paket ragged2e bietet einen Ersatz für die herkömmlichen Umgebungen und Makros für den Flattersatz, welche die Worttrennung nicht deaktivieren, siehe Tabelle 5.5. Zusätzlich lassen sie sich in ihrem Verhalten durch eine Vielzahl von Parametern beeinflussen. Diese können der Paketdokumentation [17] entnommen werden. Außerdem 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.

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* [14, S. 485]

6.1 Satzspiegelkonstruktion mit typearea

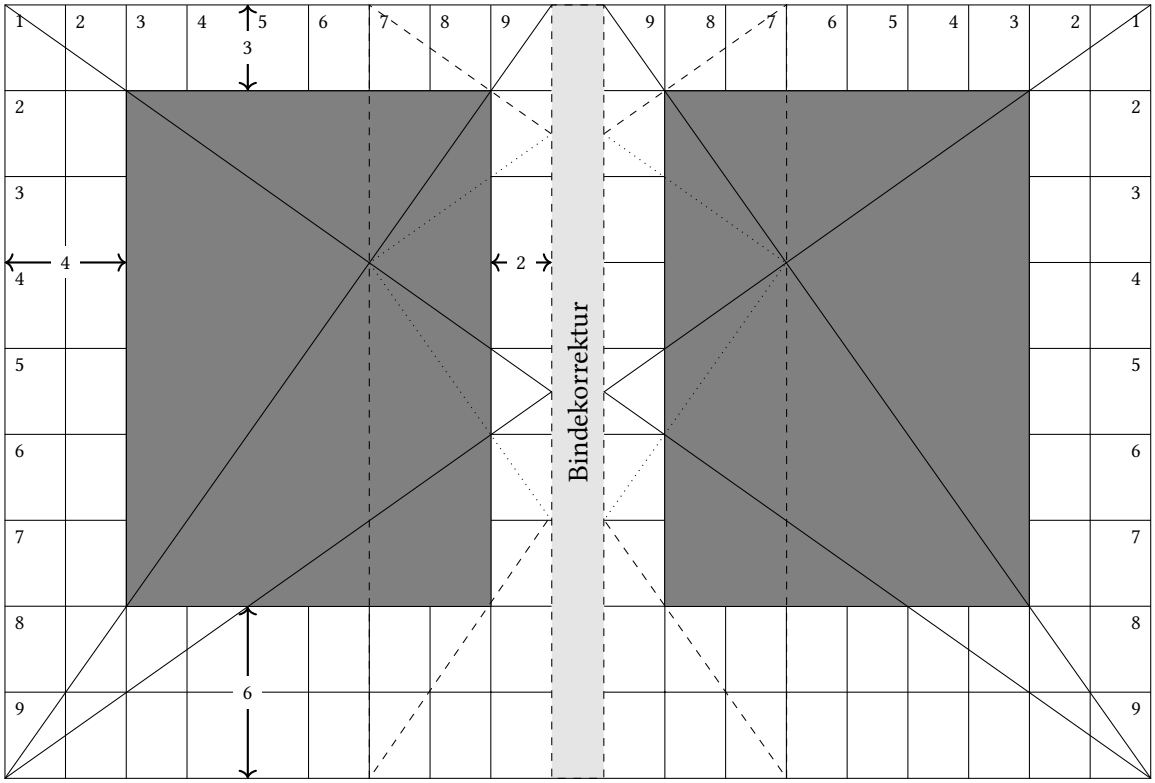


Abbildung 6.1: Doppelseite mit Satzspiegelkonstruktion in klassischer Neunerteilung (entspricht `DIV=9`) [nach 14, S.32]

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 *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) beliebig modifiziert werden:

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

nur bei Nicht-KOMA-Klassen nötig

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

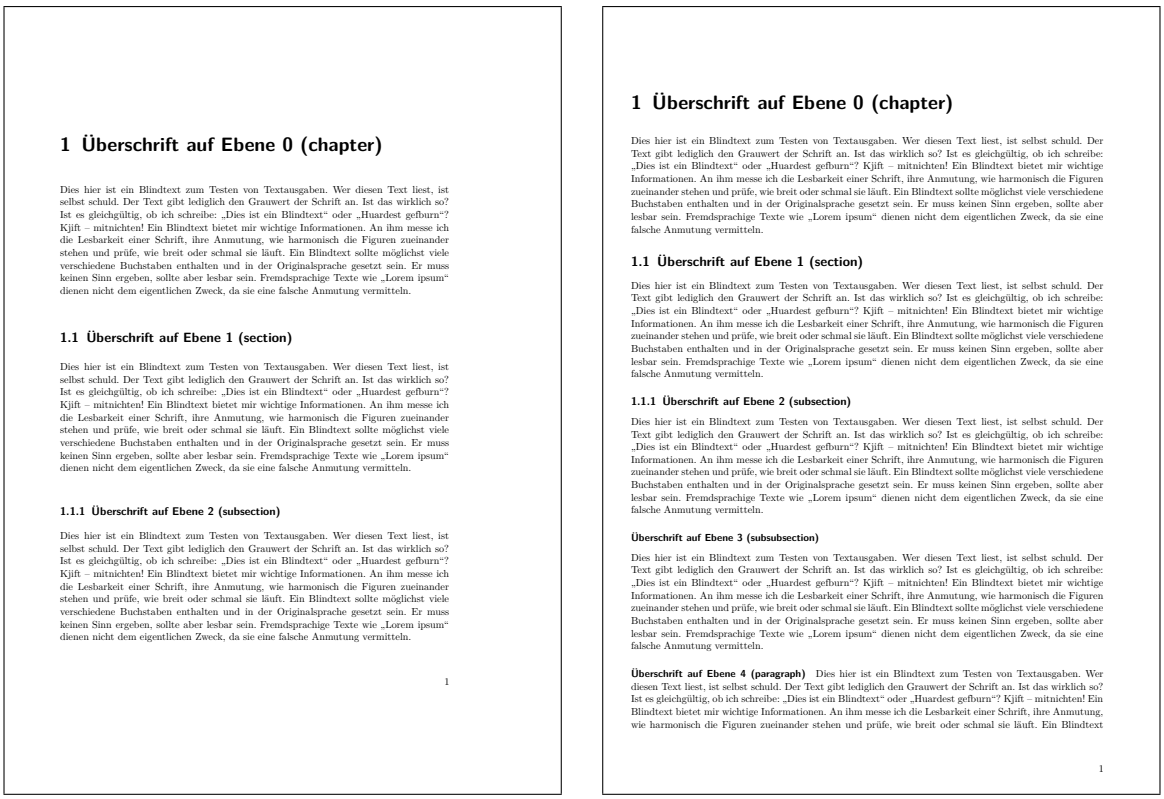
DIV=default Neuberechnung des Satzspiegels für das aktuelle Papierformat und die aktuelle Grundschrift

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



1 Überschrift auf Ebene 0 (chapter)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

1.1 Überschrift auf Ebene 1 (section)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

1.1.1 Überschrift auf Ebene 2 (subsection)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

1

1 Überschrift auf Ebene 0 (chapter)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

1.1 Überschrift auf Ebene 1 (section)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

1.1.1 Überschrift auf Ebene 2 (subsection)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Überschrift auf Ebene 3 (subsubsection)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Überschrift auf Ebene 4 (paragraph) Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext

1

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.

Neben den Rändern wird durch das Umschalten auch das Verhalten von LaTeX geändert und zwar, wie die Seite gefüllt werden soll. Aktiv ist so im doppelseitigen Satz das Makro:

`\flushbottom`

Es sorgt dafür, dass dehnbare vertikale Abstände (auch `\parskip`, selbst bei `parskip=false` vgl. Unterabschnitt 5.4.1) so weit gedehnt werden, dass die unterste Zeile auf jeder Seite die gleiche Position hat und bündig mit dem Rand abschließt.

`\raggedbottom`

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

Kopf- und Fußzeile

Neben den Einstellungen für die Ausmaße des Satzspiegels ist es auch wichtig, sich damit zu beschäftigen, 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. Ist die Kopfzeile eher ein Teil des Textes oder des Randes? Grundsätzlich lässt sich 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 [12].

Was an Einstellungen genau 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 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.

`\usepackage{geometry}`

Tabelle 6.1: Übersicht der 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 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

Diese Variante ist jedoch nicht sonderlich elegant, weswegen man auf sie nur dann zurückgreifen sollte, wenn es absolut keine andere Möglichkeit gibt.

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

`\geometry{`*Option1,...*`}`	Änderung des Layouts entsprechend der Optionen; Optionen werden zusätzlich zu vorherigen verwendet; Sollte nur in der Präambel verwendet werden
`\newgeometry{`*Option1,...*`}`	Überschreibung der bisherigen Optionen; Papiergröße ist damit nicht einstellbar
`\restoregeometry`	Aktivierung der ursprünglichen Einstellungen aus dem Header
`\savegeometry{`*Name*`}`	Speicherung der aktuellen Einstellungen unter *Name*
`\loadgeometry{`*Name*`}`	Aktivierung der 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` erstellt werden. Zusätzlich existieren in Standard-LaTeX die Makros:

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

Beide Befehle beenden die aktuelle Seite und geben ggf. noch zu platzierende Gleitobjekte aus (vgl. `clearpage`, Unterabschnitt 5.4.5). Bei `\twocolumn` existiert zusätzlich die Möglichkeit noch Text – wie beispielsweise eine Überschrift – über beide Spalten hinweg zu platzieren bevor in den zweispaltigen Satz gewechselt wird.

Das multicol-Paket
Das multicol-Paket behebt einige Unschönheiten von Standard-LaTeX. Beispielsweise füllt es nicht zuerst die linke Spalte vollständig auf, sondern auf jeder Seite beide Spalten gleichmäßig. Zudem erlaubt es bis zu zehnspaltigen Textsatz. Allerdings sind innerhalb dieser Umgebung keine Gleitobjekte[2] möglich. Mit der Umgebung

```
\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.

[1] Für genauere Hinweise zu diesem Thema, siehe Ende des Abschnitts 6.1.
[2] Gleitobjekte sind Elemente, die keine feste Position haben, sondern möglichst seitenfüllend platziert werden, siehe Kapitel 12 ab 99.

7 Der Seitenstil

7.1 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. Es kann allerdings erforderlich sein manuell einzugreifen, um zum Beispiel ein Bild zu positionieren das die Kopfzeile teilweise überdeckt. Das manuelle Umschalten erfolgt über die Befehle:

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

Dabei ist `\pagestyle` ein Schalter, der den Seitenstil global umstellt, wohingegen sich das Makro `\thispagestyle` lediglich auf die aktuelle Seite bezieht. Ohne zusätzliche Pakete existieren die folgenden Seitenstile:

empty Kopf und Fußzeilen bleiben vollständig leer.

plain Keine Inhalte im Seitenkopf. Die Seitennummer wird in der Fußzeile außen (zweiseitig) oder zentriert (einseitig) ausgegeben.

headings Seitenstil für lebende Kolumnentitel[1]. Die Überschriften werden automatisch in die Kopfzeile übernommen. Bei scrbook und scrreprt werden im doppelseitigen Layout die Kapitelüberschriften (links) und auf der anderen Seite die Abschnittsüberschriften (rechts) außen in der Kopfzeile platziert. Die Seitenzahl befindet sich im Fuß außen. Im einseitigen Layout wird nur die oberste Ebene verwendet und mit der Seitennummer zentriert ausgegeben.

myheadings Eigene Kolumnentitel. Entspricht `headings`, allerdings werden hier die Inhalte nicht automatisch erzeugt. Um Inhalte einzufügen existieren die Befehle `\markboth` und `\markright`.

```
\markboth{Inhalt für linke Seiten}{Inhalt für rechte Seiten}
\markright{Inhalt für rechte Seiten}
```

Im einseitigen Layout wird lediglich der Inhalt für die rechten Seiten verwendet.

[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 sich dagegen im Verlauf des Werkes nicht.

Kapitelanfangsseiten haben normalerweise keinen Seitenkopf, da die Überschrift dort sonst doppelt erscheinen würde. Sie verwenden daher automatisch den Stil `plain`.

KOMA-Script ermöglicht es solche Voreinstellungen zu ändern. Welcher Seitenstil auf besonderen Seiten verwendet wird ist daher jeweils in einem eigenen Makro gespeichert:

`\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

Eine Anpassung ist mithilfe von `\renewcommand` möglich (Abschnitt 4.3), dabei entspricht die Bedeutung dem Namen des Stils der verwendet werden soll.

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. Dieses bietet ein Ebenenmodell, wie es von Bildbearbeitungsprogrammen bekannt sein könnte sowie ein darauf basierendes Seitenstil-Modell. Zu den grundlegenden Funktionen gehören zwei individuell konfigurierbare Seitenstile: `scrheadings` und `plain.scrheadings`.

Die neuen Seitenstile werden automatisch beim Laden des Pakets aktiviert und ersetzen die Stile `headings` und `plain`. Der Seitenstil `scrheadings` kann außerdem auch manuell über den `\pagestyle` Befehl aktiviert werden.

Für automatische Kolumnentitel sollte die Paketoption

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

gewählt werden. Standardmäßig werden somit die Marken so gesetzt, dass in Klassen mit `\chapter` die Kapitelüberschrift links und die Abschnittsüberschrift rechts erscheinen. In den übrigen Klassen ist dieses Verhalten eine Ebene nach unten verschoben. Analog zu `headings` wird im einseitigen Layout nur der Inhalt für die rechte Seite ausgegeben.

Neben der Option die global wirkt, können die lebenden Kolumnentitel auch innerhalb des Dokuments an- oder ausgeschaltet werden.

```
\manualmark
\automark*[Ebene rechte Seite]{Ebene linke Seite}
```

`\automark*` ändert dabei nur die Anpassung für die angegebenen Ebenen, wohingegen die Variante ohne Sternchen alle vorherigen Einstellungen abschaltet. Als „Ebene“ wird hierbei der Name der Gliederungsebene verwendet, dessen Überschrift dort eingetragen werden soll.

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`

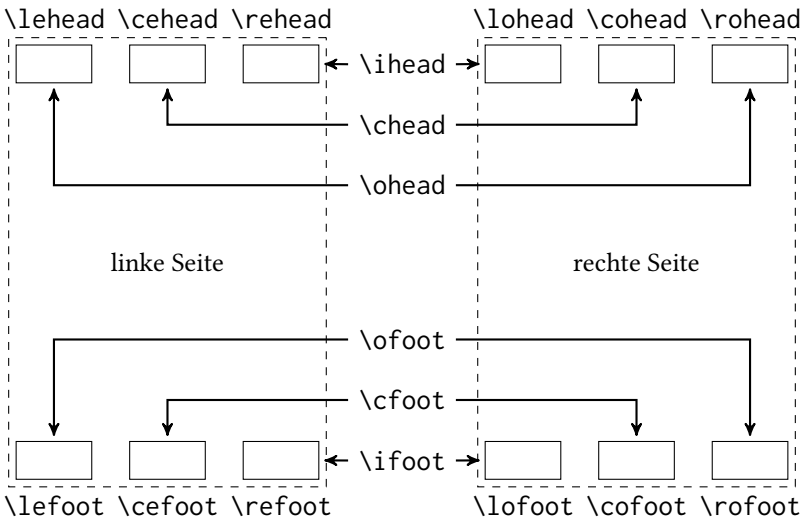


Abbildung 7.1: Zuordnung der Befehle für scrlayer-scrpage [12, S. 277]

so, „dass es den Abstand der letzten Grundlinie des Texbereichs von der ersten Standard-Grundlinie des Fußes darstellt“ [12, S. 268].

Falls die Werte nicht passen, gibt KOMA-Script eine Warnung aus. Diese enthält eine Empfehlung, um die Längen anpassen zu können.

7.2.2 Seitenstile modifizieren

Die Modifikationen 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, nutzt man:

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

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

```
\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, nutzt man am besten:

`\clearmainofpairofpagestyles`	Hauptstil (`scrheadings`)
`\clearplainofpairofpagestyles`	plain-Stil (`plain.scrheadings`)
`\clearpairofpagestyles`	Leerung aller Felder des ganzen Seitenstilpaares

Dies erspart beispielsweise viel Schreibarbeit, wenn man nur zwei der sechs möglichen Felder nutzen will. Zum Beispiel bewirken die beiden folgenden Varianten dasselbe:

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

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

Im Folgenden sollen Befehle erklärt werden, die in den vorangegangenen Beispielen schon erwähnt, aber noch nicht besprochen wurde.

Um die Kolumnentitel möglichst simpel zu setzen, gibt es vordefinierte Befehle, die es vereinfachen lebende Kolumnentitel zu erzeugen.

`\leftmark`
`\rightmark`

Diese Befehle greifen auf die Kolumnentitel zurück, die für die linke bzw. für die rechte Seite gedacht sind. Für die Platzierung in den Seitenköpfen gibt es noch das Makro:

`\headmark`

Es unterscheidet selbstständig die Seiten und entspricht auf linken Seiten `\leftmark` und auf rechten Seiten `\rightmark`.

Die Inhalte können jedoch (wie im Stil `myheadings`) jederzeit manuell überschrieben werden. Hierfür werden Befehle wie `markboth` noch erweitert, um alle Felder ohne Einschränkung auch einzeln setzen zu können:

`\markleft{`*Inhalt links*`}`
`\markright{`*Inhalt rechts*`}`
`\markboth{`*Inhalt links*`}{`*Inhalt rechts*`}`
`\markdouble{`*Inhalt für beide Seiten*`}`

Die Seitenzahl wird über `\pagemark` gesetzt:

`\pagemark`

Dieser Befehl beinhaltet im Gegensatz zur direkten Ausgabe des Zählers auch die Anwendung des zugehörigen KOMAfont Elements (vgl. Unterabschnitt 5.1.4).

7.2.3 Formatierung der Kopf- und Fußzeilen

Eine Änderung der Schriftart innerhalb der Kopf und Fußzeilen funktioniert vollkommen analog zum Ändern der Schriftart der Überschriften (siehe Unterabschnitt 5.1.4), z. B.:

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

`headwidth=`**`\repl`**`{Breite}:`**`\repl`**`{Verschiebung}`
`footwidth=`**`\repl`**`{Breite}:`**`\repl`**`{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* (selbsterklärende Namen): 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 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=`*true/false*
`plainheadsepline=`*true/false*
`plainfootsepline=`*true/false*
`plainfootbotline=`*true/false*

Teil III

Alles außer Fließtext

8 Aufzählungen

Es gibt drei Umgebungen, um Aufzählungen vorzunehmen:

```
\begin{itemize} \item ...\end{itemize}
\begin{enumerate} \item ...\end{enumerate}
\begin{description} \item[Begriff 1] ...\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 nummeriert und bei der `description`-Umgebung erscheint der optionale Name, als zu erläuternder Begriff fett gedruckt.

```
\item[Labeltext]
```

Das `item`-Makro verarbeitet unabhängig von der Listenumgebung ein optionales Argument. Um manuelle Nummerierungen für alle Einträge einzeln vorzunehmen, sollte es jedoch nicht verwendet werden. Der eingefügte Text muss nicht weiter gruppiert werden und darf Absatzumbrüche enthalten.

Ein `\item` wird immer durch ein nachfolgendes Element der selben Ebene oder durch das Ende der Umgebung abgeschlossen.

Die Aufzählungsumgebungen können bis zu viermal ineinander verschachtelt werden und je nach Verschachtelungstiefe ändert sich das Markierungszeichen sowie die Einrückung am Anfang:

```
\begin{itemize}
\item 1. Stufe 1. Zeile
\begin{itemize}
\item 2. Stufe 1. Zeile
\begin{itemize}
\item 3. Stufe 1. Zeile
\begin{itemize}
\item 4. Stufe 1. Zeile
\item 4. Stufe 2. Zeile
\end{itemize}
\item 3. 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
    - 3. Stufe 1. Zeile
      - 4. Stufe 1. Zeile
      - 4. Stufe 2. Zeile
    - 3. Stufe 2. Zeile
  - 2. Stufe 2. Zeile
- 1. Stufe 2. Zeile

8.1 Aufzählungslabel ändern

Die Aufzählungszeichen können 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 Zähler (vgl. Abschnitt 4.1), `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 (Unterabschnitt 5.1.4) 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}
  \item[Begriff] ...
\end{labeling}
```

Das Trennzeichen ist beliebig und das Muster gibt die Einrücktiefe bei den Stichpunkten an. Genau wie bei den Standardaufzählugen beginnt jeder Punkt mit einem `\item`.

```
\begin{labeling}[~–]{description}
    \item[description] Begriffserklärung
    \item[enumerate] Nummerierte Auflistung
    \item[itemize] Nicht nummerierte Auflistung
    \item[labeling] Individualisierbare Auflistung
\end{labeling}
```

description – Begriffserklärung

enumerate – Nummerierte Auflistung

itemize – Nicht nummerierte Auflistung

labeling – Individualisierbare Auflistung

Für genauere Informationen und weitere Beispiele sei auf [14] verwiesen.

8.3 Eigene Auflistungen und Listenlayouts (Das enumitem-Paket)

Das Paket enumitem ist das Standard-Paket für Anpassungen von Listen.

```
\usepackage{enumitem}
```

Es definiert zu den drei Umgebungen ein zusätzliches optionales Argument. Hierüber können viele Anpassungen gemacht werden. Um ein möglichst einheitliches Dokument zu erhalten, empfiehlt es sich das Argument nicht zu verwenden, sondern eigene Listentypen zu definieren oder die Einstellungen global anzupassen.

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

Allerdings können diese zusätzlichen Optionen zum Beispiel dazu genutzt werden, um eine Liste zu unterbrechen und diese später fortzusetzen:

```
\begin{enumerate}
\item Punkt 1
\item Punkt 2
\end{enumerate}
Hier steht Text, der beide
Listen voneinander trennt.
\begin{enumerate}[resume]
\item Punkt 3
\item Punkt 4
\end{enumerate}
```

1. Punkt 1
2. Punkt 2

Hier steht Text, der beide Listen voneinander trennt.

3. Punkt 3
4. Punkt 4

Alle weiteren Befehle und Optionen finden sich in der Paketdokumentation [8].

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

Falls kein *Listenname* oder keine *Ebene* angegeben wird, werden die Optionen auf alle Listen bzw. alle Ebenen angewandt.

Eine mögliche Option ist `label`. Damit lässt sich die Beschriftung der Listenpunkte für die jeweilige Ebene ändern. Für die `itemize`-Umgebung kann man beliebige Symbole nutzen, wie z. B. einen Pfeil: → (Befehl: `$\rightarrow$`). Für die `enumerate`-Umgebung gibt es `\alph*`, `\Alph*`, `\arabic*`, `\roman*`, `Roman*`. Das Sternchen wird dabei automatisch durch den Zählernamen der

zugehörigen Ebene ersetzt.

Darüber hinaus kann die Ausrichtung oder aber auch die Abstände zwischen den einzelnen Einträgen sehr einfach mit dem Paket angepasst werden (vgl. ([8]).

Möchte man die Beschriftung dauerhaft ändern, so kann man die bestehenden Listen abwandeln oder zusätzliche erstellen:

```
\newlist{Listenname}{Typ}{maximale Schachtelungstiefe}
\renewlist{Listenname}{Typ}{maximale Schachtelungstiefe}
```

Für *Typ* wird entweder `enumerate`, `itemize` oder `description` eingesetzt. *Listenname* ist der Name der (neuen) Umgebung. Möchte man die Standardlisten neu definieren, so benutzt man stattdessen `\renewlist`.

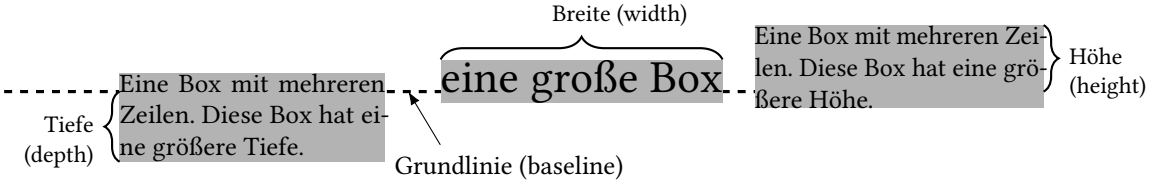
9 Das Prinzip der Boxen

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

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

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. Diese ermöglichen es den Platzhalter für die Box zu rekonstruieren. Die Größen sind die einzigen Informationen über die Box, die LATEX außerhalb der Box kennt:



9.1 Bearbeitungsmodi

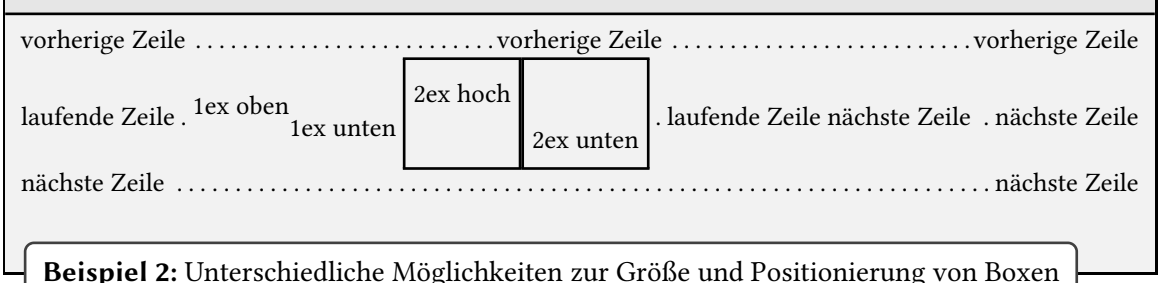
Innerhalb des Textkörpers kennt LATEX drei unterschiedliche Modi:

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

Mathematical Mode mathematischer Modus
z. B. `\begin{equation}...\end{equation}`, `$...$` (Kapitel 14)

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)

```
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
```



Beispiel 2: Unterschiedliche Möglichkeiten zur Größe und Positionierung von Boxen

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. Mathematische Befehle funktionieren zum Beispiel 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*}
`\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`

Wie man sieht, sind die Abstände jedoch 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.

Um weitere Einstellungen vorzunehmen, können die beiden Längen `\fboxrule` und `\fboxsep` verwendet werden, vgl. auch Abschnitt 4.2.

Die Länge `\fboxrule` definiert hierbei die Dicke des Rahmens, also die Linienstärke. `\fboxsep` hingegen definiert den Abstand zwischen dem Rahmen einer `\fbox` (oder `\framebox`) und ihrem Inhalt.

Für ein einheitliches Boxenlayout empfiehlt es sich daher zu Beginn die Definiton für alle Boxen vorzunehmen.

```
\fbox{text}
\setlength{\fboxrule}{2pt}
\fbox{text}
\setlength{\fboxsep}{5mm}
\fbox{text}
```

text text text

Positionierung von LR-Boxen

Um eine `\mbox` zu erzeugen gibt es außerdem die Möglichkeit einer `\raisebox`.

```
\raisebox{Offset}[Höhe][Tiefe]{Text}
```

Die darin enthaltenen Boxen (bzw. Zeichen) können allerdings 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 2).

Innerhalb des ersten Arguments können zusätzlich die Makros

```
\height
\depth
\width
```

für die Angabe der Verschiebung genutzt werden. Sie beziehen sich auf die Maße des Inhalts. Damit ist es möglich beispielsweise die Grundlinie einer Grafik an die Oberkante zu verschieben (vgl. Beispiel 3).

```
Baseline\rule{1cm}{1pt}
\raisebox{-\height}{%
    \includegraphics[width=5cm, height=1cm]{example-image}%
}
```

Baseline 

Beispiel 3: Raisebox zur Verschiebung der Grundlinie einer Grafik. Nützlich ist dies z. B. wenn Bilder oben bündig aneinenader ausgerichtet werden sollen.

```
\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 4: Positionsparameter für Absatzboxen

9.3 Absatzboxen

Es stellt sich auch die Notwendigkeit nach Boxen mit Zeilenumbruch. Hierfür gibt es die folgenden zwei Möglichkeiten `\parbox` und `minipage`. Je nach Anwendungsfall kann eine Umgebung gegenüber einem Makro mit Argument für den Inhalt Vor- oder Nachteil haben. Die Boxen unterscheiden sich ansonsten hauptsächlich durch das Verhalten von Fußnoten. Eine `minipage` bekommt einen eigenen Fußnotenblock am Ende der Box.

```
\begin{minipage}[Position][Höhe][vertikale Pos.]{Breite}
  …
\end{minipage}
```

```
\parbox[Position][Höhe][vertikal Pos.]{Breite}{Text}
```

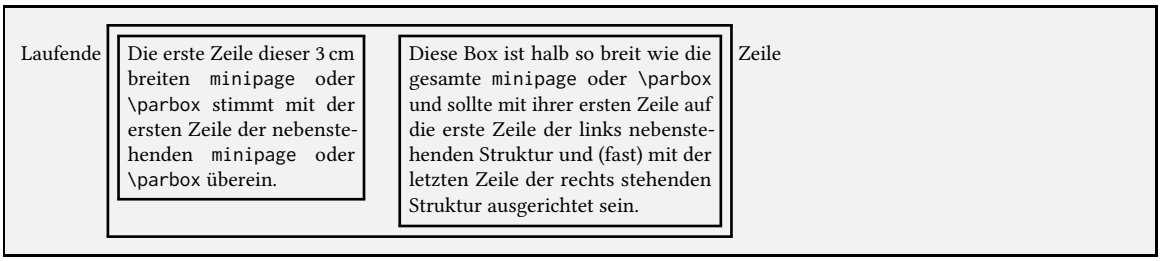
Als Position zur laufenden Zeile können die Optionen `t` (top), `b` (bottom), `c` (center) genutzt werden. Die vertikale Position innerhalb der Boxen verarbeitet die gleichen Parameter. Ein Beispiel der

Ausrichtung ist in Beispiel 4 gezeigt.

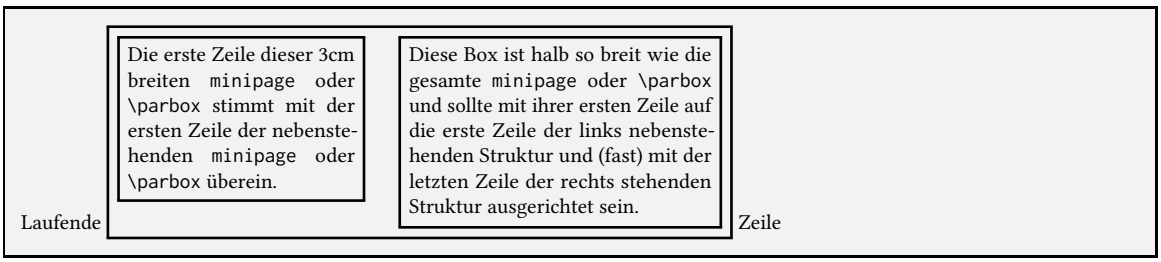
Innerhalb einer Box wird eine andere Box als eine Bearbeitungseinheit angesehen, also wie ein einzelnes Zeichen, was gerne zu überraschenden Ergebnissen führt. So bringt die Kombination

```
\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 durch eine zusätzliche Zeile. Sie bewirkt, dass die minipage zweizeilig wird und somit an der untersten Zeile ausgerichtet werden kann.



Erzeugt wird dies mit dem Code:

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

Gegebenenfalls ist ein weiterer Ausgleich notwendig sowie die Verwendung von \strut (vgl. Übungen).

9.4 Balkenboxen

Ein einfacher Balken wird mit

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

gebildet und kann unteranderem so aussehen:  (\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 etwas zu speichern, existieren ein Makro und eine Umgebung:

```
\savebox{\Boxname}[Breite][Position]{Code}
```

```
\begin{lrbox}{\Boxname}
  ...
\end{lrbox}
```

Anschließend kann die Box immer wieder eingefügt werden mit:

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

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

laufende Zeile Text laufende Zeile

Neben der Ausgabe der gesamten Box können auch die Maße gespeicherter Boxen einzeln ausgelesen werden. Das ist beispielsweise für dynamische Positionierungen oder komplexere Berechnungen bei der Erstellung von Templates hilfreich.

`\ht`*\Boxname*	Höhe
`\dp`*\Boxname*	Tiefe
`\wd`*\Boxname*	Breite

Hierbei handelt es sich um Längen, die wie Längen in Abschnitt 4.2 verwendet oder ausgegeben werden können.

10 Grafiken

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

```
\usepackage{graphicx}
```

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

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

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

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

Es besteht die Möglichkeit diese Optionen zu kombinieren, jedoch sollte dabei auf eine sinnvolle Kombination geachtet werden. Bei der Angabe von Breite und Höhe kann es zum Beispiel dazu kommen, dass das Bild verzerrt wird. Bei den Größenangaben ist es empfehlenswert das Bild mit dem Satzspiegel skalieren zu lassen, denn falls später die Ränder oder das Papierformat geändert werden, passt die Abbildung trotzdem.
Die möglichen Bildformate hängen hierbei vom verwendeten Compiler ab. Mit `lualatex` können `.pdf`, `.png` und `.jpg`-Dateien eingebunden werden. Bei Verwendung von `latex` ist man auf PostScript-Dateien, wie `.ps` und `.eps` beschränkt. Mit entsprechenden Zusätzen kann auch `lualatex` PostScript verarbeiten. In der TEX Live-Distribution ist nicht einmal ein Zusatzpaket notwendig. Bei MiKTEX benötigt man zusätzlich das Paket epstopdf.

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

Die Bilddatei sollte im selben Ordner wie das zu kompilierende Dokument liegen, ansonsten muss der Dateipfad oder Unterordner angegeben werden. Hier ist auch unter Windows ein Schrägstrich (/) zu verwenden.

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

11.1 Tabellen ohne zusätzliche Pakete

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

```
\begin{tabular}[Position]{Spaltendefinition}...\end{tabular}
\begin{array}[Position]{Spaltendefinition}...\end{array}
```

`array` ist die Variante für den Mathe-Modus (vgl. Kapitel 14). Die Funktionsweise ist jedoch identisch. Beide Umgebungen entsprechen einer Absatzbox, wie der `\parbox`. Sie erzeugen jeweils eine Tabelle in der 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. Ohne Zusatzpakete existieren die folgenden Spaltentypen:

`l`,`c`,`r`	links, rechts und zentriert
`p{`*Breite*`}`	`\parbox` mit der angegebenen *Breite* und Ausrichtung `t`

Um aus dem Code eine Tabelle bauen zu können, werden nun noch Befehle für den Spalten- und Zeilenwechsel benötigt:

`&`	Wechsel in die nächste Spalte; „alignment tab character“
`\tabularnewline`	Sofortige Beendigung der aktuellen Tabellenzeile, wie `\newline` nur für Tabellen
`\\`	Kurzform für `\tabularnewline`

Damit ist es bereits möglich eine komplette Tabelle zu bauen:

```
\begin{tabular}{lcr}
    l&c&r\\
    l2&c1&r1
\end{tabular}
```

l	c	r
l2	c1	r1

11.1.1 Linien

Linien können dabei helfen die Ausrichtung leichter zu erkennen. Es ist möglich Linien zwischen die einzelnen Zeilen einzuziehen. LaTeX selbst liefert hierfür die Makros:

```
\hline
\cline{Nummer der Anfangsspalte-Nummer der Endspalte}
```

Um eine vertikale Linie zu setzen, gibt es die Möglichkeit ein | zwischen den Spaltendefinitionen einzufügen. Aus typografischer Sicht sind allerdings vertikale Linien hinderlich für den Lesefluss und sollten daher vermieden werden.

In diesem Beispiel werden sie lediglich zur Illustration verwendet:

```
\begin{tabular}{l|cr}
    \hline
    l&c&r\\\hline
    l2&c1&r1\\\hline
\end{tabular}
```

l	c	r
l2	c1	r1

11.1.2 Spaltenzwischenräume überschreiben

Durch die Linien wird sichtbar, dass die Breite der Tabelle die Breite der Inhalte überragt. LaTeX fügt vor und nach jeder Zelle einen Abstand der Größe `\tabcolsep` (bei `array` `\arraycolsep`) ein.

Um den Text bündig mit dem Rand der Tabelle zu setzen wird das `@`-Zeichen verwendet:

```
@{Ersatzinhalt für den Spaltenzwischenraum}
```

Falls ein `@` im Argument für die Spaltenspezifikation platziert ist, wird der Zwischenraum durch den Inhalt des Arguments ersetzt.

Zwischen zwei Spalten werden beide Zwischenräume gemeinsam ersetzt. Wenn das Argument leer bleibt entfällt der Zwischenraum.

```
\begin{tabular}{@{}l|c@{-}r@{}}
    \hline
    l&c&r\\\cline{2-2}
    l2&c1&r1\\\hline
\end{tabular}
```

l	c – r
l2	c1–r1

11.1.3 Spalten vervielfältigen

Bei längeren Tabellen kann die Angabe eines einzelnen Spaltentypparameters pro Spalte schnell unübersichtlich werden. LaTeX stellt hierfür eine Syntax zur Verfügung, die es ermöglicht stattdessen eine Anzahl anzugeben:

```
*{Anzahl}{Spaltendefinition(en)}
```

Somit kann statt `ccc` auch `*{3}{c}` gesetzt werden. Das zweite Argument kann dabei auch mehr, als nur einen Buchstaben oder Spaltentyp enthalten.

11.1.4 Zellen über mehere Spalten – multicolumn

`\multicolumn{`*Spaltenanzahl*`}{`*Spaltenformatierung*`}{`*Text*`}`

Mehrere Spalten werden bei LaTeX über das `\multicolumn`-Makro zusammengefasst. Es kann außerdem die Spaltenformatierung für eine einzelne Zelle überschreiben.

Falls die überstehenden Spaltenzwischenräume über die `@`-Syntax entfernt wurden, müssen diese mit bei der Spaltenformatierung angegeben werden.

```
\begin{tabular}{@{}lcr@{}}
    links&zentriert&rechts\\\hline
    \multicolumn{2}{@{}c}{zentriert über zwei Spalten}&r
\end{tabular}
```

links	zentriert	rechts
zentriert über zwei Spalten		r

11.1.5 Tabellen mit fester Breite

`\begin{tabular*}[`*Position*`]{`*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}}
```

eingetragen werden. Alle darauf folgenden Spaltenzwischenräume werden entsprechend gestreckt. Anstelle von `\fill` können auch sonstige dehnbare Längendefinitionen verwendet werden.

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}|}
```

A	B	C	D

11.2 Weitere Spaltenformatierungen – Das array-Paket

Das array-Paket erweitert die Standard-Spaltentypen um ein paar praktische Ergänzungen.
Zum einen können andere Ausrichtungen für die Spalten mit fester Breite (p) gewählt werden:

m{*Breite*}	vertikal zentriert
b{*Breite*}	unten bündig

Außerdem erweitert es den Tabellenmechanismus und erlaubt, dass eine Formatierung mit in eine Spalte „geschoben“ werden kann.

>{*Inhalt/Einstellungen*}
<{*Inhalt/Einstellungen*}

Diese Syntax erlaubt es von links (>) oder von rechts (<) Code in eine Spalte zu schieben. Der Code wird dann zu Beginn oder am Ende jeder Zelle dieser Spalte ausgeführt.

Im Beispiel wird eine Spalte mit blauem Text erzeugt:

```
\begin{tabular}{>{\color{blue}}cc}
    Text&Text
\end{tabular}
```

Text Text

Wenn Spalten mit gleicher Formatierung öfters benötigt werden, ist es sinnvoll einen neuen Spaltentyp zu definieren:

```
\newcolumntype{Name}[Anzahl der Argumente]{Spaltendefinition}
```

Das optionale Argument der Argumentenanzahl wird wie bei \newcommand (Abschnitt 4.3) verwendet. Optionale Argumente sind nicht möglich.

Für obiges Beispiel, würde dies eine Spaltendefinition mit

```
\newcolumntype{B}{>{\color{blue}}c}
\begin{tabular}{Bc}
    Text&Text
\end{tabular}
```

bedeuten. Der neue Spaltentyp B kann dann wie die Standardtypen benutzt werden. Die Definition sollte aber am besten in der Präambel platziert werden.

11.3 Tabellen 2.0 – Das tabularray-Paket

Der Tabellenmechanismus in LaTeX basiert immer noch auf den Grundprinzipien, die TeX ursprünglich mal für Ausrichtungen implementiert hat. Mittlerweile sind die Ansprüche an Tabellen allerdngs extrem gewachsen. Zellen, die über mehrere Spalten laufen sind genauso üblich, wie Farben innerhalb von Tabellen. Lange wurden all diese Features durch einzelne Pakete ergänzt. Allerdings gibt es mittlerweile ein Paket, dass die meisten dieser Varianten direkt unterstützt ohne das zusätzliche Pakete notwendig und somit Konflikte vermieden werden.

```
\usepackage{tabularray}
```

Darüber hinaus ist tabularray von Jyu deutlich flexibler, da es den kompletten Tabellenmechanismus selbst implementiert [9]. Allerdings verarbeitet tabularray die gesamte Tabelle auf einmal. Das

beschränkt die Größe und sorgt für längere Kompilierzeiten. Bei sehr großen Tabellen ist daher der alte Weg noch sinnvoller. Diese sind in den folgenden Abschnitten beschrieben.

Die Beschreibung der Mechanismen ist hier momentan sehr minimal und wird wird noch ergänzt.

11.3.1 Die tblr-Umgebung

```
\begin{tblr}{Spaltendefinition}...\end{tblr}
```

Das Argument für die Spaltenformatierung erhält einen Spaltentypparameter pro Tabellenspalte. Es existieren die folgenden Spaltentypen:

`l`,`c`,`r`	links, rechts und zentriert
`p`{*Breite*}	Mehrzeilige Spalte mit der angegebenen *Breite* und Ausrichtung `t`
`X`	Dehnbare Spalte, die den übrigen Platz bis zur Tabellenbreite (normalerweise Zeilenbreite) auffüllt

Um aus dem Code eine Tabelle zu bauen werden nun noch Befehle für den Spalten- und Zeilenwechsel benötigt.

`&`	„alignment tab character“. Wechselt in die nächste Spalte
`\\`	wie `\newline` nur für Tabellen. Es beendet die aktuelle Tabellenzeile sofort.

Damit ist es bereits möglich eine komplette Tabelle zu bauen:

```
\begin{tblr}{lcr}
        l&c&r\\
        l2&c1&r1
\end{tblr}
```

l	c	r
l2	c1	r1

11.3.2 Linien

Teilweise können Linien dabei helfen die Ausrichtung leichter zu erkennen. Es ist möglich Linien zwischen die einzelnen Zeilen einzuziehen. LaTeX selbst liefert hierfür die Makros

```
\hline
\cline{Nummer der Anfangsspalte-Nummer der Endspalte}
```

Zusätzlich existiert die Möglichkeit ein | zwischen den Spaltendefinitionen einzufügen um eine vertikale Linie zu setzen. Allerdings sind vertikale Linien aus typografischer Sicht für den Lesefluss hinderlich und sollten daher nicht genutzt werden. Hier werden sie lediglich zur Illustration verwendet:

```
\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 5: Die tabularx-Umgebung

```
\begin{tblr}{l|cr}
        \hline
        l&c&r\\\hline
        l2&c1&r1\\\hline
    \end{tblr}
```

l	c	r
l2	c1	r1

11.4 Variable Spaltenbreite – Das tabularx-Paket

```
\usepackage{tabularx}
```

Wird dieses Paket benutzt, so wird automatisch auch das array-Paket geladen. Weil alle Befehle des array-Pakets bereitstehen (siehe Abschnitt 11.2), muss dieses nicht direkt geladen werden.

```
\begin{tabularx}[Position]{Breite}{Spaltendefinition}
  …
\end{tabularx}
```

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 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.

11.5 Variable Spaltenbreite mit Ausrichtung – Das tabulary-Paket

```
\usepackage{tabulary}
```

Das tabulary-Paket liefert ähnlich dem tabularx-Paket (siehe Abschnitt 11.4) eine Tabellenumgebung mit der man die Breite der Tabelle festlegen kann. Auch hier wird das array-Paket automatisch geladen.

```
\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 6: Die `tabulary`-Umgebung

```
\begin{tabulary}{maximale Breite}{Spaltenformatierungen}
  …
\end{tabulary}
```

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.

11.6 Verbesserte Tabellenlayouts – Das booktabs-Paket

Der Standard-LaTeX-Befehl `\hline` für horizontale Linien erzeugt ungleiche Abstände. Zusätzlich sind die Abstände sehr knapp bemessen.

```
\usepackage{booktabs}
```

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 Abgrenzungslinien 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*){*Nummer der Anfangsspalte-Nummer der Endspalte*}

Optisch entspricht diese Linie einer 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 zum Zelleninhalt abschließt.

11.7 Tabellenpakete als Ergänzung zur klassischen Tabellensyntax

Da tabularray noch relativ neu ist, finden sich gerade in älteren Dokumenten noch viele Tabellen, die mit Ergänzungpaketen erzeugt wurden. In diesem Abschnitt finden sich die Erläuterungen zu den Ergänzungen. Falls möglich sollte dennoch tabularray verwendet werden, da es deutlich mehr Anpassungsmöglichkeiten bietet.

11.7.1 Zellen über mehrere Zeilen – Das multirow-Paket

\usepackage{multirow}

Das multirow-Paket ermöglicht es, dass ähnlich zu \multicolumn mehrere Zellen zusammengefasst werden. In diesem Fall jedoch über mehrere Zeilen anstatt mehrerer Spalten.

\multirow[*vpos*]{*Zeilenanzahl*}[*bigstruts*]{*width*}[*vmode*]{*Inhalt*}

\multirow wird immer in der Zeile in der die mehrzeilige Zelle beginnt ausgeführt. Das Argument der Zeilenanzahl ist zwingend, die Breite kann auch als * angegeben werden. Dann wird die natürliche Breite der Spalte verwendet.

Common g text	Column g2a Column g2b Column g2c Column g2d
test Common g text test2	Column g2a
	Column g2b
	Column g2c
Common g text, but a bit longer.	Column g2a
	Column g2b
	Column g2c
	Column g2d
Common g text	Column g2a Column g2b Column g2c Column g2d

11.7.2 Mehrseitige Tabellen – Das longtable-Paket

```
\usepackage{longtable}
```

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 Position]{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 durch `\\*` 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. Der Seitenumbruch kann ansonsten nicht immer besonders platzsparend gelingen. LaTeX bietet hier verschiedene Möglichkeiten. Wenn genug Platz für das Objekt vorhanden ist, dann wird es an den Ort seiner Erscheinung entsprechend dem Quellcode gesetzt. Wenn dies nicht möglich ist, dann wird der Text weitergeführt und das Objekt an geeigneter Stelle neu positioniert.

Dies alles geschieht mit:

[1] Diese Fußnote hat keinen Textbezug. Sie dient lediglich der Veranschaulichung der Standard-Positionierung von Fußnoten im Zusammenhang mit unten auf der Seite positionierten Gleitobjekten (Positionsparameter b). Für eine genauere Beschreibung dieses Sachverhaltes siehe den entsprechenden Absatz auf Seite 101.

Abbildung 12.1: Jo, der persöhnliche Assistent und Haustier der Kursleiterin

```
\begin{table}[Positionsparameter]
eigentliche Tabelle
\caption{Tabellenunterschrift}
\label{Markername}
\end{table}
```

```
\begin{figure}[Positionsparameter]
eigentliche Tabelle
\caption{Tabellenunterschrift}
\label{Markername}
\end{figure}
```

Bei der Verwendung einer zweispaltigen Formatierung ist es sinnvoll die Sternchenvariante der Umgebungen zu verwenden. Damit kann ein Gleitobjekt über beide Spalten hinweg angelegt werden.

Der optionale Positionsparameter kann als eine Präferenzliste aus `t` „top“, `b` „bottom“, `h` „here“ oder `p` „page“ (extra Seite) gesetzt werden. In der Sternchen-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` und 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 den Positionsangaben verstärkt den Positionierungswunsch des Autors und versucht die gewünschte Position zu erzwingen. Um Warnungen vorzubeugen sollte in jeder Liste, 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 (Unterabschnitt 4.4.1). Ein einfaches Beispiel für ein Bild ist Abbildung 12.1. Wenn man jedoch die Bildbeschreibung neben dem Bild positionieren möchte, dann gibt es die beiden Möglichkeiten in Abbildung 12.2 und Unterabschnitt 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 der 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 an die Vorgaben der Platzierungsparameter. Tabelle 12.1 zeigt die wichtigsten. Eine komplette Liste findet sich zum Beispiel in [16, S.171ff.].
- Im Fall der Kombination `ht` ist `h` von höherer Priorität. Das Gleitobjekt wird an dem 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.

Tabelle 12.1: Parameter für die Positionierung von Gleitobjekten. Zähler können mithilfe der in Abschnitt 4.1 besprochenen Befehle manipuliert werden. Makros können über `\renewcommand` geändert werden (Abschnitt 4.3). Zulässig sind Werte < 1. Eine Zusammenfassung aller Gleitobjektparameter findet sich in [18, 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`, aber für die Positionierung unten auf der Seite (Standardwert: 1)
`totalnumber`	Zähler (Standardwert: 3), welcher die maximale Anzahl der Gleitobjekte pro Seite angibt
`\.7`	Maximaler Seitenbruchteil für `t`-Platzierungen (Standardwert: 0.7)
`\.3`	Maximaler Seitenbruchteil für `b`-Platzierungen (Standardwert: 0.3)
`\.2`	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.

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 derselben 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 aus Text. Sie gehören also eher zum Textkörper als Gleitobjekte und werden deshalb auch näher zu diesem gesetzt.

Möchte man dieses Verhalten unterbinden, weil beispielsweise 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 [23] verwiesen.

12.1 Bildpositionierung einschränken

Die Regeln, welche LaTeX für die Positionierung von Gleitobjekten beachtet, liefern meistens sehr gute Ergebnisse. Es kann allerdings passieren, dass für zu viele Gleitobjekte nicht ausreichend Text im Dokument vorhanden ist. Dadurch sind die Möglichkeiten zur Platzierung von Objekten beschränkt.

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

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

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

Tabelle 12.2: Auswahl der 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`

`\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}
```

können Autor*innen mit dem Befehl

`\FloatBarrier`

Barrieren setzen, welche Gleitumgebungen nicht überwinden können. Sollen Gleitobjekte immer innerhalb des aktuellen Abschnitts platziert werden, so kann placeins bei jeder `\section` automatisch eine Barriere setzen. Dafür muss die Option `section` beim Laden des Pakets angegeben werden.
`\FloatBarrier` geht sehr strikt vor. Dadurch könnten große Lücken im Satz entstehen. Es existiert mit den Optionen `above` und `below` die Möglichkeit das etwas zu lockern. Dann können Objekte noch auf derselben Seite wie die Barriere ausgegeben werden. Dennoch sollte es in jedem Fall nur genutzt werden, wenn es absolut keine andere Möglichkeit gibt.

12.2 Gleitobjektbeschriftungen anpassen

12.2.1 Position und Abstände

LaTeX geht beim Befehl `\caption` immer von einer *Unter*schrift der Gleitumgebungen aus. Große Tabellen sollten aber eine *Über*schrift aufweisen, damit die Leser:in zu Beginn weiß, worum es sich auch in der nachfolgenden Tabelle handelt. Die `\caption` kann grundsätzlich einfach über dem Objekt platziert werden. Allerdings passen dann die Abstände nicht richtig.

```
\captionabove[Verzeichniseintrag]{Text}
\captionbelow[Verzeichniseintrag]{Text}
```

Die Makros \captionabove/\captionbelow ermöglichen für ein einzelnes Objekt die Abstände entsprechend zu ändern, dass sie passend gesetzt werden. Allerdings ist es nicht sinnvoll für jede Beschriftung einzeln zu entscheiden wie sie gewählt werden sollen.
Üblicherweise werden alle Beschriftungen unter oder über den Objekten platziert. Es ist höchstens noch eine Unterscheidung nach Objekttyp möglich (z. B. Abbildungen haben Unterschriften, Tabellen erhalten die Beschriftung über dem Objekt).

Um diese Einstellungen global zu tätigen liefert KOMA-Script die Option captions. Tabelle 12.2 zeigt mögliche Werte. Das oben erwähnte verhalten wird demnach mit der Option vaptions=tableheading erwirkt. Dann haben Beschriftungen in Tabellen die Abstände einer Über- und in Abbildungsumgebungen die einer Unterschrift.

```
\documentclass[captions=Option]{scr...}
```

Beschreibung ohne Gleitobjekt

\caption hängt immer mit dem äußeren Gleitobjekt zusammen. Dies liegt daran, dass das Makro außerhalb einer Umgebung nicht weiß, welche Bezeichnung verwendet werden soll (Tabelle oder Abbildung).

Falls Abbildungen nicht als Gleitobjekt gesetzt werden sollen, weil beispielsweise das gesamte Anhangskapitel nur aus Abbildungen besteht, können Objekte mithilfe des Makros

```
\captionof{Objekttyp}[Verzeichniseintrag]{Beschriftung}
```

auch ohne ein umgebendes Gleitobjekt beschriftet werden. Es existieren analog zu den Makros \captionabove/below auch die Varianten \captionaboveof/belowof.
Es kann allerdings dann theoretisch ein Seitenumbruch zwischen Abbildung und Objekt stattfinden.

13 Verzeichnisse

13.1 Abbildungs- und Tabellenverzeichnis

Analog zum Inhaltsverzeichnis werden Abbildungs- und Tabellenverzeichnisse mit den Makros

```
\listoffigures
\listoftables
```

erstellt. Die zugehörigen Hilfsdateien haben die Endung `.lof` (Abbildungen) und `.lot` (Tabellen). Wie beim Inhaltsverzeichnis wird somit wieder zweifaches Kompilieren benötigt.
Die Verzeichnisse erzeugen normalerweise eine Überschrift der höchsten Ebene ohne Nummerierung und werden auch nicht im Inhaltsverzeichnis eingetragen. Um dieses Verhalten zu ändern stellt KOMA-Script die Option `listof` zur Verfügung. Da die Werte einige Varianten haben sei hier lediglich auf die Dokumentation [12] verwiesen. Dort findet sich auch die Dokumentation des Pakets tocbasic, mit welchem die Verzeichnisse erzeugt werden. Damit sind auch Layoutanpassungen sowie die Manipulation der Einträge möglich.

13.2 Literaturverzeichnis

13.2.1 Manuelle Erstellung des Literaturverzeichnisses

Ein Literaturverzeichnis kann manuell erzeugt werden mit der Umgebung:

```
\begin{thebibliography}{Mustermarke}
  \bibitem[Marke1]{Bezug1} Eintragstext1
  \bibitem[Marke2]{Bezug2} Eintragstext2
  \bibitem[Marke3]{Bezug3} Eintragstext3
  ...
\end{thebibliography}
```

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).

```
\cite[Informationen]{Marke1, Marke2}
```

An der Stelle im Text an der man den Literaturverweis haben möchte, schreibt man entsprechend:

```
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 [5] und dem Buch von Helmut Kopka [15, 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 können noch weitere beliebige Angaben, wie Seitenzahl oder Kapitel zur Verfügung gestellt werden.

Oft werden beim Erstellen eines Dokuments entsprechende Literaturverweise ergänzt und 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.

13.2.2 Automatisches Literaturverzeichnis mit Biber und dem biblatex-Paket

LaTeX bietet eine bequeme Möglichkeit die verwendeten Literaturzitate, die der Autor LaTeX in einem `.bib`-File mitteilt mit einer Literaturdatenbank zu synchronisieren. 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.

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}
\addbibresource{.bib-File ohne Endung}
```

Neben der Benutzung von Biber und biblatex steht auch noch das Programm BibTeX zur Verfügung. Weil es das ältere von beiden ist, ist BibTeX weiter verbreitet. Biber und das Paket biblatex haben jedoch wesentliche Vorteile, weshalb hier auch ihre Verwendung empfohlen wird. Zum einen ist Biber vollständig unicodefähig. Es 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.

Zusätzlich zum Paket selbst muss auch die Literaturdatenbank geladen werden. Dies geschieht durch die Angabe des Dateipfades mit dem `\bibliography` (falls sich die Datei im selben Ordner befindet genügt der Name). Die Dateiendung `.bib` muss dabei nicht angegeben werden. Allerdings ist zu beachten, dass wenn es sich um mehrere Dateien handelt, die Namen derer durch Kommata und 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, auf die im Text entweder mit einem gedrucktem oder unterdrücktem Bezug verwiesen werden abgedruckt.

`\cite[vorher][nachher]{Bezug}`

Das Makro `\cite` setzt dabei einen Bezug zum betreffenden Eintrag des Literaturverzeichnisses.

`\cite[vgl.][S. 345]{einfuehrung}`	[vgl. 27, S. 345]

Vorher und *nachhher* sind dabei Textbausteine, die z. B. noch zusätzlich auf einen bestimmten Teil der Literatur, also eine Seite oder ein Kapitel verweisen.

`\nocite*{Bezug}`

`\nocite*` setzt dahingegen einen stummen Bezug. Dieser Bezug erscheint nicht und ermöglicht es aber 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 mit dem zusätzlichen Sternchen.

Damit die `.aux`- und `.bcf`-Dateien aktualisiert werden, beginnt das Ausführen von Biber ganz normal mit dem Starten des LaTeX-Interpreters. Anschließend wird Biber gestartet (erzeugt eine `.bbl`-Datei) und dann muss noch zweimal kompiliert werden, 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. Die meisten LaTeX Editoren sind jedoch leider noch auf das BibTeX-Programm voreingestellt. Eventuell ist hier eine Anpassung in den Editor-Einstellungen notwendig.

BibTeX-Datenbank

In der `.bib`-Datei stehen alle Informationen über das zitierte Werk. Der Datentyp hierfür heißt leider wie das alte BibTeX-Programm. Das ist einerseits ein Vorteil, weil alte Daten weiter verwendet werden können, sorgt jedoch hin und wieder für Verwirrung. In diesem Abschnitt wird Beispielhaft das Format gezeigt. Allgemein ist es jedoch hilfreich eine Literaturverwaltungssoftware zu nutzen. Als kostenfreie OpenSource Lösung steht hier beispielsweise JabRef [7] für alle Betriebssysteme zur Verfügung.

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 von-bis},
    MONTH = {Erscheinungsmonat}
}
@Typ{Bezug2,
    AUTHOR = {...
```

Die Angabe der Einträge ist dabei nicht zwingend notwendig. Biber erwartet jedoch je nach *Typ* und Stil bestimmte Angaben. Die Groß- und Kleinschreibung spielt bei den Bezeichnern „AUTHOR“, „TITLE“ usw. keine Rolle. Als Beispiel dient wieder [27].

```
@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 [21] 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 [21]. Darüber hinaus kann man über das `texdoc`-Programm oder CTAN[1] 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}
```

[1] https://www.ctan.org/topic/biblatex

Tabelle 13.1: 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`	Auflistung nach Zitierreihenfolge, unsortiert
`nty`	Name, Titel, Jahr
`nyt`	Name, Jahr, Titel
`nyvt`	Name, Jahr, Volume, Titel
`anyt`	Label alphabetisch, Name, Jahr, Titel
`anyvt`	Label alphabetisch, Name, Jahr, Volume, Titel
`ynt`	Jahr, Name, Titel
`ydnt`	Jahr absteigend, Name, Titel

Man bekommt 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. Man kann sich aber auch sein eigenes Schema basteln. Hier sei wieder auf die Paketdokumentation verwiesen.

```
sorting = \repl{Sortieroptionen}
```

In Tabelle 13.1 findet man die möglichen *Sortieroptionen*.

Zitierbefehle

Mit dem biblatex-Paket gibt es mehrere Zitierbefehle. Diese haben immer die Struktur:

```
\cite[vorher][nachher]{Bezug}
```

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 nutzen. Mögliche Zitierbefehle siehe Tabelle 13.2. Das zweite optionale Argument ergänzt zudem die Angabe „S. “, falls nur eine Zahl enthalten ist. Diese Angabe ist babel-Sprachabhängig.
Weitere nützliche Paketoptionen findet man in Tabelle 13.3.

Wie man kleine Formatierungen der Zitate und der Einträge im Literaturverzeichnis vornimmt soll anhand eines kleinen Beispiels gezeigt werden:

Tabelle 13.2: 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*`}`	Erzeugung der Literaturangabe und Fußnote
`\autocite[vorher][nachher]{`*Bezug*`}`	Unterschiedliche Ausgabe abhängig vom Zitierstil, siehe Paketoption autocite
`\textcite[vorher][nachher]{`*Bezug*`}`	Verfügbarkeit in allen nicht-verbose Stilen, gibt Autoren gefolgt vom Zitierschlüssel aus
`\supercite{`*Bezug*`}`	Verfügbarkeit in numerischen Stilen, ergibt hochgestellte Zitatnummer ohne Klammern
`\nocite{`*Bezug*`/*}`	Hinzufügung key/aller Einträge zu Bibliographie ohne Zitat zur Datenbank
`\fullcite[vorher][nachher]{`*Bezug*`}`	Einfügung des kompletten Literaturverweis wie aus Bibliographie
`\citeauthor[vorher][nachher]{`*Bezug*`}`	Ausgabe Autoren, Editoren oder Übersetzer
`\citetitle[vorher][nachher]{`*Bezug*`}`	Ausgabe (wenn vorhanden) von `shorttitle` ansonsten `title`
`\citeyear[vorher][nachher]{`*Bezug*`}`	Ausgabe Erscheinungsjahr
`\citedate[vorher][nachher]{`*Bezug*`}`	Ausgabe volles Datum
`\citeurl[vorher][nachher]{`*Bezug*`}`	Ausgabe url-Feld

Tabelle 13.3: Weitere Paketoptionen für das Paket biblatex

`autocite`	`plain`	Verhalten wie `\cite`
	`inline`	Verhalten wie `\parencite`
	`footnote`	Verhalten wie `\footcite`
	`superscript`	Verhalten wie `\supercite`
		Standardwert hängt vom Zitierstil ab
`backref`	`true/false`	Anfügung der Seitenzahlen der Zitate an die Bibliographieeinträge
`backend`	`biber`	Unterstützung von ASCII bis UTF-8, on-the-fly rencoding, Sortierungsregeln über sortlocale, sortcase, sortupper einstellbar
	`bibtex`	BibTeX traditionell, unterstützt nur ASCII, Sortierung ist immer abhängig von Groß-/Kleinschreibung
`heading`	`label`	Angabe eigener Bibliographieüberschriften, Definition 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 und danach immer ein `spadesuit`-Symbol. Bei den Zitaten im Text werden die Buchtitel mit Sternchen umschlossen. Die Titel der Artikel stehen in eckigen Klammern in den Zitaten. Bei der Angabe mehrerer Autoren werden diese mit einem Pluszeichen getrennt. Außer vor dem letzten wo hier „und manchmal“ steht.

Eine komplette Liste der formatierbaren Attribute findet man in der Paketdokumentation.

13.3 Stichwortverzeichnis

Die Indexerstellung bei LaTeX funktioniert analog zum Literaturverzeichnis. Es gibt eine manuelle Variante, die jedoch sehr aufwendig ist und es gibt die automatische Indexerstellung über das Zusatzprogramm `makeindex`.

```
\makeindex
\index{Eintrag}
```

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.

```
\usepackage{makeidx}
```

Zum Ausdrucken und zur Formatierung des Index benutzt man üblicherweise das Paket `makeidx`. Dies ermöglicht eine einfache Ausgabe (siehe Beispiel 7) durch das Makro:

```
\printindex
```

Analog zur Bibliografie muss hierbei zunächst einmal mit dem LaTeX-Compiler kompiliert werden. Anschließend lässt man `makeindex` laufen, bevor wiederum zwei LaTeX-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

```
\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 7: Einfache Indexerstellung mit makeidx. In Anlehnung an [27, Beispiel 10-7-02]

Tabelle 13.4 [vgl. 27, S. 512].
Im Zusammenhang mit den `index`-Makro stellen die folgenden Makros Beispiele für die Verwendung der Syntax dar:

Tabelle 13.4: 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{`*anderer Eintrag*`}`.
`|(`	Die Seitenzahlen des Indexeintrages werden bis zu einem Eintrag mit `|)` zusammengefasst.
`|)`	Gegenstück zu `|(`

```
\index{Eintrag}
\index{Haupteintrag!Untereintrag}
\index{Eintrag|(}% Muss zusammen mit |) gesetzt werden
\index{Eintrag|)% Eintrag bekommt dann einen Seitenbereich zugewiesen
\index{Sortiereintrag@\textbf{Eintrag}}
\index{Eintrag|see{anderer Eintrag}}
```

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) $18°21′4″$
- 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 (`$1{,}11$` entspricht 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: $f\colon D \to Z$ anstatt $f : D \to Z$
- Folgt auf den Doppelpunkt ein Gleichheitszeichen, so kann man das mit dem Symbol $\coloneqq$ (`\coloneqq`) aus dem mathttools-Paket setzen. Bei der Zeichenfolge `:=` ist der Doppelpunkt nicht ganz zur Achse des Gleichheitszeichens zentriert. Benutzt man das mathtools-Paket, kann man auch einfach den Doppelpunkt durch das Makro `\mathtoolsset{centercolon}` generell zentrieren lassen. Damit erreicht man dasselbe Ergebnis mit der Kurzschreibweise.

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 Unterabschnitt 5.6.2). Es sind kleine Unterschiede mit großer Wirkung:

```
Falsch: a-b+2*c=D \hfill $\leftrightarrow$ \hfill 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:

```
\newcommand*{\MStyleExample}{
    \mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}}
    \int_{-\infty}^{+\infty}f(t)\mathrm{e}^{-\mathrm{i}\omega t}
    \,\mathrm{d}t
}
```

```
$\displaystyle\MStyleExample$\\
$\textstyle\MStyleExample$\\
$\scriptstyle\MStyleExample$\\
$\scriptscriptstyle\MStyleExample$
```

$$\mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$
$$\mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$
$$\mathrm{FT}(f)=\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$
$$\mathrm{FT}(f)=\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t$$

Beispiel 8: Mathematische Schriftstile im Vergleich
Als Abkürzung für den Beispielcode dient `\MStyleExample`

`\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-Paket, Frakturschrift $\mathfrak{ABCabc}$
`\boldsymbol{`*Zeichenfolge*`}`	benötigt amsmath-Paket, setzt auch griechische Buchstaben und Symbole fett, erzeugt jedoch unschöne Abstände
`\boldmath{`*Zeichenfolge*`}`	Schalterbefehl, muss vor der Matheumgebung angewendet werden; (fast) alle Zeichen fett
`\unboldmath{`*Zeichenfolge*`}`	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 8). Man nennt diese Befehle bewusst „styles“ und nicht „Größen“, da sich auch die Form der Zeichen unterscheidet. Beispielsweise ist in Textformeln die Größe von Zähler und Nenner kleiner, da dies für den Zeilenabstand günstiger ist, als wenn sie nicht auf bzw. unter dem Bruchstrich stünden. 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. Die beiden äußeren Zeilen werden allerdings 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, dann lässt sich dies eigentlich nicht vermeiden: $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$
Es gibt jedoch auch 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 (Unterabschnitt 14.4.2) zu setzen, um das Layout nicht zu stören.

14.4.1 Zeilenmodus

Der Zeilenmodus kann durch drei verschiedene Umgebungen aktiviert werden:
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 sinnvoller die LaTeX-Version zu nutzen, da die Fehlermeldungen in diesem Fall verständlicher sind.

Neben dem manuellen Mathemodus existiert noch ein Makro, das den Mathemodus erzwingt:

`\ensuremath{`*Code*`}`

Dieses Makro testet zuerst, ob man sich gerade im Mathemodus befindet. Ist dies der Fall wird das Argument eins zu eins übernommen. Sollte dies jedoch nicht zutreffen, wird vorher in den Mathemodus und anschließend wieder zurück gewechselt.

Vor allem wird `\ensuremath` innerhalb von Makrodefinitionen benötigt, wie zum Beispiel `\SI` (vgl. Abschnitt 14.17). Die sollten in beiden Modi verwendbar sein.

14.4.2 Abgesetzter Modus

Einzeilige Umgebungen
Analog zur `\(\)`-Struktur gibt es eine abgesetzte Variante. Daneben existieren noch richtige Umgebungen[1], die allerdings etwas mehr Schreibarbeit bedeuten:

[1] Zusätzlich existiert noch eine für TeX gültige Makrokombination: `$$..$$`. Diese Syntax ist jedoch veraltet und aus verschiedensten Gründen zu vermeiden!

```
\begin{displaymath}Formelcode\end{displaymath}
\begin{equation}Formelcode\end{equation}
\begin{equation*}Formelcode\end{equation*}
```

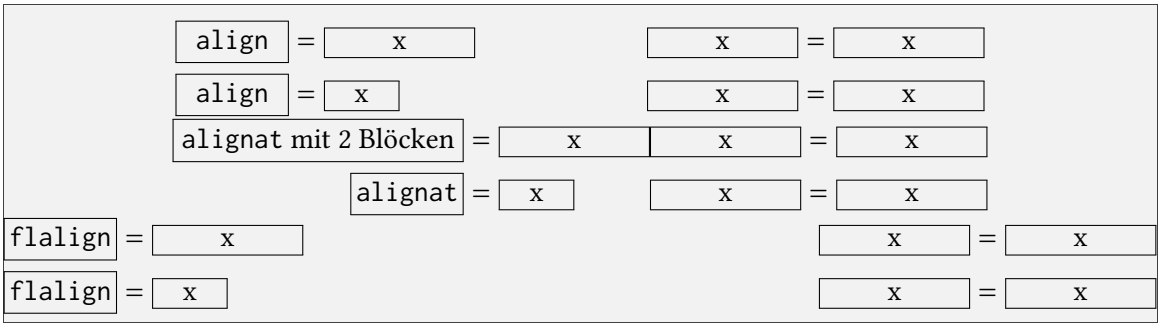
Mehrzeilige Umgebungen
Es gibt drei verschiedene Arten von `align`-Umgebungen.

```
\begin{align}Formelcode\end{align}
\begin{flalign}Formelcode\end{flalign}
\begin{alignat}{Anzahl der Blöcke}Formelcode\end{alignat}
```

Alle drei existieren auch in einer Sternchenform mit unterdrückter Nummerierung. Will man die Nummer lediglich für einzelne Zeilen unterdrücken, so setzt man vor den Zeilenumbruch den Befehl:

```
\nonumber
```

Er unterdrückt die Nummerierung der aktuellen Formelzeile.
Die Darstellung der `align`-Umgebungen entspricht folgendem Prinzip:



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* das Gleichheitszeichen bzw. das Relationssymbol gesetzt werden. Dadurch wird erreicht, dass die Ordnungssymbole alle direkt untereinander sind. 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. Sie kann allerdings geschachtelt und somit innerhalb einer beliebigen anderen Matheumgebung angewendet

```
\[
\left.
\begin{aligned}
2x+3 &=7&       2x+5-5&=7-3\\
2x   &=4& \frac{2x}{2}&=2\\
x    &=2
\end{aligned}
\right\}
\]
```

$$\left.\begin{aligned} 2x+3 &= 7 & 2x+5-5 &= 7-3 \\ 2x &= 4 & \frac{2x}{2} &= 2 \\ x &= 2 \end{aligned}\right\}$$

Beispiel 9: Die `aligned`-Umgebung (auf die Struktur der Klammern sowie der Makros `\left` und `\right` wird in Abschnitt 14.12 genauer eingegangen)

werden. Man kann mit ihrer Hilfe erreichen, dass zum Beispiel drei Gleichungen nur eine einzelne gemeinsame Nummer erhalten, siehe Beispiel 9.

```
\begin{gather}Formelcode\end{gather}
\begin{gathered}Formelcode\end{gathered}
```

Die `gather`-Umgebung zentriert Formeln 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 bei `gather` einfach nur zentriert.

```
\begin{multlined}Formelcode\end{multlined}
```

Die `multline`-Umgebung ist ebenfalls eine mehrzeilige Umgebung, die jedoch insbesondere für sehr lange Gleichungen verwendet wird:

```
\begin{multline}
links\\
zentriert\\
zentriert\\
zentriert\\
rechts
\end{multline}
```

$$\begin{multline} \text{links} \\ \text{zentriert} \\ \text{zentriert} \\ \text{zentriert} \\ \text{rechts} \end{multline} \tag{14.1}$$

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 einen ordentlichen Formelsatz einfacher zu erreichen, existiert der Befehl:

```
\phantom{Phantomtext}
```

Er setzt eine leere Box in der Größe des „Phantomtextes“. Somit kann man einen Abstand der Breite

```
\(|a| =
\begin{cases}
    \phantom{-}a;&a\geq 0\\
    -a;&a<0
\end{cases}
\)
```

$$|a| = \begin{cases} a; & a \geq 0 \\ -a; & a < 0 \end{cases}$$

anstatt

$$|a| = \begin{cases} a; & a \geq 0 \\ -a; & a < 0 \end{cases}$$

Beispiel 10: Manipulierte Ausrichtung mithilfe von phantom

eines Objektes einfügen, siehe Beispiel 10.

14.5 Referenz und Bezug

Querverweise innerhalb von Formeln können, wie auch im laufenden Text mit `\label` und `\ref` gesetzt werden. Zusätzlich liefert jedoch das amsmath-Paket den Befehl:

`\eqref{`*Markername*`}`

Dieser setzt Klammern um die Zahl (vgl. (1.1) statt vgl. 1.1) und kennzeichnet den Bezug somit unverwechselbar als Formel. Zusätzlich bietet sich noch die Möglichkeit die Beschriftung manuell zu generieren und zwar mit dem Befehl:

`\tag{`*Beschriftung*`}`

Außerdem existiert eine Sternchenform `\tag*`, bei der die Klammern um die *Beschriftung* entfallen.

```
\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.

14.6 Mathematische Akzente

Die in Unterabschnitt 5.6.3 beschriebenen 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“. Platziert werden können sie jedoch auf allen Buchstaben.

Zusätzlich zu den einfachen Befehlen `\hat` und `\tilde` gibt es auch breitere Versionen, die über mehrere Buchstaben gesetzt werden können. So bewirkt zum Beispiel das Makro `\widehat\{1-x\}`

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}`				

die Zeichenkombination $\widehat{1-x}$. `\widetilde` funktioniert analog.

14.7 Exponenten und Indizes

Innerhalb der Matheumgebungen geht mittels ^ Hochstellen von Zahlen und Buchstaben und mit _ Tiefstellen. Auch mehrfache Indizes mit beliebiger Tiefe sind möglich. Soll mehr als ein Zeichen hochgestellt werden, muss die Zeichenfolge in geschweifte Klammern gesetzt werden. Ansonsten 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 a_n+ x_i^n+ x^{2n} x_{2y} $A_{i,j,k}^{-n+2}$ $x^{y^{z^2}}$ x^{y_1} $A^{x_i^2}_{j^{2n}_{n,m}}$

Darüber hinaus ist es mit dem mathtools-Paket auch möglich Indizes auf der linken Seite von Zeichen oder Ausdrücken zu setzen.

`\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. Da die Abstände und Ausrichtungen in diesem Fall jedoch formal nicht korrekt sind, sollte 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. Dann befinden sich alle Indizes auf gleicher Höhe:

```
$a_l^2 b_m^{} c_n^2$\qquad
$a_l^2 b_m c_n^2$
```

$a_l^2 b_m c_n^2$ $a_l^2 b_m c_n^2$

14.8 Brüche und Binomialkoeffizienten

Um Brüche darzustellen wird folgender Befehl verwendet:

`\frac{`*Zähler*`}{`*Nenner*`}`

```
\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 11: Brüche

Dieser Befehl setzt den Bruch und erzeugt einen Bruchstrich der die Länge des Zählers oder des Nenners hat. Je nachdem, welcher breiter von beiden ist (siehe Beispiel 11).

Das amsmath-Paket liefert das Makro `\genfrac` („generalized fraction“). Es benötigt sechs Parameter die teilweise leer bleiben können. Alle sind allerdings 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, dann 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.

amsmath definiert intern noch ein paar Kurzformen für die alltägliche Anwendung von `\genfrac`:

`\binom{`*oberer Teil*`}{`*unterer Teil*`}`	Binomialkoeffizient
`\dfrac{`*Zähler*`}{`*Nenner*`}`	Bruch im `\displaystyle` (vgl. Abschnitt 14.3)

Die Formeln werden im normalen Textsatz dann größer dargestellt. Dies bringt eine bessere Lesbarkeit mit sich, hat jedoch den Nachteil, dass die Zeilenabstände unschön beeinflusst werden. Das Makro bietet allerdings, (wenn es denn sein muss) eine Möglichkeit Doppelbrüche auch im Zeilenmodus lesbar darzustellen, vgl. Beispiel 12.

\textstyle $\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$ äußerster Bruch im \displaystyle $\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$

gesamter Bruch im \displaystyle $$\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$$

Beispiel 12: Größere Brüche mit \dfrac

14.9 Wurzeln

Wurzelausdrücke werden erzeugt mit:

```
\sqrt[Ordnung]{Formelcode}
```

Dabei definiert *Ordnung* die Wurzelexponente. Alles was innerhalb der Wurzel stehen soll wird in den *Formelcode* geschrieben. Hier können auch weitere Ausdrücke und Formeln stehen.

```
\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}
```

Eine vollständige Auflistung der standardmäßigen Symbol-Operatoren findet sich in Tabelle 14.2. Diese Symbol-Operatoren erscheinen in drei Größen, abhängig davon welcher Schriftstil aktiv ist (vgl. Abschnitt 14.3). Für die Position der Indizes gibt es ebenfalls eine Unterscheidung nach

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 13: Operatoren mit und ohne `\limits`

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 13). 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, wie 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.

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 simpler.

`\DeclareMathOperator*{`*Makro*`}{`*Operatorname*`}`

Beide Makros dürfen lediglich in der Präambel verwendet werden. Die dadurch erzeugten Markos können dann analog zu den vordefinierten Operatoren genutzt werden, siehe Beispiel 14. Eine Definition mit Sternchen ermöglicht zusätzlich die Verwendung von `\limits`.

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 14: Eigene Operatoren definieren

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*`}`

14.11 Spezielle Buchstaben und Zeichen

Griechische und hebräische Zeichen können innerhalb des Mathemodus oftmals entsprechend ihrem Wortlaut eingegeben werden, z. B. `\alpha` α. Der entsprechende Großbuchstabe wird dabei durch Großschreibung des ersten Zeichens des Makronamens 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. $\alpha \rightarrow \mathrm{A}$). 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 an, 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 stellt dabei das Onlinetool „Detexify“ [1][2] dar. Damit ist es möglich,

[2] http://detexify.kirelabs.org

Tabelle 14.4: Die in Standard-LaTeX verfügbaren Klammern. Klammernbefehle, welchen bereits ein `\big` vorangestellt wurde, existieren nicht in der kleinsten Größe.

`()`	()	`[]`	[]
`\{\}`	{}	`\langle \rangle`	⟨ ⟩
`/|\backslash`	/ | \	`\big\lmoustache \big\rmoustache`	⎰ ⎱
`\vert`	|	`\Vert`	‖
`\uparrow \downarrow \updownarrow`	↑ ↓ ↕	`\Uparrow \Downarrow \Updownarrow`	⇑ ⇓ ⇕
`\lceil \rceil`	⌈ ⌉	`\lfloor \rfloor`	⌊ ⌋

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 sowie ggf. notwendige Zusatzpakete angezeigt.

Für die Suche außerhalb des Internets ist „The Comprehensive LaTeX Symbol list“ [20] eine gute, wenn auch nicht ganz so komfortable Alternative. Die dazugehörige Datei `symbols-a4.pdf` ist Bestandteil der Basis jeder TeX-Distribution und über das Programm `texdoc` (vgl. Abschnitt 3.3) abrufbar:

Kommandozeile

```
texdoc symbols-a4
```

14.12 Klammern

Die Größe einzelner mathematischer Klammern kann manuell mit

```
\bigKlammerzeichen/-makro
\BigKlammerzeichen/-makro
\biggKlammerzeichen/-makro
\BiggKlammerzeichen/-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.

```
\leftKlammerzeichen/-makro
\rightKlammerzeichen/-makro
```

Für die korrekte Begrenzung der Höhe ist es jedoch zwingend erforderlich sowohl eine linke als

```
\begin{align*}
&\left(\begin{array}{c}9 \\ 2 \\ 4 \end{array}\right)&&
2\pi\hbar\left\{a+\frac{\frac{3}{2}\sum x_i^2}{m^*}\right\}&
y&=\left\{
    \begin{array}{r@{\quad:\quad}l}
        -1 & x<0 \\0 & x=0 \\1 & x>0
    \end{array}
\right.
\end{align*}
```

$$\left(\begin{array}{c}9 \\ 2 \\ 4 \end{array}\right) \qquad 2\pi\hbar\left\{a+\frac{\frac{3}{2}\sum x_i^2}{m^*}\right\} \qquad y=\left\{\begin{array}{rcl} -1 & : & x<0 \\ 0 & : & x=0 \\ 1 & : & x>0 \end{array}\right.$$

Beispiel 15: Automatische Größenanpassung von Klammern

```
% Definitionen am besten in der Präambel
\DeclarePairedDelimiter{\Bra}{\langle}{\rvert}
\DeclarePairedDelimiter{\Ket}{\lvert}{\rangle}
\[\Bra{A}^\dagger = \Ket{A}\]
```

$$\langle A|^\dagger = |A\rangle$$

Beispiel 16: Eigene Klammernmakros am Beispiel der Dirac-Notation (Quantenmechanik)

auch rechte Begrenzung zu setzen. Allerdings ist es über die Syntax `\left.` bzw. `\right.` möglich, ein Klammernsymbol zu unterdrücken, siehe Beispiel 15. Für Fallunterscheidungen stellt amsmath zusätzlich die Umgebung `cases` zur Verfügung. Ihre Verwendung ist in Beispiel 10 auf Seite 120 gezeigt.
Da die Verwendung von unterschiedlichen Klammern meistens auch ein Unterschiedliches MarkUp haben sollte gibt es mit dem mathtools Paket auch die Möglichkeit eigene Klammernpaare zu definieren.

`\DeclarePairedDelimiter{`*Makro*`}{`*Klammer links*`}{`*Klammer rechts*`}`

Ein komplettes Beispiel findet sich in Beispiel 16.

`\overbrace` und `\underbrace`

Neben den normalen vertikalen Klammern existieren auch horizontale geschweifte Klammern, um damit Formelsegmente beschriften zu können:

```
\[
\left(
\begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{array}
\right)
\]
```

$$\left(\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array} \right)$$

Beispiel 17: 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. Dies wird in Beispiel 17 gezeigt.

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 17. 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

amsmath bietet mit den Befehlen

`\overset{`*Ausdruck über*`}{`*Formeltext*`}`
`\underset{`*Ausdruck unter*`}{`*Formeltext*`}`

die Möglichkeit Objekte direkt über oder unter Formelfragmenten anzuordnen.

$$\overset{\text{Ausdruck über}}{\mathit{Formel}} \qquad \underset{\text{Ausdruck unter}}{\mathit{Formel}}$$

Dieser Befehl wird zum Beispiel genutzt, um Beschriftungen zu erzeugen oder chemische Formeln mit Oxidationszahlen zu versehen.

$$\overset{+\mathrm{VII}}{\mathrm{Mn\,O_4^-}} + 8\,\mathrm{H_3O^+} + 5\,\underset{\uparrow\text{freies Elektron}}{\mathrm{e^-}} \longrightarrow \overset{+\mathrm{II}}{\mathrm{Mn^{2+}}} + 12\,\mathrm{H_2O}$$

Zusätzlich existiert das Makro:

`\sideset{_{`*links unten*`}^{`*links oben*`}}{_{`*rechts unten*`}^{`*rechts oben*`}}{`*Operator*`}`

Hiermit können Indizes bei Operatoren (vgl. Abschnitt 14.10) auch linksseitig platziert werden.

$$\sideset{_{\text{links unten}}^{\text{links oben}}}{_{\text{rechts unten}}^{\text{rechts oben}}}{\prod}_{\text{unten}}^{\text{oben}}$$

14.15 Text im Mathemodus

Mit dem amsmath-Paket gibt es außerdem den Befehl:

`\text{`*Text*`}`

Hier wird der Text entsprechend des mathematischen Schriftstiles gewählt und als richtige Textbox formatiert. Das Makro `\mathrm` hingegen setzt 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 verwendet werden.

Soll Text zwischen die Zeilen einer mehrzeiligen Formel eingefügt werden, so geschieht dies mittels:

`\intertext{`*Text*`}`

Der Text wird dann so gesetzt und eingefügt, als würde die Matheumgebung beendet und anschließend neu begonnen werden nur mit der Fortsetzung der Ausrichtung.

Andere Möglichkeiten zum Setzen von Text im Mathemodus existieren zwar, sollten jedoch aufgrund der verminderten Anpassungsfähigkeiten im Gegensatz zu `\text` vermieden werden.

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

14.16 Theoreme

Der Autor kann auch eigene Strukturen bzw. Umgebungen mit eigenen Zählern 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. □

Tabelle 14.6: Wichtigste Optionen zu siunitx

Option=Wert	*Beispiel*	*Beschreibung*
`output-decimal-marker={,}`	1,23	Komma 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

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)
- Abstand eines halben Leerzeichens zwischen Wert und Einheit
- Einheitliche Struktur der Angaben (insbesondere sprachliche Besonderheiten: Punkt/Komma, Malpunkt/Produktkreuz, …)

Das Paket siunitx liefert bequeme Möglichkeiten für die Umsetzung. Hierbei wird das Markup effektiv genutzt. Für die Angabe innerhalb des Dokumentes ist es somit nicht von Bedeutung, welche Sprache genutzt wird und welche Formalitäten einzuhalten sind. Diese Aspekte sind lediglich für die Paketeinstellungen wichtig.

```
\usepackage[globale Optionen]{siunitx}
\sisetup{lokale Optionen}
```

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}
\unit[Optionen]{Einheiten}
```

Die Einheiten werden dafür als Textmakros eingegeben. Dies ermöglicht mithilfe der Einstellungen eine flexiblere Formatierung und verbessert die Lesbarkeit des Codes erheblich. Ein Beispiel dafür wäre: `\unit{\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

```
\begin{tabular}{@{}S[table-format=2.3]cl@{}}
    \toprule
    {Werte}&\option{c}-Spalte&\option{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 18: Verwendung der von siunitx definierten S-Spalte sowie des `\tablenum`-Makros

alle Möglichkeiten den hiesigen Rahmen. Die Einheiten lassen sich jedoch relativ einfach in der Paketanleitung [28] finden.

Für die Angabe einer Wert-Einheiten-Kombination steht ebenfalls ein gesondertes Makro zur Verfügung. Um also den richtigen Abstand gewährleisten zu können, verwendet man:

`\qty[`*Optionen*`]{`*Wert*`}{`*Einheit*`}`

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 [28] verwiesen.

14.17.2 Wertetabellen

Neben den Möglichkeiten für den Satz von Werten oder Einheiten beinhaltet das siunitx-Paket auch Optionen für Wertetabellen.

Werte können etwa innerhalb einer Tabelle am Dezimaltrennzeichen ausgerichtet werden. Der hierfür verwendete Spaltentyp ist die S-Spalte. Diese 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. Außerdem kann als Option das Format der enthaltenen Ziffernfolgen angegeben werden, vgl. Beispiel 18.

Neben dem neuen Spaltentyp ist außerdem noch dieses Makro vorhanden:

`\tablenum[`*Optionen*`]{`*Wert*`}`

Es ermöglicht eine Ausrichtung am Trennzeichen auch für Zellen, die mit `\multicolumn` zusammengefasst wurden. Ein Beispiel für die Verwendung ist in Beispiel 18 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, die auch Formatierungen und Ausrichtung anpassen können. Diese Varianten sind jedoch sehr speziell und können bei Bedarf der Paketanleitung [28] entnommen werden.

14.18 mhchem-Paket

```
\usepackage[version=4]{mhchem}
```

Das mhchem-Paket vereinfacht die nötige Befehlsstruktur um chemische Formeln zu setzen. Da es unterschiedliche Versionen gibt, muss darauf geachtet werden die richtige Version zu wählen. Wir nutzen Version 4. Außerdem ermöglicht dieses Paket Indizes auf der linken Seite hinzuzufügen, was sich beim Setzen von Isotopen als besonders praktisch erweist.
Der hauptsächliche Befehl ist dieser:

```
\ce{Summenformel}
```

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}+
}
```

H_2O Sb_{203} H^+ $CrO_4{}^{2-}$ $[AgCl_2]-$ Y^{99+}

Mengen Mengen werden direkt vor die restliche 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 [6]

```
\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 [6].

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] *DIN 5008:2020-03. Schreib- und Gestaltungsregeln für die Text- und Informationsverarbeitung*. März 2020. DOI: https://dx.doi.org/10.31030/3134524. URL: https://doi.org/10.31030/3134524.

[5] Florian Rödl. *Einführung in LaTeX 2ε*. 2012.

[6] Martin Hensel. *The mhchem Bundle*. 16. Jan. 2017. URL: http://www.ctan.org/pkg/mhchem.

[7] *JabRef. Free Reference Manager – Stay on top of your Literature*. URL: https://www.jabref.org/ (besucht am 02. 04. 2023).

[8] 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.

[9] Jianrui Jyu. *Tabularray. Typeset Tabulars and Arrays with LaTeX3*. 1. März 2023. URL: http://mirrors.ctan.org/macros/latex/contrib/tabularray/tabularray.pdf (besucht am 10. 03. 2023).

[10] Uwe Kern. *Extending LaTeX's color facilities: the xcolor package*. 31. Okt. 2021. URL: http://mirrors.ctan.org/macros/latex/contrib/xcolor/xcolor.pdf.

[11] Donald Ervin Knuth. *The TeXbook*. reprint. Bd. A. Computers and typesetting. Reading, Mass: Addison Wesley und Addison-Wesley, 2006.

[12] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script. Die Anleitung*. 12. Okt. 2022. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/doc/scrguide.pdf (besucht am 01. 01. 2023).

[13] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script. The Guide*. 12. Okt. 2022. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/doc/scrguien.pdf (besucht am 01. 01. 2023).

[14] 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.

[15] Helmut Kopka. *Einführung*. 1. Aufl., [unveränd. Nachdr.] Bd. 1. LaTeX. Bonn [u.a.]: Addison-Wesley, 1994.

[16] 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.

[17] Martin Schröder. *The ragged2e-package*. URL: http://truefunny.com/m/ctan/macros/latex/contrib/ms/ragged2e.pdf.

[18] Frank Mittelbach und Michel Goossens. *The LaTeX companion*. 2nd ed. Boston [etc.]: Addison-Wesley, 2004.

[19] Marion Neubauer. „Feinheiten bei wissenschaftlichen Publikationen – Mikrotypographie-Regeln, Teil I“. In: *Die TeXnische Komödie* 1996.4 (1996), S. 23–40.

[20] Scott Pakin. *The Comprehensive LaTeX Symbol List*. 19. Jan. 2017. URL: http://www.ctan.org/tex-archive/info/symbols/comprehensive/#symbols-a4.pdf.

[21] Philipp Lehman. *The biblatex Package: Programmable Bibliographies and Citations*. URL: http://ftp.fau.de/ctan/macros/latex/contrib/biblatex/doc/biblatex.pdf.

[22] 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).

[23] Robin Fairbairns. *footmisc: a portmanteau package for customising footnotes in LaTeX*. URL: http://ftp.fau.de/ctan/macros/latex/contrib/footmisc/footmisc.pdf.

[24] Robert Schlicht. *The microtype package. Subliminal refinements towards typographical perfection*. 13. März 2023. URL: http://mirrors.ctan.org/macros/latex/contrib/microtype/microtype.pdf.

[25] Till Tantau, Joseph Wright und Vedran Miletić. *The beamer class: User Guide for version 3.20*. URL: http://mirror.physik-pool.tu-berlin.de/tex-archive/macros/latex/contrib/beamer/doc/beameruserguide.pdf.

[26] TUG. *Installing TeX Live over the Internet*. URL: https://www.tug.org/texlive/acquire-netinstall.html.

[27] Herbert Voß. *Einführung in LaTeX 2ε: Unter Berücksichtigung von pdfLaTeX, XeLaTeX und LuaLaTeX*. 1. Aufl. Lehmanns Media, 2012.

[28] Joseph Wright. *siunitx: A comprehensive (SI) units package*. URL: http://ctan.space-pro.be/tex-archive/macros/latex/contrib/siunitx/siunitx.pdf.

Index

Index der Makros, Umgebungen und Optionen