KOMA-Script

ein wandelbares LaTeX 2$_\varepsilon$-Paket

Die Anleitung

KOMA-Script

Markus Kohm Jens-Uwe-Morawski

2012-01-01

Autoren des KOMA-Script-Pakets: Frank Neukam, Markus Kohm, Axel Kielhorn

Rechtliche Hinweise:

Die Autoren dieser Anleitung sind in ihrer Eigenschaft als Autoren dieser Anleitung nicht verantwortlich für die Funktion oder Fehler der in dieser Anleitung beschriebenen Software. Bei der Erstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden.

Die Autoren können jedoch für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen. Für Verbesserungsvorschläge und Hinweise auf Fehler sind die Autoren dankbar.

In dieser Anleitung werden Warennamen ohne der Gewährleistung der freien Verwendbarkeit und ohne besondere Kennzeichnung benutzt. Es ist jedoch davon auszugehen, dass viele der Warennamen gleichzeitig eingetragene Warenzeichnen oder als solche zu betrachten sind.

Den Freunden der Typografie!

Vorwort zu KOMA-Script 3

KOMA-Script 3 unterscheidet sich in vielerlei Hinsicht von KOMA-Script 2.5 bis 2.98 – jedenfalls solange man sich auf die dokumentierten Möglichkeiten beschränkt. Daher werden auch diejenigen, die bereits frühere Auflagen kennen, in dieser Anleitung sehr viel Neues finden. Aber nicht nur wegen der neuen Möglichkeiten unterscheidet sich die neue Auflage von früheren.

In den vergangenen Jahren haben mich einige Kommentare von Lesern erreicht. Obwohl die Mehrheit insgesamt doch recht zufrieden schien, gab es auch die eine oder andere kritische Stimme. Während einigen die Anleitung zu kurz war, war sie anderen wiederum zu ausführlich. Zwar ist es nicht möglich, solch gegensätzliche Kritiken komplett umzusetzen. Mit der Aufteilung in mehrere Teile hoffe ich aber, sie in Form eines Kompromisses doch aufgegriffen zu haben.

Mancher bemängelte, dass das Kapitel über die Briefklasse zu wenige Beispiele enthielte. Ein komplettes Beispiel war tatsächlich erst am Ende des Kapitels zu finden. Diese Kritik habe ich aufgegriffen. In dem komplett überarbeiteten Kapitel stehen nun vollständige Briefbeispiele im Vordergrund. Durch die Weiterentwicklung von eher minimalen zu immer ausführlicheren Beispielen kann der Leser Unterschiede zwischen Voreinstellungen und selbst gewählten Konfigurationsmöglichkeiten direkt sehen. Die vielen Abbildungen ermöglichen dem Anwender außerdem, direkt herauszugreifen, was er als Endergebnis wünscht, und dann die Einstellungen aus dem zugehörigen abgedruckten Quellcode zu übernehmen und anzupassen.

Einige beklagten, dass die Nutzung der Briefklasse unbedingt erforderte, auch die Anleitung zu den Hauptklassen heranzuziehen. Auch diese Kritik habe ich in der Buchfassung aufgegriffen. Jedes Kapitel behandelt dort das jeweilige Thema nun in sich abgeschlossen. Wird etwas von den in einem Kapitel behandelten Klassen oder Paketen bereitgestellt, wird für die Erklärung dieser Möglichkeit nicht mehr insgesamt auf andere Kapitel verwiesen. Lediglich für größere Beispiele, weiterführende Informationen, eine Abbildung oder eine Tabelle kann es einmal notwendig sein, einem Querverweis in ein anderes Kapitel zu folgen. In der vorliegenden freien Version der Anleitung habe ich darauf jedoch verzichtet, um die PDF-Datei kleiner zu halten. Durch die Möglichkeit, per Hyperlink an die referenzierte Stelle zu springen, sollte dies jedoch eine vertretbare Einschränkung sein.

Die Hauptkritik betraf die Unterteilung der Kapitel. Zwar war die Unterteilung so aufgebaut, dass die Dinge, die am Anfang eines Dokuments stehen, auch am Anfang des Kapitels zu finden sind. Allerdings betraf dies auch die Klassenoptionen, weil diese nun einmal im Dokument ganz zu Anfang angegeben werden. Ich muss mir die Kritik gefallen lassen, dass ich dabei nicht berücksichtigt hatte, dass die Optionen zwar im Kopf des Dokuments gesetzt werden, ein Autor viele der Optionen aber erst dann setzt, wenn er sich mit dem zugehörigen Thema befasst. Deshalb sind die Kapitel nun komplett thematisch geordnet. Das führte nebenbei auch

zu einer flacheren Hierarchie in der Gliederung.

Neben diesen Änderungen, die nur den Lesern auffallen werden, die bereits frühere Auflagen studiert haben, gibt es auch eine ganz augenfällige Änderung. Das Buch wurde wieder einmal erheblich dicker. Der gesteigerte Umfang ist zum einen in der erwähnten Umsetzung der Leserkritik begründet. Er ist aber auch den erheblich erweiterten Möglichkeiten von KOMA-Script 3 geschuldet.

Eine erfreuliche Kritik zu den früheren Auflagen war, dass es mir gelungen sei, eine eigentlich trockene Anleitung so mit Hintergrundwissen zu vermischen und in eine Sprache zu fassen, dass ein lesenswertes Buch entstanden sei. Ich wünsche dem geneigten Leser, dass mir dies erneut gelungen ist.

Leser dieser freien Bildschirm-Version müssen allerdings mit einigen Einschränkungen leben. So sind einige Informationen – hauptsächliche solche für fortgeschrittene Anwender oder die dazu geeignet sind, aus einem Anwender einen fortgeschrittenen Anwender zu machen – der Buchfassung vorbehalten. Das führt auch dazu, dass einige Links in dieser Anleitung lediglich zu einer Seite führen, auf der genau diese Tatsache erwähnt ist.

Zum Schluss möchte ich mich bei allen bedanken, die an der Entstehung dieses Buches beteiligt waren. Besonders erwähnt seien mein Co-Autor, Jens-Uwe Morawski, die Beta-Tester und die Testleser. Darüber hinaus gilt mein Dank DANTE e.V. und Lehmanns Media, namentlich Klaus Höppner und Christoph Kaeder, die es überhaupt erst möglich gemacht haben, dass niemand mehr diese Anleitung ausdrucken muss, sondern mit [KM08] eine um rund 200 Seiten umfangreichere Version dieser Anleitung in gebundener Form verfügbar ist.

Markus Kohm, Neckarhausen im September 2008

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

Dieses Kapitel enthält unter anderem wichtige Informationen über den Aufbau der Anleitung und die Geschichte von KOMA-Script, die Jahre vor der ersten Version beginnt. Darüber hinaus finden Sie Informationen für den Fall, dass Sie KOMA-Script noch nicht installiert haben, oder auf Fehler stoßen.

1.1. Vorwort

KOMA-Script ist ein sehr komplexes Paket (engl. *bundle*). Dies ist schon allein darin begründet, dass es nicht nur aus einer einzigen Klasse (engl. *class*) oder einem einzigen Paket (engl. *package*), sondern einer Vielzahl derer besteht. Zwar sind die Klassen als Gegenstücke zu den Standardklassen konzipiert (siehe Kapitel 3), das heißt jedoch insbesondere nicht, dass sie nur über die Befehle, Umgebungen und Einstellmöglichkeiten der Standardklassen verfügen oder deren Aussehen als Standardeinstellung übernehmen. Die Fähigkeiten von KOMA-Script reichen teilweise weit über die Fähigkeiten der Standardklassen hinaus. Manche davon sind auch als Ergänzung zu den Grundfähigkeiten des LaTeX-Kerns zu betrachten.

Allein aus dem Vorgenannten ergibt sich schon zwangsläufig, dass die Dokumentation zu KOMA-Script sehr umfangreich ausfällt. Hinzu kommt, dass KOMA-Script in der Regel nicht gelehrt wird. Das heißt, es gibt keinen Lehrer, der seine Schüler kennt und damit den Unterricht und das Unterrichtsmaterial entsprechend wählen und anpassen kann. Es wäre ein Leichtes, die Dokumentation für irgendeine Zielgruppe zu verfassen. Die Schwierigkeit, der sich die Autoren gegenüber sehen, besteht jedoch darin, dass eine Anleitung für alle möglichen Zielgruppen benötigt wird. Wir haben uns bemüht, eine Anleitung zu erstellen, die für den Informatiker gleichermaßen geeignet ist wie für die Sekretärin des Fischhändlers. Wir haben uns bemüht, obwohl es sich dabei eigentlich um ein unmögliches Unterfangen handelt. Ergebnis sind zahlreiche Kompromisse. Wir bitten jedoch, die Problematik bei eventuellen Beschwerden zu berücksichtigen und bei der Verbesserung unserer derzeitigen Lösung zu helfen.

Trotz des Umfangs der Anleitung bitten wir außerdem darum, im Falle von Problemen zunächst die Dokumentation zu konsultieren. Als erste Anlaufstelle sei auf den mehrteiligen Index am Ende des Dokuments hingewiesen. Zur Dokumentation gehören neben dieser Anleitung auch alle Text-Dokumente, die Bestandteil des Pakets sind. Sie sind in `manifest.txt` vollständig aufgeführt

1.2. Dokumentaufbau

Diese Anleitung ist in mehrere Teile untergliedert. Es gibt einen Teil für Anwender, einen für fortgeschrittene Anwender und Experten und einen Anhang mit weiterführenden Informatio-

nen und Beispielen für diejenigen, die es ganz genau wissen wollen.

Teil I richtet sich dabei an alle KOMA-Script-Anwender. Das bedeutet, dass hier auch einige Informationen für LaTeX-Neulinge zu finden sind. Insbesondere ist dieser Teil mit vielen Beispielen angereichert, die dem reinen Anwender zur Verdeutlichung der Erklärungen dienen sollen. Scheuen Sie sich nicht, diese Beispiele selbst auszuprobieren und durch Abwandlung herauszufinden, wie KOMA-Script funktioniert. Trotz allem ist diese Anleitung jedoch keine Einführung in LaTeX. LaTeX-Neulingen seien daher Dokumente wie [SKPH99] nahe gelegt. Wiedereinsteiger aus der Zeit von LaTeX 2.09 sei zumindest [Tea05b] empfohlen. Auch das Studium des einen oder anderen Buches zu LaTeX wird empfohlen. Literaturempfehlungen finden sich beispielsweise in [Wik]. Der Umfang von [Wik] ist ebenfalls erheblich. Dennoch wird darum gebeten, das Dokument nicht nur irgendwo vorliegen zu haben, sondern es mindestens einmal zu lesen und bei Problemen zu konsultieren.

Teil II richtet sich an fortgeschrittene KOMA-Script-Anwender. Das sind all diejenigen, die sich bereits mit LaTeX auskennen oder schon einige Zeit mit KOMA-Script gearbeitet haben und jetzt etwas besser verstehen wollen, wie KOMA-Script funktioniert, wie es mit anderen Paketen interagiert und wie man speziellere Aufgaben mit KOMA-Script lösen kann. Dazu werden die Klassenbeschreibungen aus Teil I in einigen Aspekten nochmals aufgegriffen und näher erläutert. Dazu kommt die Dokumentation von Anweisungen, die speziell für fortgeschrittene Anwender und Experten vorgesehen sind. Ergänzt wird dies durch die Dokumentation von Paketen, die für den Anwender normalerweise insofern verborgen sind, als sie unter der Oberfläche der Klassen und Anwenderpakete ihre Arbeit verrichten. Diese Pakete sind ausdrücklich auch für die Verwendung durch andere Klassen- und Paketautoren vorgesehen.

Der Anhang, der nur in der Buchfassung zu finden ist, richtet sich an diejenigen, denen all diese Informationen nicht genügen. Es gibt dort zum einen Hintergrundwissen zu Fragen der Typografie, mit denen dem fortgeschrittenen Anwender eine Grundlage für fundierte eigene Entscheidungen vermittelt werden soll. Darüber hinaus sind dort Beispiele für angehende Paketautoren zu finden. Diese Beispiele sind weniger dazu gedacht, einfach übernommen zu werden. Vielmehr sollen sie Wissen um Planung und Durchführung von LaTeX-Projekten sowie einiger grundlegender LaTeX-Anweisungen für Paketautoren vermitteln.

Die Kapitel-Einteilung der Anleitung soll ebenfalls dabei helfen, nur die Teile lesen zu müssen, die tatsächlich von Interesse sind. Um dies zu erreichen, sind die Informationen zu den einzelnen Klassen und Paketen nicht über das gesamte Dokument verteilt, sondern jeweils in einem Kapitel konzentriert. Querverweise in ein anderes Kapitel sind damit in der Regel auch Verweise auf einen anderen Teil des Gesamtpakets. Da die drei Hauptklassen in weiten Teilen übereinstimmen, sind sie in einem gemeinsamen Kapitel zusammengefasst. Die Unterschiede werden deutlich hervorgehoben, soweit sinnvoll auch durch eine entsprechende Randnotiz. Dies

scrartcl

geschieht beispielsweise wie hier, wenn etwas nur die Klasse scrartcl betrifft. Nachteil dieses Vorgehens ist, dass diejenigen, die KOMA-Script insgesamt kennenlernen wollen, in einigen Kapiteln auf bereits Bekanntes stoßen werden. Vielleicht nutzen Sie die Gelegenheit, um Ihr Wissen zu vertiefen.

Unterschiedliche Schriftarten werden auch zur Hervorhebung unterschiedlicher Dinge verwendet. So werden die Namen von Paketen und Klassen anders angezeigt als `Dateinamen`. `Optionen`, `\Anweisungen`, `Umgebungen`, `Variablen` und `Pseudolängen` werden einheitlich hervorgehoben. Der *`Platzhalter`* für einen Parameter wird jedoch anders dargestellt als ein konkreter `Wert` eines Parameters. So zeigt etwa `\begin{`*`Umgebungsname`*`}`, wie eine Umgebung ganz allgemein eingeleitet wird, wohingegen `\begin{document}` angibt, wie die konkrete Umgebung `document` beginnt. Dabei ist dann `document` ein konkreter Wert für den Parameter *`Umgebungsname`* der Anweisung `\begin`.

1.3. Die Geschichte von KOMA-Script

Anfang der 90er Jahre wurde Frank Neukam damit beauftragt, ein Vorlesungsskript zu setzen. Damals war noch LaTeX 2.09 aktuell und es gab keine Unterscheidung nach Klassen und Paketen, sondern alles waren Stile (engl. *styles*). Die Standarddokumentstile erschienen ihm für ein Vorlesungsskript nicht optimal und boten auch nicht alle Befehle und Umgebungen, die er benötigte.

Zur selben Zeit beschäftigte sich Frank auch mit Fragen der Typografie, insbesondere mit [Tsc87]. Damit stand für ihn fest, nicht nur irgendeinen Dokumentstil für Skripten zu erstellen, sondern allgemein eine Stilfamilie, die den Regeln der europäischen Typografie folgt. Script war geboren.

Der KOMA-Script-Autor traf auf Script ungefähr zum Jahreswechsel 1992/1993. Im Gegensatz zu Frank Neukam hatte er häufig mit Dokumenten im A5-Format zu tun. Zu jenem Zeitpunkt wurde A5 weder von den Standardstilen noch von Script unterstützt. Daher dauerte es nicht lange, bis er erste Veränderungen an Script vornahm. Diese fanden sich auch in Script-2 wieder, das im Dezember 1993 von Frank veröffentlicht wurde.

Mitte 1994 erschien dann LaTeX 2_ε. Die damit einhergehenden Änderungen waren tiefgreifend. Daher blieb dem Anwender von Script-2 nur die Entscheidung, sich entweder auf den Kompatibilitätsmodus von LaTeX zu beschränken, oder auf Script zu verzichten. Wie viele andere wollte ich beides nicht. Also machte der KOMA-Script-Autor sich daran, einen Script-Nachfolger für LaTeX 2_ε zu entwickeln, der am 7. Juli 1994 unter dem Namen KOMA-Script erschienen ist. Ich will hier nicht näher auf die Wirren eingehen, die es um die offizielle Nachfolge von Script gab und warum dieser neue Name gewählt wurde. Tatsache ist, dass auch aus Franks Sicht KOMA-Script der Nachfolger von Script-2 ist. Zu erwähnen ist noch, dass KOMA-Script ursprünglich ohne Briefklasse erschienen war. Diese wurde im Dezember 1994 von Axel Kielhorn beigesteuert. Noch etwas später erstellte Axel Sommerfeldt den ersten richtigen scrguide zu KOMA-Script.

Seither ist einige Zeit vergangen. LaTeX hat sich kaum verändert, die LaTeX-Landschaft erheblich. KOMA-Script wurde weiterentwickelt. Es findet nicht mehr allein im deutschsprachigen Raum Anwender, sondern in ganz Europa, Nordamerika, Australien und Asien. Diese Anwender suchen bei KOMA-Script nicht allein nach einem typografisch ansprechenden Er-

gebnis. Zu beobachten ist vielmehr, dass bei KOMA-Script ein neuer Schwerpunkt entstanden ist: Flexibilisierung durch Variabilisierung. Unter diesem Schlagwort verstehe ich die Möglichkeit, in das Erscheinungsbild an vielen Stellen eingreifen zu können. Dies führte zu vielen neuen Makros, die mehr schlecht als recht in die existierende Dokumentation integriert wurden. Irgendwann wurde es damit auch Zeit für eine komplett überarbeitete Anleitung.

1.4. Danksagung

Eine Danksagung in der Einleitung? Gehört sie nicht vielmehr an den Schluss? Richtig! Eigentlich gehört die an das Ende. Mein Dank gilt hier jedoch nicht primär denjenigen, die diese Anleitung möglich gemacht haben. Für den Dank an die Guide-Autoren mache ich den Leser zuständig! Mein persönlicher Dank gilt Frank Neukam, ohne dessen Script-Familie es vermutlich KOMA-Script nie gegeben hätte. Mein Dank gilt denjenigen, die an der Entstehung von KOMA-Script und den Anleitungen mitgewirkt haben. Stellvertretend für andere möchte ich hier Jens-Uwe Morawski, Torsten Krüger und Gernot Hassenpflug nennen. Jens' unermüdlichem Einsatz ist neben vielem Anderen die englische Übersetzung der Anleitung zu verdanken, die Gernot in wesentlichen Teilen mitgestaltet hat. Torsten ist der beste Betatester, den ich je hatte. Er hat damit insbesondere an der Verwendbarkeit von scrlttr2 und scrpage2 einen erheblichen Anteil. Mein Dank gilt auch allen, die mich immer wieder aufgemuntert haben, weiter zu machen und dieses oder jenes noch besser, weniger fehlerhaft oder schlicht zusätzlich zu implementieren.

Ganz besonderen Dank bin ich den Gründern von DANTE, Deutschsprachige Anwendervereinigung TEX e.V., schuldig, durch die letztlich die Verbreitung von TEX und LATEX und allen Paketen einschließlich KOMA-Script an einer zentralen Stelle überhaupt ermöglicht wird. In gleicher Weise bedanke ich mich bei den aktiven Helfern auf der Mailingliste TEX-D-L (siehe [Wik]) und in der Usenet-Gruppe de.comp.text.tex, die mir so manche Antwort auf Fragen zu KOMA-Script abnehmen.

1.5. Rechtliches

1.6. Installation

Die drei wichtigsten TEX-Distributionen, MacTEX, MiKTEX und TEX Live, stellen KOMA-Script über Ihre jeweiligen Paketverwaltungen bereit. Es wird empfohlen, die Installation und Aktualisierung von KOMA-Script darüber vorzunehmen. Die manuelle Installation von KOMA-Script ohne Verwendung der jeweiligen Paketverwaltung wird in der Datei `INSTALLD.txt`, die Bestandteil jeder KOMA-Script-Verteilung ist, beschrieben. Beachten Sie dazu auch die jeweilige Dokumentation zur installierten TEX-Distribution.

1.7. Fehlermeldungen, Fragen, Probleme

Sollten Sie der Meinung sein, dass Sie einen Fehler in der Anleitung, einer der KOMA-Script-Klassen, einem der KOMA-Script-Pakete oder einem anderen Bestandteil von KOMA-Script gefunden haben, so sollten Sie wie folgt vorgehen. Prüfen Sie zunächst, ob inzwischen eine neue Version von KOMA-Script erschienen ist. Installieren Sie diese neue Version und kontrollieren Sie, ob der Fehler oder das Problem auch dann noch vorhanden ist.

Wenn es sich nicht um einen Fehler in der Dokumentation handelt und der Fehler oder das Problem nach einem Update noch immer auftritt, erstellen Sie bitte wie in [Wik] angegeben ein minimales Beispiel. Gehen Sie dazu wie unten beschrieben vor. Oft lässt sich ein Problem durch ein minimales Beispiel so weit eingrenzen, dass bereits vom Anwender selbst festgestellt werden kann, ob es sich um einen Anwendungsfehler handelt oder nicht. Auch ist so sehr häufig zu erkennen, welche Pakete oder Klassen konkret das Problem verursachen und ob es sich überhaupt um ein KOMA-Script-Problem handelt. Dies können Sie gegebenenfalls zusätzlich überprüfen, indem Sie statt einer KOMA-Script-Klasse einen Test mit der entsprechenden Standardklasse vornehmen. Danach ist dann auch klar, ob der Fehlerbericht an den Autor von KOMA-Script oder an den Autor eines anderen Pakets zu richten ist. Sie sollten spätestens jetzt noch einmal gründlich die Anleitungen der entsprechenden Paket, Klassen und KOMA-Script-Bestandteile studieren sowie [Wik] konsultieren. Möglicherweise existiert ja bereits eine Lösung für Ihr Problem, so dass sich eine Fehlermeldung erübrigt.

Wenn Sie denken, dass Sie einen noch unbekannten Fehler gefunden haben, oder es aus anderem Grund für sinnvoll oder notwendig erachten, den KOMA-Script-Autor zu kontaktieren, so sollten Sie dabei folgende Angaben keinesfalls vergessen:

- Tritt das Problem auch auf, wenn statt einer KOMA-Script-Klasse eine Standardklasse verwendet wird? In dem Fall liegt der Fehler höchst wahrscheinlich nicht bei KOMA-Script. Es ist dann sinnvoller, die Frage in einem öffentlichen Forum, einer Mailingliste oder im Usenet zu stellen.

- Welche KOMA-Script-Version verwenden Sie? Entsprechende Informationen finden Sie in der `log`-Datei des LATEX-Laufs jedes Dokuments, das eine KOMA-Script-Klasse verwendet.

- Welches Betriebssystem und welche TEX-Distribution wird verwendet? Diese Angaben erscheinen bei einem bestriebssystemunabhängigen Paket wie KOMA-Script oder LATEX eher überflüssig. Es zeigt sich aber immer wieder, dass sie durchaus eine Rolle spielen können.

- Was genau ist das Problem oder der Fehler? Beschreiben Sie das Problem oder den Fehler lieber zu ausführlich als zu knapp. Oftmals ist es sinnvoll auch die Hintergründe zu erläutern.

- Wie sieht ein vollständiges Minimalbeispiel aus? Ein solches vollständiges Minimalbeispiel kann jeder leicht selbst erstellen, indem Schritt für Schritt Inhalte und Pakete aus dem Problemdokument auskommentiert werden. Am Ende bleibt ein Dokument, das nur die Pakete lädt und nur die Teile enthält, die für das Problem notwendig sind. Außerdem sollten alle geladenen Abbildungen durch `\rule`-Anweisungen entsprechender Größe ersetzt werden. Vor dem Verschicken entfernt man nun die auskommentierten Teile, fügt als erste Zeile die Anweisung `\listfiles` ein und führt einen weiteren LATEX-Lauf durch. Man erhält dann am Ende der `log`-Datei eine Übersicht über die verwendeten Pakete. Das vollständige Minimalbeispiel und die `log`-Datei fügen Sie ihrer Beschreibung hinzu.

Schicken Sie keine Pakete, PDF- oder PS- oder DVI-Dateien mit. Falls die gesamte Problem- oder Fehlerbeschreibung einschließlich Minimalbeispiel und `log`-Datei größer als ein paar Dutzend KByte ist, haben Sie mit größter Wahrscheinlichkeit etwas falsch gemacht. Anderenfalls schicken Sie Ihre Mitteilung an komascript@gmx.info.

Häufig werden Sie eine Frage zu KOMA-Script oder im Zusammenhang mit KOMA-Script lieber öffentlich, beispielsweise in `de.comp.text.tex` oder dem Forum auf [KDP], stellen wollen. Auch in diesem Fall sollten Sie obige Punkte beachten, in der Regel jedoch auf die `log`-Datei verzichten. Fügen Sie stattdessen nur die Liste der Pakete und Paketversionen aus der `log`-Datei an. Im Falle einer Fehlermeldung zitieren Sie diese ebenfalls aus der `log`-Datei.

1.8. Weitere Informationen

Sobald Sie im Umgang mit KOMA-Script geübt sind, werden Sie sich möglicherweise Beispiele zu schwierigeren Aufgaben wünschen. Solche Beispiele gehen über die Vermittlung von Grundwissen hinaus und sind daher nicht Bestandteil dieser Anleitung. Auf den Internetseiten des KOMA-Script Documentation Projects [KDP] finden Sie jedoch weiterführende Beispiele. Diese sind für fortgeschrittene LATEX-Anwender konzipiert. Für Anfänger sind sie wenig oder nicht geeignet.

Teil I.

KOMA-Script für Autoren

In diesem Teil sind die Informationen für die Autoren von Artikeln, Berichten, Büchern und Briefen zu finden. Dabei wird davon ausgegangen, dass der normale Anwender sich weniger dafür interessiert, wie in KOMA-Script die Dinge implementiert sind und wo die Schwierigkeiten dabei liegen. Auch ist es für den normalen Anwender wenig interessant, welche obsoleten Optionen und Anweisungen noch enthalten sind. Er will wissen, wie er aktuell etwas erreichen kann. Eventuell ist er noch an typografischen Hintergrundinformationen interessiert.

Die wenigen Passagen, die weiterführende Informationen und Begründungen enthalten und deshalb für ungeduldige Leser weniger von Interesse sind, wurden in diesem Teil in serifenloser Schrift gesetzt und können bei Bedarf übersprungen werden. Wer hingegen noch mehr Informationen zu Hintergründen der Implementierung, Nebenwirkungen bei Verwendung anderer Pakete und zu obsoleten Optionen oder Anweisungen sucht, sei auf Teil II ab Seite 251 verwiesen. Darüber hinaus beschäftigt sich jener Teil von KOMA-Script auch mit all den Möglichkeiten, die speziell für Autoren von Paketen und Klassen geschaffen wurden.

Satzspiegelberechnung mit typearea.sty

Viele LaTeX-Klassen, darunter auch die Standardklassen, bieten dem Anwender eine weitgehend feste Aufteilung von Rändern und Textbereich. Bei den Standardklassen ist die konkrete Aufteilung in engen Grenzen von der gewählten Schriftgröße abhängig. Darüber hinaus gibt es Pakete wie geometry (siehe [Ume00]), die dem Anwender die volle Kontrolle, aber auch die Verantwortung für die Einstellungen des Textbereichs und der Ränder überlassen.

KOMA-Script geht mit dem Paket typearea einen etwas anderen Weg. Hier werden basierend auf einer in der Typografie etablierten Konstruktion Einstellmöglichkeiten und Automatismen geboten, die es dem Anwender erleichtern, eine gute Wahl zu treffen.

Es wird darauf hingewiesen, dass sich typearea des Pakets scrbase bedient. Letzteres Paket ist im Expertenteil dieser Anleitung in Kapitel 10 ab Seite 251 erklärt. Die Mehrzahl der dort dokumentierten Anweisungen richten sich jedoch nicht an Anwender, sondern an Klassen- und Paketautoren.

2.1. Grundlagen der Satzspiegelkonstruktion

Betrachtet man eine einzelne Seite eines Buches oder eines anderen Druckwerkes, so besteht diese auf den ersten Blick aus den Rändern, einem Kopfbereich, einem Textkörper und einem Fußbereich. Genauer betrachtet kommt noch ein Abstand zwischen Kopfbereich und Textkörper sowie zwischen Textkörper und Fußbereich hinzu. Der Textkörper heißt in der Fachsprache der Typografen und Setzer *Satzspiegel*. Die Aufteilung dieser Bereiche, sowie ihre Anordnung zueinander und auf dem Papier nennen wir *Satzspiegeldefinition* oder *Satzspiegelkonstruktion*.

In der Literatur werden verschiedene Algorithmen und heuristische Verfahren zur Konstruktion eines guten Satzspiegels vorgeschlagen und diskutiert[Koh02]. Häufig findet man dabei ein Verfahren, das mit verschiedenen Diagonalen und Schnittpunkten arbeitet. Das gewünschte Ergebnis dabei ist, dass das Seitenverhältnis des Satzspiegels dem Seitenverhältnis *der Seite* entspricht. Bei einem einseitigen Dokument sollen außerdem der linke und der rechte Rand gleich breit sein, während der obere zum unteren Rand im Verhältnis 1:2 stehen sollte. Bei einem doppelseitigen Dokument, beispielsweise einem Buch, ist hingegen zu beachten, dass der gesamte innere Rand genauso groß sein sollte wie jeder der beiden äußeren Ränder. Eine einzelne Seite steuert dabei jeweils nur die Hälfte des inneren Randes bei.

Im vorherigen Absatz wurde *die Seite* erwähnt und hervorgehoben. Irrtümlich wird oftmals angenommen, das Format der Seite wäre mit dem Format des Papiers gleichzusetzen. Betrachtet man jedoch ein gebundenes Druckerzeugnis, so ist zu erkennen, dass ein Teil des Papiers in der Bindung verschwindet und nicht mehr als Seite zu sehen ist. Für den Satzspiegel ist jedoch nicht entscheidend, welches Format das Papier hat, sondern, was der Leser für einen Eindruck vom Format der Seite bekommt. Damit ist klar, dass bei der Berechnung des Satzspiegels der Teil, der

durch die Bindung versteckt wird, aus dem Papierformat herausgerechnet und dann zum inneren Rand hinzugefügt werden muss. Wir nennen diesen Teil *Bindekorrektur*. Die Bindekorrektur ist also rechnerischer Bestandteil des *Bundstegs*, nicht jedoch des sichtbaren inneren Randes.

Die Bindekorrektur ist vom jeweiligen Produktionsvorgang abhängig und kann nicht allgemein festgelegt werden. Es handelt sich dabei also um einen Parameter, der für jeden Produktionsvorgang neu festzulegen ist. Im professionellen Bereich spielt dieser Wert nur eine geringe Rolle, da ohnehin auf größere Papierbögen gedruckt und entsprechend geschnitten wird. Beim Schneiden wird dann wiederum sichergestellt, dass obige Verhältnisse für die sichtbare Doppelseite eingehalten sind.

Wir wissen nun also, wie die einzelnen Teile zueinander stehen. Wir wissen aber noch nicht, wie breit und hoch der Satzspiegel ist. Kennen wir eines dieser beiden Maße, so ergeben sich zusammen mit dem Papierformat und dem Seitenformat oder der Bindekorrektur alle anderen Maße durch Lösung mehrerer mathematischer Gleichungen.

$$
\begin{aligned}
\mathit{Satzspiegelhöhe} : \mathit{Satzspiegelbreite} &= \mathit{Seitenhöhe} : \mathit{Seitenbreite} \\
\mathit{oberer\ Rand} : \mathit{unterer\ Rand} &= 1 : 2 \\
\mathit{linker\ Rand} : \mathit{rechter\ Rand} &= 1 : 1 \\
\mathit{innerer\ Randanteil} : \mathit{äußerer\ Rand} &= 1 : 2 \\
\mathit{Seitenbreite} &= \mathit{Papierbreite} - \mathit{Bindekorrektur} \\
\mathit{oberer\ Rand} + \mathit{unterer\ Rand} &= \mathit{Seitenhöhe} - \mathit{Satzspiegelhöhe} \\
\mathit{linker\ Rand} + \mathit{rechter\ Rand} &= \mathit{Seitenbreite} - \mathit{Satzspiegelbreite} \\
\mathit{innerer\ Randanteil} + \mathit{äußerer\ Rand} &= \mathit{Seitenbreite} - \mathit{Satzspiegelbreite} \\
\mathit{innerer\ Randanteil} + \mathit{Bindekorrektur} &= \mathit{Bundsteg}
\end{aligned}
$$

Dabei gibt es *linker Rand* und *rechter Rand* nur im einseitigen Druck. Entsprechend gibt es *innerer Randanteil* und *äußerer Rand* nur im doppelseitigen Druck.

In den Gleichungen wird mit *innerer Randanteil* gearbeitet, weil der komplette innere Rand ein Element der vollständigen Doppelseite ist. Zu einer Seite gehört also nur die Hälfte des inneren Randes: *innerer Randanteil*.

Die Frage nach der Breite des Satzspiegels wird in der Literatur ebenfalls diskutiert. Die optimale Satzspiegelbreite ist von verschiedenen Faktoren abhängig:

- Größe, Laufweite und Art der verwendeten Schrift
- Verwendeter Durchschuss
- Länge der Worte
- Verfügbarer Platz

Der Einfluss der Schrift wird deutlich, wenn man sich bewusst macht, wozu Serifen dienen. Serifen sind kleine Striche an den Linienenden der Buchstaben. Buchstaben, die mit vertikalen Linien auf

die Grundlinie der Textzeile treffen, lösen diese eher auf, als dass sie das Auge auf der Linie halten. Genau bei diesen Buchstaben liegen die Serifen horizontal auf der Grundlinie und verstärken damit die Zeilenwirkung der Schrift. Das Auge kann der Textzeile nicht nur beim Lesen der Worte, sondern insbesondere auch beim schnellen Zurückspringen an den Anfang der nächsten Zeile besser folgen. Damit darf die Zeile bei einer Schrift mit Serifen genau genommen länger sein als bei einer Schrift ohne Serifen.

Unter dem Durchschuss versteht man den Abstand zwischen Textzeilen. Bei LaTeX ist ein Durchschuss von etwa 20 % der Schriftgröße voreingestellt. Mit Befehlen wie `\linespread` oder besser mit Hilfe von Paketen wie setspace (siehe [Tob00]) kann der Durchschuss verändert werden. Ein großer Durchschuss erleichtert dem Auge die Verfolgung einer Zeile. Bei sehr großem Durchschuss wird das Lesen aber dadurch gestört, dass das Auge zwischen den Zeilen weite Wege zurücklegen muss. Daneben wird sich der Leser des entstehenden Streifeneffekts sehr deutlich und unangenehm bewusst. Der Graueindruck der Seite ist in diesem Fall gestört. Dennoch dürfen bei großem Durchschuss die Zeilen länger sein.

Auf der Suche nach konkreten Werten für gute Zeilenlängen findet man in der Literatur je nach Autor unterschiedliche Angaben. Teilweise ist dies auch in der Muttersprache des Autors begründet. Das Auge springt nämlich üblicherweise von Wort zu Wort, wobei kurze Worte diese Aufgabe erleichtern. Über alle Sprachen und Schriftarten hinweg kann man sagen, dass eine Zeilenlänge von 60 bis 70 Zeichen, einschließlich Leer- und Satzzeichen, einen brauchbaren Kompromiss darstellen. Ein gut gewählter Durchschuss wird dabei vorausgesetzt. Bei den Voreinstellungen von LaTeX braucht man sich über Letzteres normalerweise keine Sorgen zu machen. Größere Zeilenlängen darf man nur Gewohnheitslesern zumuten, die täglich viele Stunden lesend zubringen. Aber auch dann sind Zeilenlängen jenseits von 80 Zeichen unzumutbar. In jedem Fall ist dann der Durchschuss anzupassen. 5 % bis 10 % zusätzlich sind dabei als Faustregel empfehlenswert. Bei Schriften wie Palatino, die bereits bei einer normalen Zeilenlänge nach 5 % mehr Durchschuss verlangt, können es auch mehr sein.

Bevor wir uns an die konkrete Konstruktion machen, fehlen jetzt nur noch Kleinigkeiten, die man wissen sollte. LaTeX beginnt die erste Zeile des Textbereichs einer Seite nicht am oberen Rand des Textbereichs, sondern setzt die Grundlinie der Zeile mit einem definierten Mindestabstand zum oberen Rand des Textbereichs. Des Weiteren verfügt LaTeX über die beiden Befehle `\raggedbottom` und `\flushbottom`. Der erste dieser Befehle legt fest, dass die letzte Zeile einer jeden Seite dort liegen soll, wo sie eben zu liegen kommt. Das kann dazu führen, dass sich die Position der letzten Zeile von Seite zu Seite vertikal um nahezu eine Zeile verändern kann – bei Zusammentreffen des Seitenendes mit Überschriften, Abbildungen, Tabellen oder Ähnlichem auch mehr. Im doppelseitigen Druck ist das in der Regel unerwünscht. Mit dem zweiten Befehl, `\flushbottom`, wird hingegen festgelegt, dass die letzte Zeile immer am unteren Rand des Textbereichs zu liegen kommt. Um diesen vertikalen Ausgleich zu erreichen, muss LaTeX gegebenenfalls dehnbare vertikale Abstände über das erlaubte Maß hinaus strecken. Ein solcher Abstand ist beispielsweise der Absatzabstand. Dies gilt in der Regel auch, wenn man gar keinen Absatzabstand verwendet. Um nicht bereits auf normalen Seiten, auf denen der Absatzabstand das einzige dehnbare vertikale Maß darstellt,

eine Dehnung zu erzwingen, sollte die Höhe des Textbereichs ein Vielfaches der Textzeilenhöhe zuzüglich des Abstandes der ersten Zeile vom oberen Rand des Textbereichs sein.

Damit sind alle Grundlagen der Satzspiegelberechnung, die bei KOMA-Script eine Rolle spielen, erklärt. Wir können also mit den konkreten Konstruktionen beginnen.

2.2. Satzspiegelkonstruktion durch Teilung

Der einfachste Weg, um zu erreichen, dass der Textbereich dasselbe Verhältnis aufweist wie die Seite, ist folgender:

- Zunächst zieht man an der Innenseite des Papiers den Teil *BCOR*, der für die Bindekorrektur benötigt wird, ab und teilt die restliche Seite vertikal in eine Anzahl *DIV* gleich hoher Streifen.
- Dann teilt man die Seite horizontal in die gleiche Anzahl *DIV* gleich breiter Streifen.
- Nun verwendet man den obersten horizontalen Streifen als oberen und die beiden untersten horizontalen Streifen als unteren Rand. Im doppelseitigen Druck verwendet man außerdem den innersten vertikalen Streifen als inneren und die beiden äußersten vertikalen Streifen als äußeren Rand.
- Zum inneren Rand gibt man dann noch *BCOR* hinzu.

Was nun innerhalb der Seite noch übrig bleibt, ist der Textbereich. Die Breite bzw. Höhe der Ränder und des Textbereichs resultiert damit automatisch aus der Anzahl *DIV* der Streifen. Da für die Ränder insgesamt jeweils drei Streifen benötigt werden, muss *DIV* zwingend größer als drei sein. Damit der Satzspiegel horizontal und vertikal jeweils mindestens doppelt so viel Platz wie die Ränder einnimmt, sollte *DIV* sogar mindestens 9 betragen. Mit diesem Wert ist die Konstruktion auch als *klassische Neunerteilung* bekannt (siehe Abbildung 2.1).

Bei KOMA-Script ist diese Art der Konstruktion im Paket typearea realisiert, wobei der untere Rand weniger als eine Textzeile kleiner ausfallen kann, um die im vorherigen Abschnitt erwähnte Nebenbedingung für die Satzspiegelhöhe einzuhalten und damit die dort erwähnte Problematik in Bezug auf `\flushbottom` zu mindern. Dabei sind für A4-Papier je nach Schriftgröße unterschiedliche Werte für *DIV* voreingestellt, die Tabelle 2.2, Seite 35 zu entnehmen sind. Bei Verzicht auf Bindekorrektur, wenn also $BCOR = 0\,\mathrm{pt}$ gilt, ergeben sich in etwa die Satzspiegelmaße aus Tabelle 2.1, Seite 34.

Neben den voreingestellten Werten kann man *BCOR* und *DIV* direkt beim Laden des Pakets als Option angeben (siehe Abschnitt 2.6 ab Seite 32). Zusätzlich existiert ein Befehl, mit dem man einen Satzspiegel explizit berechnen kann und dem man die beiden Werte als Parameter übergibt (Seite 38).

Das typearea-Paket bietet außerdem die Möglichkeit, den optimalen *DIV*-Wert automatisch zu bestimmen. Dieser ist von der Schrift und dem Durchschuss abhängig, der zum Zeitpunkt der Satzspiegelberechnung eingestellt ist (Seite 35).

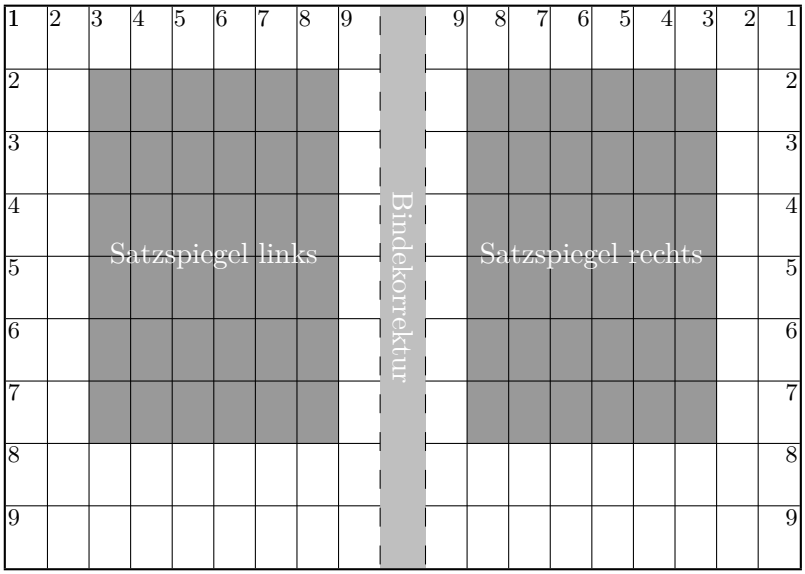


Abbildung 2.1.: Doppelseite mit der Rasterkonstruktion für die klassische Neunerteilung nach Abzug einer Bindekorrektur

2.3. Satzspiegelkonstruktion durch Kreisschlagen

Neben der zuvor beschriebenen Satzspiegelkonstruktion gibt es in der Literatur noch eine eher klassische oder sogar mittelalterliche Methode. Bei diesem Verfahren will man die gleichen Werte nicht nur in Form des Seitenverhältnisses wiederfinden; man geht außerdem davon aus, dass das Optimum dann erreicht wird, wenn die Höhe des Textbereichs der Breite der Seite entspricht. Das genaue Verfahren ist beispielsweise in [Tsc87] nachzulesen.

Als Nachteil dieses spätmittelalterlichen Buchseitenkanons ergibt sich, dass die Breite des Textbereichs nicht mehr von der Schriftart abhängt. Es wird also nicht mehr der zur Schrift passende Textbereich gewählt, stattdessen muss der Autor oder Setzer unbedingt die zum Textbereich passende Schrift wählen.

Im typearea-Paket wird diese Konstruktion dahingehend abgewandelt, dass durch Auswahl eines ausgezeichneten – normalerweise unsinnigen – *DIV*-Wertes oder eine spezielle Paket-Option derjenige *DIV*-Wert ermittelt wird, bei dem der resultierende Satzspiegel dem spätmittelalterlichen Buchseitenkanon am nächsten kommt. Es wird also wiederum auf die Satzspiegelkonstruktion durch Teilung zurückgegriffen.

2.4. Frühe oder späte Optionenwahl

In diesem Abschnitt wird eine Besonderheit von KOMA-Script vorgestellt, die neben typearea auch andere KOMA-Script-Pakete und -Klassen betrifft. Damit die Anwender alle Informationen zu einem Paket oder einer Klasse im jeweiligen Kapitel finden, ist dieser Abschnitt nahezu gleichlautend in mehreren Kapiteln zu finden. Anwender, die nicht nur an der Anlei-

tung zu einem Paket oder einer Klasse interessiert sind, sondern sich einen Gesamtüberblick über KOMA-Script verschaffen wollen, brauchen diesen Abschnitt nur in einem der Kapitel zu lesen und können ihn beim weiteren Studium der Anleitung dann überspringen.

```
\documentclass[Optionenliste]{KOMA-Script-Klasse}
\usepackage[Optionenliste]{Paket-Liste}
```

Bei LaTeX ist vorgesehen, dass Anwender Klassenoptionen in Form einer durch Komma getrennten Liste einfacher Schlüsselwörter als optionales Argument von `\documentclass` angeben. Außer an die Klasse werden diese Optionen auch an alle Pakete weitergereicht, die diese Optionen verstehen. Ebenso ist vorgesehen, dass Anwender Paketoptionen in Form einer durch Komma getrennten Liste einfacher Schlüsselwörter als optionales Argument von `\usepackage` angeben. KOMA-Script erweitert den Mechanismus der Optionen für die KOMA-Script-Klassen und einige Pakete um weitere Möglichkeiten. So haben die meisten Optionen bei KOMA-Script zusätzlich einen Wert. Eine Option hat also nicht unbedingt nur die Form *Option*, sondern kann auch die Form *Option=Wert* haben. Bis auf diesen Unterschied arbeiten `\documentclass` und `\usepackage` bei KOMA-Script wie in [Tea05b] oder jeder LaTeX-Einführung, beispielsweise [SKPH99], beschrieben.

v3.00

Bei Verwendung einer KOMA-Script-Klasse sollten im Übrigen beim überflüssigen, expliziten Laden des Pakets typearea oder scrbase keine Optionen angegeben werden. Das ist darin begründet, dass die Klasse diese Pakete bereits ohne Optionen lädt und LaTeX das mehrmalige Laden eines Pakets mit unterschiedlicher Angabe von Optionen verweigert.

```
\KOMAoptions{Optionenliste}
\KOMAoption{Option}{Werteliste}
```

v3.00

KOMA-Script bietet bei den meisten Klassen- und Paketoptionen auch die Möglichkeit, den Wert der Optionen noch nach dem Laden der Klasse beziehungsweise des Pakets zu ändern. Man kann dann wahlweise mit der Anweisung `\KOMAoptions` die Werte einer Reihe von Optionen ändern. Jede Option der *Optionenliste* hat dabei die Form *Option=Wert*.

Einige Optionen besitzen auch einen Säumniswert (engl. *default value*). Versäumt man die Angabe eines Wertes, verwendet man die Option also einfach in der Form *Option*, so wird automatisch dieser Säumniswert angenommen.

Manche Optionen können gleichzeitig mehrere Werte besitzen. Für solche Optionen besteht die Möglichkeit, mit Hilfe von `\KOMAoption` der einen *Option* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *Werteliste* durch Komma voneinander getrennt.

Falls man in der *Optionenliste* eine Option auf einen unzulässigen Wert setzt oder die *Werteliste* einen unzulässigen Wert enthält, so wird ein Fehler gemeldet. Wird LaTeX in einem Modus verwendet, in dem im Fehlerfall Interaktionen möglich sind, so stoppt LaTeX in diesem Fall. Durch Eingabe von »`h`« erhält man dann eine Hilfe, in der auch die möglichen Werte für die entsprechende Option angegeben sind.

Soll in einem *Wert* ein Gleichheitszeichen oder ein Komma vorkommen, so ist der *Wert* in geschweifte Klammern zu setzen.

KOMA-Script bedient sich für die Realisierung dieser Möglichkeit der Anweisungen `\FamilyOptions` und `\FamilyOption` mit der Familie »`KOMA`«. Näheres zu diesen Anweisungen ist in Abschnitt 10.2, Seite 255 zu finden.

2.5. Kompatibilität zu früheren Versionen von KOMA-Script

Wer seine Dokumente im Quellcode archiviert, legt in der Regel allergrößten Wert darauf, dass bei zukünftigen LaTeX-Läufen immer wieder exakt dasselbe Ergebnis erzielt wird. In einigen Fällen ist es aber so, dass Verbesserungen und Korrekturen am Paket zu Änderungen im Verhalten, inbesondere beim Umbruch, führen. Dies ist in einigen Fällen jedoch eher unerwünscht.

`version=`*Wert*
`version=first`
`version=last`

v3.01a

Seit Version 2.96a besteht bei KOMA-Script, bei scrlttr2 bereits seit Version 2.9t und bei typearea erst seit Version 3.01b, die Wahl, ob eine Quelldatei soweit irgend möglich auch zukünftig bei einem LaTeX-Lauf zu exakt demselben Ergebnis führen soll oder ob jeweils entsprechend der Anpassungen der neusten Version zu setzen ist. Zu welcher Version Kompatibilität herzustellen ist, wird dabei über die Option `version` festgelegt. Kompatibilität zur ältesten unterstützten KOMA-Script-Version kann mit `version=first` oder `version=2.9` oder `version=2.9t` erreicht werden. Bei Angabe einer unbekannten Version als *Wert* wird eine Warnung ausgegeben und sicherheitshalber `version=first` angenommen.

Mit `version=last` kann die jeweils neuste Version ausgewählt werden. In diesem Fall wird also auf zukünftige Kompatibilität verzichtet. Wird die Option ohne Wertangabe verwendet, so wird ebenfalls `last` angenommen. Dies entspricht auch der Voreinstellung, solange keine obsolete Option verwendet wird.

Bei der Verwendung einer obsoleten Option von KOMA-Script 2 setzt KOMA-Script 3 automatisch `version=first`. In der dabei ausgegebenen Warnung wird erklärt, wie man diese Kompatibilitätsumschaltung verhindern kann. Alternativ kann man auch nach der obsoleten Option selbst eine abweichende Einstellung für Option `version` wählen.

Die Frage der Kompatibilität betrifft in erster Linie Fragen des Umbruchs. Neue Möglichkeiten, die sich nicht auf den Umbruch auswirken, sind auch dann verfügbar, wenn man per Option die Kompatibilität zu einer älteren Version ausgewählt hat. Die Option hat keine Auswirkungen auf Umbruchänderungen, die bei Verwendung einer neueren Version durch Beseitigung eindeutiger Fehler entstehen. Wer auch im Fehlerfall unbedingte Umbruchkompatibilität benötigt, sollte stattdessen mit dem Dokument auch die verwendete KOMA-Script-Version archivieren.

Es ist zu beachten, dass die Option `version` nach dem Laden des Pakets typearea nicht mehr verändert werden kann. Das Setzen mit `\KOMAoptions` oder `\KOMAoption` ist für diese Option daher nicht vorgesehen.

2.6. Einstellung des Satzspiegels und der Seitenaufteilung

Das Paket typearea bietet zwei unterschiedliche Benutzerschnittstellen, um auf die Satzspiegelkonstruktion Einfluss zu nehmen. Die wichtigste Möglichkeit ist die Angabe von Optionen. Wie im vorherigen Abschnitt erwähnt, können die Optionen dabei auf unterschiedlichen Wegen gesetzt werden.

In diesem Abschnitt wird die Klasse protokol verwendet werden. Es handelt sich dabei nicht um eine KOMA-Script-Klasse, sondern um eine hypothetische Klasse. Diese Anleitung geht von dem Idealfall aus, dass für jede Aufgabe eine dafür passende Klasse zur Verfügung steht.

`BCOR=`*`Korrektur`*

v3.00 Mit Hilfe der Option `BCOR=`*`Korrektur`* geben Sie den absoluten Wert der Bindekorrektur an, also die Breite des Bereichs, der durch die Bindung von der Papierbreite verloren geht. Dieser Wert wird in der Satzspiegelkonstruktion automatisch berücksichtigt und bei der Ausgabe wieder dem inneren beziehungsweise linken Rand zugeschlagen. Als *`Korrektur`* können Sie jede von TEX verstandene Maßeinheit angeben.

Beispiel: Angenommen, Sie erstellen einen Finanzbericht. Das Ganze soll einseitig in A4 gedruckt und anschließend in eine Klemmmappe geheftet werden. Die Klemme der Mappe verdeckt 7,5 mm. Der Papierstapel ist sehr dünn, deshalb gehen beim Knicken und Blättern durchschnittlich höchstens weitere 0,75 mm verloren. Sie schreiben dann also:

```
\documentclass[a4paper]{report}
\usepackage[BCOR=8.25mm]{typearea}
```

oder

```
\documentclass[a4paper,BCOR=8.25mm]{report}
\usepackage{typearea}
```

bei Angabe von `BCOR` als globale Option.

Bei Verwendung einer KOMA-Script-Klasse kann das explizite Laden von typearea entfallen:

```
\documentclass[BCOR=8.25mm]{scrreprt}
```

Die Option `a4paper` konnte bei scrreprt entfallen, da diese der Voreinstellung bei allen KOMA-Script-Klassen entspricht.

Setzt man die Option erst später auf einen neuen Wert, verwendet man also beispielsweise

```
\documentclass{scrreprt}
\KOMAoptions{BCOR=8.25mm}
```

so werden bereits beim Laden der Klasse scrreprt Standardeinstellungen vorgenommen. Beim Ändern der Einstellung mit Hilfe einer der Anweisung `\KOMAoptions` oder `\KOMAoption` wird dann automatisch ein neuer Satzspiegel mit neuen Randeinstellungen berechnet.

Bitte beachten Sie unbedingt, dass diese Option bei Verwendung einer der KOMA-Script-Klassen wie im Beispiel als Klassenoption oder per `\KOMAoptions` beziehungsweise `\KOMAoption` nach dem Laden der Klasse übergeben werden muss. Weder sollte das Paket typearea bei Verwendung einer KOMA-Script-Klasse explizit per `\usepackage` geladen, noch die Option dabei als optionales Argument angegeben werden. Wird die Option per `\KOMAoptions` oder `\KOMAoption` nach dem Laden des Pakets geändert, so werden Satzspiegel und Ränder automatisch neu berechnet.

`DIV=`*`Faktor`*

v3.00

Mit Hilfe der Option `DIV=`*`Faktor`* wird festgelegt, in wie viele Streifen die Seite horizontal und vertikal bei der Satzspiegelkonstruktion eingeteilt wird. Die genaue Konstruktion ist Abschnitt 2.2 zu entnehmen. Wichtig zu wissen ist, dass gilt: Je größer der *`Faktor`*, desto größer wird der Textbereich und desto kleiner die Ränder. Als *`Faktor`* kann jeder ganzzahlige Wert ab 4 verwendet werden. Bitte beachten Sie jedoch, dass sehr große Werte dazu führen können, dass Randbedingungen der Satzspiegelkonstruktion, je nach Wahl der weiteren Optionen, verletzt werden. So kann die Kopfzeile im Extremfall auch außerhalb der Seite liegen. Bei Verwendung der Option `DIV=`*`Faktor`* sind Sie für die Einhaltung der Randbedingungen sowie eine nach typografischen Gesichtspunkten günstige Zeilenlänge selbst verantwortlich.

In Tabelle 2.1 finden Sie für das Seitenformat A4 ohne Bindekorrektur die aus einigen *DIV*-Faktoren resultierenden Satzspiegelgrößen. Dabei werden die weiteren von der Schriftgröße abhängigen Nebenbedingungen nicht berücksichtigt.

Beispiel: Angenommen, Sie schreiben ein Sitzungsprotokoll. Sie verwenden dafür die Klasse protokol. Das Ganze soll doppelseitig werden. In Ihrer Firma wird die Schriftart Bookman in 12 pt verwendet. Diese Schriftart, die zu den Standard-PostScript-Schriften gehört, wird in LaTeX mit der Anweisung `\usepackage{bookman}` aktiviert. Die Schriftart Bookman läuft sehr weit, das heißt, die einzelnen Zeichen sind im Verhältnis zur Höhe relativ breit. Deshalb ist Ihnen die Voreinstellung für den *DIV*-Wert in typearea zu gering. Statt einem Wert von 12 sind Sie nach gründlichem Studium dieses Kapitels einschließlich der weiterführenden Abschnitte überzeugt, dass ein Wert 15 angebracht ist. Das Protokoll wird nicht gebunden, sondern gelocht und in einen Ordner abgeheftet. Eine Bindekorrektur ist deshalb nicht notwendig. Sie schreiben also:

Tabelle 2.1.: Satzspiegelmaße in Abhängigkeit von *DIV* bei A4 ohne Berücksichtigung von `\topskip`

	Satzspiegel		Ränder	
DIV	Breite	Höhe	oben	innen
6	105,00	148,50	49,50	35,00
7	120,00	169,71	42,43	30,00
8	131,25	185,63	37,13	26,25
9	140,00	198,00	33,00	23,33
10	147,00	207,90	29,70	21,00
11	152,73	216,00	27,00	19,09
12	157,50	222,75	24,75	17,50
13	161,54	228,46	22,85	16,15
14	165,00	233,36	21,21	15,00
15	168,00	237,60	19,80	14,00

(alle Längen in mm)

```
\documentclass[a4paper,twoside]{protokol}
\usepackage{bookman}
\usepackage[DIV=15]{typearea}
```

Als Sie fertig sind, macht man Sie darauf aufmerksam, dass die Protokolle neuerdings gesammelt und am Quartalsende alle zusammen als Buch gebunden werden. Die Bindung erfolgt als einfache Leimbindung, weil den Band ohnehin nie wieder jemand anschaut und er nur wegen ISO 9000 angefertigt wird. Für die Bindung einschließlich Biegefalz werden durchschnittlich 12 mm benötigt. Sie ändern die Optionen von typearea also entsprechend ab und verwenden die Klasse für Protokolle nach ISO 9000:

```
\documentclass[a4paper,twoside]{iso9000p}
\usepackage{bookman}
\usepackage[DIV=15,BCOR=12mm]{typearea}
```

Natürlich können Sie auch hier wieder eine KOMA-Script-Klasse verwenden:

```
\documentclass[twoside,DIV=15,BCOR=12mm]{scrartcl}
\usepackage{bookman}
```

Die Option `a4paper` konnte bei der Klasse scrartcl entfallen, da diese der Voreinstellung bei allen KOMA-Script-Klassen entspricht.

Bitte beachten Sie unbedingt, dass die Option `DIV` bei Verwendung einer der KOMA-Script-Klassen wie im Beispiel als Klassenoption oder per `\KOMAoptions` beziehungsweise `\KOMAoption` nach dem Laden der Klasse übergeben werden muss. Weder sollte das Paket typearea bei Verwendung einer KOMA-Script-Klasse explizit per `\usepackage` geladen, noch die Option dabei als optionales Argument angegeben werden. Wird die Option per

Tabelle 2.2.: *DIV*-Voreinstellungen für A4 bei Verzicht auf *BCOR*

Grundschriftgröße:	10 pt	11 pt	12 pt
DIV:	8	10	12

`\KOMAoptions` oder `\KOMAoption` nach dem Laden des Pakets geändert, so werden Satzspiegel und Rändern automatisch neu berechnet.

```
DIV=calc
DIV=classic
```

v3.00 Wie bereits in Abschnitt 2.2 erwähnt, gibt es nur für das Papierformat A4 feste Voreinstellungen für den *DIV*-Wert. Diese sind Tabelle 2.2 zu entnehmen. Wird ein anderes Papierformat gewählt, so berechnet typearea selbst einen guten *DIV*-Wert. Natürlich können Sie diese Berechnung auch für A4 wählen. Hierzu verwenden Sie einfach die Option `DIV=calc` an Stelle von `DIV=`*`Faktor`*. Selbstverständlich können Sie diese Option auch explizit bei allen anderen Papierformaten angeben. Wenn Sie die automatische Berechnung wünschen, ist diese Angabe sogar sinnvoll, da die Möglichkeit besteht, in einer Konfigurationsdatei andere Voreinstellungen zu setzen (siehe Abschnitt 15.2). Eine explizit angegebene Option `DIV=calc` überschreibt diese Vorkonfiguration aber.

Die in Abschnitt 2.3 erwähnte klassische Konstruktion, der mittelalterliche Buchseitenkanon, ist ebenfalls auswählbar. Verwenden Sie in diesem Fall an Stelle von `DIV=`*`Faktor`* oder `DIV=calc` einfach `DIV=classic`. Es wird dann ein *DIV*-Wert ermittelt, der eine möglichst gute Näherung an den mittelalterlichen Buchseitenkanon darstellt.

Beispiel: In dem bei der Option `DIV=`*`Faktor`* aufgeführten Beispiel mit der Schriftart Bookman gab es ja genau das Problem, dass man einen zur Schriftart besser passenden *DIV*-Wert haben wollte. Man könnte also in Abwandlung des ersten Beispiels auch einfach die Ermittlung dieses Wertes typearea überlassen:

```
\documentclass[a4paper,twoside]{protokol}
\usepackage{bookman}
\usepackage[DIV=calc]{typearea}
```

Bitte beachten Sie unbedingt, dass diese Option bei Verwendung einer der KOMA-Script-Klassen wie im Beispiel als Klassenoption oder per `\KOMAoptions` beziehungsweise `\KOMAoption` nach dem Laden der Klasse übergeben werden muss. Weder sollte das Paket typearea bei Verwendung einer KOMA-Script-Klasse explizit per `\usepackage` geladen, noch die Option dabei als optionales Argument angegeben werden. Wird die Option per `\KOMAoptions` oder `\KOMAoption` nach dem Laden des Pakets geändert, so werden Satzspiegel und Rändern automatisch neu berechnet.OptionDIV =`classic`

DIV=current
DIV=last

v3.00 Wenn Sie bis hier die Beispiele aufmerksam verfolgt haben, wissen Sie eigentlich bereits, wie man die Berechnung eines *DIV*-Wertes in Abhängigkeit von der gewählten Schrift erreicht, wenn eine KOMA-Script-Klasse zusammen mit einem Schriftpaket verwendet wird.

Das Problem dabei ist, dass die KOMA-Script-Klasse das Paket typearea bereits selbst lädt. Die Übergabe der Optionen als optionale Argumente von `\usepackage` ist also nicht möglich. Es würde auch nichts nützen, die Option `DIV=calc` als optionales Argument von `\documentclass` anzugeben. Diese Option würde direkt beim Laden des Pakets typearea ausgewertet. Damit würden Satzspiegel und Ränder für die LaTeX-Standardschrift und nicht für die später geladene Schrift berechnet.

Selbstverständlich ist es möglich, mit `\KOMAoptions{DIV=calc}` oder `\KOMAoption{DIV}{calc}` nach dem Laden des Schriftpakets Satzspiegel und Ränder neu berechnen zu lassen. Dabei wird dann über den Wert `calc` direkt ein *DIV*-Wert für eine gute Zeilenlänge eingefordert.

Da es aber häufig praktischer ist, die Einstellung für die Option `DIV` nicht erst nach dem Laden der Schrift vorzunehmen, sondern an herausgehobener Stelle, beispielsweise beim Laden der Klasse, bietet typearea zwei weitere symbolische Werte für diese Option.

v3.00 Mit `DIV=current` wird eine Neuberechnung von Satzspiegel und Rändern angestoßen, wobei genau der *DIV*-Wert verwendet wird, der aktuell eingestellt ist. Dies ist weniger für die Neuberechnung des Satzspiegels nach Wahl einer anderen Grundschrift von Interesse. Vielmehr ist das dann nützlich, wenn man etwa nach Änderung des Durchschusses bei Beibehaltung des Teilers *DIV* die Randbedingung sicherstellen will, dass `\textheight` abzüglich `\topskip` ein Vielfaches von `\baselineskip` sein sollte.

v3.00 Mit `DIV=last` wird eine Neuberechnung von Satzspiegel und Rändern angestoßen, wobei genau dieselbe Einstellung wie bei der letzten Berechnung verwendet wird.

Beispiel: Gehen wir wieder davon aus, dass für die Schriftart Bookman ein Satzspiegel mit guter Zeilenlänge berechnet werden soll. Gleichzeitig wird eine KOMA-Script-Klasse verwendet. Dies ist mit dem symbolischen Wert `last` und der Anweisung `\KOMAoptions` sehr einfach möglich:

```
\documentclass[BCOR=12mm,DIV=calc,twoside]
               {scrartcl}
\usepackage{bookman}
\KOMAoptions{DIV=last}
```

Wird später entschieden, dass ein anderer *DIV*-Wert verwendet werden soll, so muss nur die Einstellung im optionalen Argument von `\documentclass` geändert werden.

Eine Zusammenfassung aller möglichen symbolischen Werte für die Option `DIV` finden Sie in Tabelle 2.3. Es wird an dieser Stelle darauf hingewiesen, dass auch die Verwendung des Pakets fontenc dazu führen kann, dass LaTeX eine andere Schrift lädt.

Tabelle 2.3.: Mögliche symbolische Werte für die Option `DIV` oder das *DIV*-Argument von `\typearea[`*BCOR*`]{`*DIV*`}`

`areaset`
: Satzspiegel neu anordnen.

`calc`
: Satzspiegelberechnung einschließlich Ermittlung eines guten *DIV*-Wertes erneut durchführen.

`classic`
: Satzspiegelberechnung nach dem mittelalterlichen Buchseitenkanon (Kreisberechnung) erneut durchführen.

`current`
: Satzspiegelberechnung mit dem aktuell gültigen *DIV*-Wert erneut durchführen.

`default`
: Satzspiegelberechnung mit dem Standardwert für das aktuelle Seitenformat und die aktuelle Schriftgröße erneut durchführen. Falls kein Standardwert existiert, `calc` anwenden.

`last`
: Satzspiegelberechnung mit demselben *DIV*-Argument, das beim letzten Aufruf angegeben wurde, erneut durchführen.

Häufig wird die Satzspiegelneuberechnung im Zusammenhang mit der Veränderung des Zeilenabstandes (*Durchschuss*) benötigt. Da der Satzspiegel unbedingt so berechnet werden sollte, dass eine ganze Anzahl an Zeilen in den Textbereich passt, muss bei Verwendung eines anderen Durchschusses als des normalen der Satzspiegel für diesen Zeilenabstand neu berechnet werden.

Beispiel: Angenommen, für eine Diplomarbeit wird die Schriftgröße 10 pt bei eineinhalbzeiligem Satz zwingend gefordert. LaTeX setzt normalerweise bei 10 pt mit 2 pt Durchschuss, also 1,2-zeilig. Deshalb muss als zusätzlicher Dehnfaktor der Wert 1,25 verwendet werden. Gehen wir außerdem davon aus, dass eine Bindekorrektur von 12 mm benötigt wird. Dann könnte die Lösung dieses Problems wie folgt aussehen:

```
\documentclass[10pt,twoside,BCOR=12mm,DIV=calc]
               {scrreprt}
\linespread{1.25}
\KOMAoptions{DIV=last}
```

Da typearea selbst immer die Anweisung `\normalsize` bei Berechnung eines neuen Satzspiegels ausführt, ist es nicht zwingend notwendig, nach `\linespread` den gewählten Durchschuss mit `\selectfont` zu aktivieren, damit dieser auch tatsächlich für die Neuberechnung verwendet wird.

Das gleiche Beispiel sähe unter Verwendung des setspace-Pakets (siehe [Tob00]) wie folgt aus:

```
\documentclass[10pt,twoside,BCOR=12mm,DIV=calc]
               {scrreprt}
\usepackage{setspace}
\onehalfspacing
\KOMAoptions{DIV=last}
```

Wie man an dem Beispiel sieht, spart man sich mit dem setspace-Paket das Wissen um den korrekten Dehnungswert. Dies gilt allerdings nur für die Standardschriftgrößen 10 pt, 11 pt und 12 pt. Für alle anderen Schriftgrößen verwendet das Paket einen näherungsweise passenden Dehnungswert.

An dieser Stelle erscheint es mir angebracht, darauf hinzuweisen, dass der Zeilenabstand für die Titelseite wieder auf den normalen Wert zurückgesetzt werden sollte.

Beispiel: Ein vollständiges Beispiel wäre also:

```
\documentclass[10pt,twoside,BCOR=12mm,DIV=calc]
               {scrreprt}
\usepackage{setspace}
\onehalfspacing
\KOMAoptions{DIV=last}
\begin{document}
\title{Titel}
\author{Markus Kohm}
\begin{spacing}{1}
  \maketitle
  \tableofcontents
\end{spacing}
\chapter{Ok}
\end{document}
```

Siehe hierzu auch die Anmerkungen in Abschnitt 2.8.

Bitte beachten Sie unbedingt, dass diese Optionen auch zur Verwendung mit `\KOMAoptions` oder `\KOMAoption` nach dem Laden des Pakets vorgesehen sind und dann eine automatische Neuberechnung von Satzspiegel und Rändern auslösen.

Tabelle 2.4.: Mögliche symbolische *BCOR*-Argumente für `\typearea[`*BCOR*`]{`*DIV*`}`

`current`	Satzspiegelberechnung mit dem aktuell gültigen *BCOR*-Wert erneut durchführen.

`\typearea[`*BCOR*`]{`*DIV*`}`
`\recalctypearea`

Wird die Option `DIV` oder die Option `BCOR` nach dem Laden des Pakets typearea gesetzt, so wird intern die Anweisung `\typearea` aufgerufen. Dabei wird beim Setzen der Option `DIV` für *BCOR* intern der symbolische Wert `current` verwendet, der aus Gründen der Vollständigkeit auch in Tabelle 2.4 zu finden ist. Beim Setzen der Option `BCOR` wird für *DIV* hingegen der symbolische Wert `last` verwendet. Wollen Sie, dass Satzspiegel und Rändern stattdessen mit dem symbolischen Wert `current` für *DIV* neu berechnet werden, so können Sie direkt `\typearea{current}{current}` verwenden.

Sollen die Werte sowohl von *BCOR* als auch *DIV* geändert werden, so ist die Verwendung von `\typearea` zu empfehlen, da hierbei die Ränder und der Satzspiegel nur einmal neu berechnet werden. Hingegen werden bei `\KOMAoptions{DIV=`*DIV*`,BCOR=`*BCOR*`}` Ränder und Satzspiegel zunächst bei der Änderung für *DIV* und dann zusätzlich bei der Änderung für *BCOR* neu berechnet.

Der Befehl `\typearea` ist derzeit so definiert, dass es auch möglich ist, mitten in einem Dokument den Satzspiegel zu wechseln. Dabei werden allerdings Annahmen über den Aufbau des LaTeX-Kerns gemacht und interne Definitionen und Größen des LaTeX-Kerns verändert. Da am LaTeX-Kern nur noch zur Beseitigung von Fehlern notwendige Änderungen vorgenommen werden, ist die Wahrscheinlichkeit hoch, dass dies in zukünftigen Versionen von LaTeX 2_ε noch funktionieren wird. Eine Garantie dafür gibt es jedoch nicht. Die Verwendung innerhalb des Dokuments führt außerdem immer zu einem Seitenumbruch.

Da `\KOMAoption{DIV}{last}`, `\KOMAoptions{DIV=last}` oder `\typearea{current}{last}` für die Neuberechnung des Satzspiegels und der Ränder recht häufig benötigt werden, gibt es dafür die abkürzende Anweisung `\recalctypearea`.

v3.00

Beispiel: Wenn Ihnen die Schreibweisen

```
\KOMAoptions{DIV=last}
```

oder

```
\typearea[current]{last}
```

für die Neuberechnung von Satzspiegel und Rändern aufgrund der vielen Sonderzeichen zu umständlich ist, können Sie einfach

```
\recalctypearea
```

verwenden.

Tabelle 2.5.: Standardwerte für alle einfachen Schalter in KOMA-Script

Wert	Bedeutung
`true`[1]	aktiviert die Option
`on`	aktiviert die Option
`yes`	aktiviert die Option
`false`	deaktiviert die Option
`off`	deaktiviert die Option
`no`	deaktiviert die Option

`twoside=`*`Ein-Aus-Wert`*
`twoside=semi`

Wie in Abschnitt 2.1 erklärt, hängt die Randverteilung davon ab, ob ein Dokument ein- oder zweiseitig gesetzt werden soll. Bei einseitigem Satz sind der linke und rechte Rand gleich breit, während bei doppelseitigem Satz der innere Randanteil einer Seite nur halb so groß ist wie der jeweilige äußere Rand. Um diese Unterscheidung vornehmen zu können, muss typearea mit Option `twoside` mitgeteilt werden, ob das Dokument doppelseitig gesetzt wird. Als *`Ein-Aus-Wert`* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5 verwendet werden. Wird die Option ohne Wert-Angabe verwendet, so wird der Wert `true` angenommen, also doppelseitiger Satz verwendet. Deaktivieren der Option führt zu einseitigem Satz.

v3.00 Außer den Werten aus Tabelle 2.5 kann jedoch auch noch der Wert `semi` angegeben werden. Dieser Wert `semi` führt zu doppelseitigem Satz mit einseitigen Rändern und einseitigen, also nicht alternierenden Marginalien.

Die Option kann wahlweise als Klassenoption bei `\documentclass`, als Paketoption bei `\usepackage` oder auch nach dem Laden von typearea per `\KOMAoptions` oder `\KOMAoption` gesetzt werden. Eine Verwendung dieser Option nach dem Laden von typearea führt dabei automatisch zur Neuberechnung des Satzspiegels mittels `\recalctypearea` (siehe Seite 38). War vor der Option doppelseitiger Satz aktiv, wird noch vor der Neuberechnung auf die nächste ungerade Seite umbrochen.

`twocolumn=`*`Ein-Aus-Wert`*

Für die Berechnung eines guten Satzspiegels mit Hilfe von `DIV=calc` ist es erforderlich zu wissen, ob das Dokument ein- oder zweispaltig gesetzt wird. Da die Betrachtungen zur Zeilenlänge aus Abschnitt 2.1 dann für jede einzelne Spalte gelten, darf der Satzspiegel in doppelspaltigen Dokumenten bis zu doppelt so breit sein wie in einspaltigen Dokumenten.

Um diese Unterscheidung vornehmen zu können, muss typearea mit der Option `twocolumn` mitgeteilt werden, ob das Dokument doppelspaltig gesetzt wird. Als *`Ein-Aus-Wert`* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5 verwendet werden. Wird die Option ohne Wert-Angabe verwendet, so wird der Wert `true` angenommen, also doppelspaltiger

Satz verwendet. Ein Deaktivieren der Option führt wieder zum voreingestellten, einspaltigen Satz.

Die Option kann wahlweise als Klassenoption bei `\documentclass`, als Paketoption bei `\usepackage` oder auch nach dem Laden von typearea per `\KOMAoptions` oder `\KOMAoption` gesetzt werden. Eine Verwendung dieser Option nach dem Laden von typearea führt dabei automatischen zur Neuberechnung des Satzspiegels mittels `\recalctypearea` (siehe Seite 38).

`headinclude=`*Ein-Aus-Wert*
`footinclude=`*Ein-Aus-Wert*

Bisher wurde zwar erklärt, wie die Satzspiegelkonstruktion funktioniert und in welchem Verhältnis die Ränder zueinander und der Textkörper zur Seite steht, aber eine entscheidende Frage blieb ausgeklammert: Was ist denn eigentlich unter dem Rand zu verstehen?

Auf den ersten Blick wirkt diese Frage trivial: Der Rand ist der Teil der Seite, der oben, unten, links und rechts frei bleibt. Doch das ist nur die halbe Wahrheit. Der äußere Rand ist keineswegs immer leer. Teilweise findet man darin noch gesetzte Randnotizen (siehe den Befehl `\marginpar` beispielsweise in [SKPH99] bzw. Abschnitt 3.21).

Beim oberen und unteren Rand stellt sich die Frage, wie Kopf- und Fußzeile zu behandeln sind. Gehören diese beiden zum Textkörper oder zum jeweiligen Rand? Die Frage ist nicht einfach zu beantworten. Eindeutig ist, dass ein leerer Fuß und ein leerer Kopf zum Rand zu rechnen sind. Schließlich können sie nicht vom restlichen Rand unterschieden werden. Ein Fuß, der nur die Paginierung[2] enthält, wirkt optisch ebenfalls eher wie Rand und sollte deshalb zu diesem gerechnet werden. Für die optische Wirkung ist dabei unwesentlich, ob der Fuß beim Lesen oder Überfliegen leicht als Fuß erkannt werden kann oder nicht. Entscheidend ist, wie eine wohlgefüllte Seite bei *unscharfer Betrachtung* wirkt. Dazu bedient man sich beispielsweise seiner altersweitsichtigen Großeltern, denen man die Brille stibitzt und dann die Seite etwa einen halben Meter von der Nasenspitze entfernt hält. In Ermangelung erreichbarer Großeltern kann man sich auch damit behelfen, dass man die eigenen Augen auf Fernsicht stellt, die Seite aber nur mit ausgestreckten Armen hält. Brillenträger sind hier deutlich im Vorteil. Hat man eine Fußzeile, die neben der Paginierung weitere, weitschweifige Angaben enthält, beispielsweise einen Copyright-Hinweis, so wirkt die Fußzeile eher wie ein etwas abgesetzter Teil des Textkörpers. Bei der Berechnung des Satzspiegels sollte das berücksichtigt werden.

Bei der Kopfzeile sieht es noch schwieriger aus. In der Kopfzeile wird häufig der Kolumnentitel[3] gesetzt. Arbeitet man mit einem lebenden Kolumnentitel, also der Wiederholung der ersten bzw. zweiten Gliederungsebene in der Kopfzeile, und hat gleichzeitig sehr lange Überschriften, so erhält man automatisch sehr lange Kopfzeilen. In diesem Fall wirkt der Kopf wiederum wie ein abgesetzter Teil des Textkörpers und weniger wie leerer Rand. Verstärkt wird dieser Effekt noch, wenn neben dem Kolumnentitel auch die Paginierung im Kopf erfolgt. Dadurch erhält man einen links und rechts

[2] Unter der Paginierung versteht man die Angabe der Seitenzahl.
[3] Unter dem Kolumnentitel versteht man in der Regel die Wiederholung einer Überschrift mit Titelcharakter. Er steht häufig im Seitenkopf, seltener im Seitenfuß.

abgeschlossenen Bereich, der kaum noch als leerer Rand wirkt. Schwieriger ist es bei Paginierung im Fuß und Überschriften, deren Länge sehr stark schwankt. Hier kann der Kopf der einen Seite wie Textkörper wirken, der Kopf der anderen Seite jedoch eher wie Rand. Keinesfalls sollte man die Seiten jedoch unterschiedlich behandeln. Das würde zu vertikal springenden Köpfen führen und ist nicht einmal für ein Daumenkino geeignet. Ich rate in diesem Fall dazu, den Kopf zum Textkörper zu rechnen.

Ganz einfach fällt die Entscheidung, wenn Kopf oder Fuß durch eine Linie vom eigentlichen Textkörper abgetrennt sind. Dadurch erhält man eine geschlossene Wirkung und der Kopf bzw. Fuß sollte unbedingt zum Textkörper gerechnet werden. Wie gesagt: Die durch die Trennlinie verbesserte Erkennung des Kopfes oder Fußes ist hier unerheblich. Entscheidend ist die unscharfe Betrachtung.

Das typearea-Paket trifft die Entscheidung, ob ein Kopf oder Fuß zum Textkörper gehört oder davon getrennt zum Rand gerechnet werden muss, nicht selbst. Stattdessen kann mit den Optionen `headinclude` und `footinclude` eingestellt werden, ob der Kopf und der Fuß zum Textkörper gerechnet werden sollen. Die Optionen verstehen dabei als *Ein-Aus-Wert* die Standardwerte für einfache Schalter, die in Tabelle 2.5 angegeben sind. Man kann die Optionen auch ohne Wertzuweisung verwenden. In diesem Fall wird `true` als *Ein-Aus-Wert* verwendet, also der Kopf oder Fuß zum Satzspiegel gerechnet.

v3.00

Wenn Sie unsicher sind, was die richtige Einstellung ist, lesen Sie bitte obige Erläuterungen. Voreingestellt sind normalerweise `headinclude=false` und `footinclude=false`. Dies kann sich jedoch bei den KOMA-Script-Klassen je nach Klassenoption oder bei Verwendung anderer KOMA-Script-Pakete generell ändern (siehe Abschnitt 3.1 und Kapitel 5).

Bitte beachten Sie unbedingt, dass diese Optionen bei Verwendung einer der KOMA-Script-Klassen als Klassenoptionen oder per `\KOMAoptions` beziehungsweise `\KOMAoption` nach dem Laden der Klasse übergeben werden müssen. Eine Änderung dieser Optionen nach dem Laden von typearea führt dabei nicht zu einer automatischen Neuberechnung des Satzspiegels. Vielmehr wirkt sich die Änderung erst bei der nächsten Neuberechnung des Satzspiegels aus. Zur Neuberechnung des Satzspiegels siehe Option `DIV` mit den Werten `last` oder `current` (siehe Seite 35) oder die Anweisung `\recalctypearea` (siehe Seite 38).

`mpinclude=`*`Ein-Aus-Wert`*

v2.8q

v3.00

Neben Dokumenten, bei denen der Kopf und der Fuß der Seite eher zum Textbereich als zum Rand gehört, gibt es auch Dokumente, bei denen dies für Randnotizen zutrifft. Mit der Option `mpinclude` kann genau dies erreicht werden. Die Option versteht dabei als *Ein-Aus-Wert* die Standardwerte für einfache Schalter, die in Tabelle 2.5 angegeben sind. Man kann die Option auch ohne Wertzuweisung verwenden. In diesem Fall wird `true` als *Ein-Aus-Wert* verwendet.

Der Effekt von `mpinclude=true` ist, dass eine Breiteneinheit vom Textbereich weggenommen und als Bereich für die Randnotizen verwendet wird. Mit `mpinclude=false`, was der Voreinstellung entspricht, wird hingegen ein Teil des Randes für Randnotizen verwendet. Dies ist, je nachdem ob einseitig oder doppelseitig gearbeitet wird, ebenfalls eine Breiteneinheit

oder auch eineinhalb Breiteneinheiten. In der Regel ist die Verwendung von `mpinclude=true` nicht anzuraten und sollte Experten vorbehalten bleiben.

In den meisten Fällen, in denen die Option `mpinclude` sinnvoll ist, werden außerdem breitere Randnotizen benötigt. In sehr vielen Fällen sollte dabei aber nicht die gesamte Breite, sondern nur ein Teil davon dem Textbereich zugeordnet werden. Dies ist beispielsweise der Fall, wenn der Rand für Zitate verwendet wird. Solche Zitate werden üblicherweise im Flattersatz gesetzt, wobei die bündige Kante an den Textbereich anschließt. Da sich kein geschlossener optischer Eindruck ergibt, dürfen die flatternden Enden also durchaus teilweise in den Rand ragen. Man kann das einfach erreichen, indem man zum einen die Option `mpinclude` verwendet. Zum anderen vergrößert man die Länge `\marginparwidth` nach der Berechnung des Satzspiegels noch mit Hilfe der `\addtolength`-Anweisung. Um welchen Wert man vergrößern sollte, hängt vom Einzelfall ab und erfordert einiges Fingerspitzengefühl. Auch deshalb ist die Option `mpinclude` eher etwas für Experten. Natürlich kann man auch festlegen, dass die Randnotizen beispielsweise zu einem Drittel in den Rand hineinragen sollen, und das wie folgt erreichen:

```
\setlength{\marginparwidth}{1.5\marginparwidth}
```

Da es derzeit keine Option gibt, um mehr Platz für die Randnotizen innerhalb des Textbereichs vorzusehen, gibt es nur eine Möglichkeit, dies zu erreichen. Man verzichtet auf die Option `mpinclude` oder setzt `mpinclude` auf `false`, verringert nach der Satzspiegelberechnung die Breite des Textbereichs `\textwidth` und setzt die Breite des Bereichs der Randnotizen auf den Wert, um den man die Breite des Textbereichs verringert hat. Leider lässt sich dieses Vorgehen nicht mit der automatischen Berechnung des *DIV*-Wertes verbinden. Demgegenüber wird `mpinclude` bei `DIV=calc` (siehe Seite 35) berücksichtigt.

Bitte beachten Sie unbedingt, dass diese Option bei Verwendung einer der KOMA-Script-Klassen als Klassenoption oder per `\KOMAoptions` beziehungsweise `\KOMAoption` nach dem Laden der Klasse übergeben werden muss. Eine Änderung dieser Option nach dem Laden von typearea führt nicht zu einer automatischen Neuberechnung des Satzspiegels. Vielmehr wirkt sich die Änderung erst bei der nächsten Neuberechnung des Satzspiegels aus. Zur Neuberechnung des Satzspiegels siehe Option `DIV` mit den Werten `last` oder `current` (siehe Seite 35) oder die Anweisung `\recalctypearea` (siehe Seite 38).

`headlines=`*`Zeilenanzahl`*
`headheight=`*`Höhe`*

Es ist nun also bekannt, wie man Satzspiegel mit dem typearea-Paket berechnet und wie man dabei angibt, ob der Kopf oder Fuß zum Textkörper oder zum Rand gehört. Insbesondere für den Kopf fehlt aber noch die Angabe, wie hoch er denn eigentlich sein soll. Hierzu dienen die Optionen `headlines` und `headheight`.

v3.00

Die Option `headlines` setzt man dabei auf die Anzahl der Kopfzeilen. Normalerweise arbeitet das typearea-Paket mit 1,25 Kopfzeilen. Dieser Wert stellt einen Kompromiss dar. Zum einen ist er groß genug, um auch für eine unterstrichene Kopfzeile (siehe Abschnitt 3.1) Platz zu bieten, zum anderen ist er klein genug, um das Randgewicht nicht zu stark zu verändern,

wenn mit einer einfachen, nicht unterstrichenen Kopfzeile gearbeitet wird. Damit ist der voreingestellte Wert in den meisten Standardfällen ein guter Wert. In einigen Fällen will oder muss man aber die Kopfhöhe genauer den tatsächlichen Erfordernissen anpassen.

Beispiel: Angenommen, es soll ein Text mit einem zweizeiligen Kopf erstellt werden. Normalerweise würde dies dazu führen, dass auf jeder Seite eine Warnung »`overfull \vbox`« von LaTeX ausgegeben würde. Um dies zu verhindern, wird das typearea-Paket angewiesen, einen entsprechenden Satzspiegel zu berechnen:

```
\documentclass[a4paper]{article}
\usepackage[headlines=2.1]{typearea}
```

Es ist auch wieder möglich und bei Verwendung einer KOMA-Script-Klasse empfehlenswert, diese Option direkt an die Klasse zu übergeben:

```
\documentclass[headlines=2.1]{scrartcl}
```

Befehle, mit denen dann der Inhalt der zweizeiligen Kopfzeile definiert werden kann, sind in Kapitel 5 zu finden.

In einigen Fällen ist es nützlich, wenn man die Kopfhöhe nicht in Zeilen, sondern direkt als Längenwert angeben kann. Dies ist mit Hilfe der alternativ verwendbaren Option `headheight` möglich. Als *Höhe* sind alle Längen und Größen verwendbar, die LaTeX kennt. Es ist jedoch zu beachten, dass bei Verwendung einer LaTeX-Länge wie `\baselineskip` nicht deren Größe zum Zeitpunkt des Setzens der Option, sondern zum Zeitpunkt der Berechnung des Satzspiegels und der Rändern entscheidend ist.

Bitte beachten Sie unbedingt, dass diese Optionen bei Verwendung einer der KOMA-Script-Klassen als Klassenoptionen oder per `\KOMAoptions` beziehungsweise `\KOMAoption` nach dem Laden der Klasse übergeben werden müssen. Eine Änderung dieser Optionen nach dem Laden von typearea führt nicht zu einer automatischen Neuberechnung des Satzspiegels. Vielmehr wirkt sich die Änderung erst bei der nächsten Neuberechnung des Satzspiegels aus. Zur Neuberechnung des Satzspiegels siehe Option `DIV` mit den Werten `last` oder `current` (siehe Seite 35) oder die Anweisung `\recalctypearea` (siehe Seite 38).

`\areaset[`*BCOR*`]{`*Breite*`}{`*Höhe*`}`

Bis hier wurde nun eine Menge darüber erzählt, wie man einen guten Satzspiegel für Standardanwendungen erstellt und wie das typearea-Paket dem Anwender diese Arbeit erleichtert, ihm aber gleichzeitig Möglichkeiten der Einflussnahme bietet. Es gibt jedoch auch Fälle, in denen der Textkörper eine bestimmte Größe exakt einhalten soll, ohne dass dabei auf gute Satzspiegelkonstruktion oder auf weitere Nebenbedingungen zu achten ist. Trotzdem sollen die Ränder so gut wie möglich verteilt und dabei gegebenenfalls auch eine Bindekorrektur berücksichtigt werden. Das typearea-Paket bietet hierfür den Befehl `\areaset`, dem man neben der optionalen Bindekorrektur als Parameter die Breite und Höhe des Textbereichs übergibt. Die

Ränder und deren Verteilung werden dann automatisch berechnet, wobei gegebenenfalls auch die Einstellungen der Paketoptionen `headinclude` und `footinclude` berücksichtigt werden. Die Optionen `headlines` und `headheight` bleiben in diesem Fall jedoch unberücksichtigt!

Beispiel: Angenommen, ein Text auf A4-Papier soll genau die Breite von 60 Zeichen in der Typewriter-Schrift haben und exakt 30 Zeilen je Seite besitzen. Dann könnte mit folgender Präambel gearbeitet werden:

```
\documentclass[a4paper,11pt]{article}
\usepackage{typearea}
\newlength{\CharsLX}% Breite von 60 Zeichen
\newlength{\LinesXXX}% Hoehe von 30 Zeilen
\settowidth{\CharsLX}{\texttt{1234567890}}
\setlength{\CharsLX}{6\CharsLX}
\setlength{\LinesXXX}{\topskip}
\addtolength{\LinesXXX}{29\baselineskip}
\areaset{\CharsLX}{\LinesXXX}
```

Der Faktor von 29 statt 30 ist damit begründet, dass die Grundlinie der obersten Zeile bereits am obersten Rand des um `\topskip` verringerten Satzspiegels liegt, solange die Höhe der obersten Zeile kleiner als `\topskip` ist. Die oberste Zeile benötigt damit keine Höhe. Die Unterlängen der untersten Zeile ragen dafür unter den Satzspiegel.

Soll stattdessen ein Gedichtband gesetzt werden, bei dem es nur darauf ankommt, dass der Textbereich genau quadratisch mit einer Seitenlänge von 15 cm ist, wobei ein Binderand von 1 cm zu berücksichtigen ist, so kann dies wie folgt erreicht werden:

```
\documentclass{gedichte}
\usepackage{typearea}
\areaset[1cm]{15cm}{15cm}
```

`DIV=areaset`

v3.00 In seltenen Fällen ist es nützlich, wenn man den aktuell eingestellten Satzspiegel neu ausrichten lassen kann. Dies ist mit der Option `DIV=areaset` möglich, wobei `\KOMAoptions{DIV=areaset}` der Anweisung

```
\areaset[current]{\textwidth}{\textheight}
```

entspricht. Dasselbe Ergebnis erhält man auch, wenn `DIV=last` verwendet wird und der Satzspiegel zuletzt per `\areaset` eingestellt wurde.

Das Paket typearea ist nicht dafür gedacht, bestimmte Randbreiten einzustellen. Dafür ist das Paket geometry (siehe [Ume00]) empfehlenswert.

2.7. Einstellung des Papierformats

Das Papierformat ist ein entscheidendes Grundmerkmal eines Dokuments. Wie bereits bei der Vorstellung der unterstützten Satzspiegelkonstruktionen (siehe Abschnitt 2.1 bis Abschnitt 2.3 ab Seite 25) aufgezeigt, steht und fällt die Aufteilung der Seite und damit das gesamte Dokumentlayout mit der Wahl des Papierformats. Während die LaTeX-Standardklassen auf einige wenige Formate festgelegt sind, unterstützt KOMA-Script mit dem Paket typearea selbst ausgefallene Seitengrößen.

`paper=`*`Format`*

v3.00 Die Option `paper` ist das zentrale Element der Formatauswahl bei KOMA-Script. Als *`Format`* wird dabei zunächst das amerikanische `letter`, `legal` und `executive` unterstützt. Darüber hinaus sind die ISO-Formate der Reihen A, B, C und D möglich, also beispielsweise `A4` oder – klein geschrieben – `a4`.

v3.02c Querformate werden dadurch unterstützt, dass man die Option ein weiteres Mal mit dem Wert `landscape` oder `seascape` angibt. Dabei unterscheiden sich `landscape` und `seascape` nur darin, dass das Programm `dvips` bei `landscape` um -90° dreht, während bei `seascape` um +90° gedreht wird. Hilfreich ist `seascape` also vor allem dann, wenn ein PostScript-Anzeigeprogramm die Seiten im Querformat auf dem Kopf stellt. Damit der Unterschied eine Rolle spielt, muss auch die nachfolgend beschriebene Option `pagesize` verwendet werden.

v3.01b Zusätzlich kann das *`Format`* auch in der Form *`Breite`*`:`*`Höhe`* angegeben werden. Es wird darauf hingewiesen, dass bis Version 3.01a *`Höhe`* und *`Breite`* vertauscht waren. Dies ist insbesondere dann zu beachten, wenn mit einer entsprechenden Kompatibilitätseinstellung (siehe Option `version`, Abschnitt 2.5, Seite 31) gearbeitet wird.

Beispiel: Angenommen, es soll eine Karteikarte im Format ISO-A8 quer bedruckt werden. Dabei sollen die Ränder sehr klein gewählt werden. Außerdem wird auf eine Kopf- und eine Fußzeile verzichtet.

```
\documentclass{article}
\usepackage[headinclude=false,footinclude=false,%
            paper=A8,paper=landscape]{typearea}
\areaset{7cm}{5cm}
\pagestyle{empty}
\begin{document}
\section*{Definierte Papierformate}
letter, legal, executive, a0, a1 \dots\ %
b0, b1 \dots\ c0, c1 \dots\ d0, d1 \dots
\end{document}
```

Haben die Karteikarten das Sonderformat (Breite:Höhe) 5 cm : 3 cm, so ist dies mit

```
\documentclass{article}
\usepackage[headinclude=false,footinclude=false,
```

```
                paper=5cm:3cm]{typearea}
\areaset{4cm}{2.4cm}
\pagestyle{empty}
\begin{document}
\section*{Definierte Papierformate}
letter, legal, executive, a0, a1 \dots\ %
b0, b1 \dots\ c0, c1 \dots\ d0, d1 \dots
\end{document}
```

möglich.

In der Voreinstellung wird bei KOMA-Script mit A4-Papier in der Ausrichtung portrait gearbeitet. Dies ist ein Unterschied zu den Standardklassen, bei denen in der Voreinstellung das amerikanische Format letter verwendet wird.

Bitte beachten Sie unbedingt, dass diese Option bei Verwendung einer der KOMA-Script-Klassen als Klassenoption oder per `\KOMAoptions` beziehungsweise `\KOMAoption` nach dem Laden der Klasse übergeben werden muss. Eine Änderung des Papierformats oder der Papierausrichtung mit Hilfe der Anweisung `\KOMAoptions` oder `\KOMAoption` nach dem Laden von typearea führt nicht zu einer automatischen Neuberechnung des Satzspiegels. Vielmehr wirkt sich die Änderung erst bei der nächsten Neuberechnung des Satzspiegels aus. Zur Neuberechnung des Satzspiegels siehe Option `DIV` mit den Werten `last` oder `current` (siehe Seite 35) oder die Anweisung `\recalctypearea` (siehe Seite 38).

`pagesize=`*Ausgabetreiber*

Die oben genannten Mechanismen zur Auswahl des Papierformats haben nur insofern einen Einfluss auf die Ausgabe, als interne LaTeX-Maße gesetzt werden. Das typearea verwendet diese dann bei der Aufteilung der Seite in Ränder und Textbereich. Die Spezifikation des DVI-Formats sieht aber an keiner Stelle Angaben zum Papierformat vor. Wird direkt aus dem DVI-Format in eine Low-Level-Druckersprache wie PCL oder ESC/P2 ausgegeben, spielt dies normalerweise keine Rolle, da auch bei diesen Ausgaben der 0-Bezugspunkt wie bei DVI links oben liegt. Wird aber in Sprachen wie PostScript oder PDF übersetzt, bei denen der 0-Bezugspunkt an anderer Stelle liegt und außerdem das Papierformat in der Ausgabedatei angegeben werden sollte, so fehlt diese Information. Als Lösung des Problems verwendet der entsprechende Treiber eine voreingestellte Papiergröße, die der Anwender entweder per Option oder durch entsprechende Angabe in der TeX-Quelldatei verändern kann. Bei Verwendung des DVI-Treibers `dvips` oder `dvipdfm` kann diese Angabe in Form einer `\special`-Anweisung erfolgen. Bei pdfTeX oder VTeX werden stattdessen zwei Längen entsprechend gesetzt.

Mit der Option `pagesize` kann eingestellt werden, für welchen Ausgabetreiber die Papiergröße in das Ausgabedokument geschrieben wird. Die unterstützten Ausgabetreiber sind Tabelle 2.6 auf Seite 48 zu entnehmen. Voreingestellt ist `pagesize=false`. Die Verwendung der Option in der Form `pagesize` ohne Angabe eines Wertes entspricht `pagesize=auto`.

Tabelle 2.6.: Ausgabetreiber für Option `pagesize=`*`Ausgabetreiber`*

`auto`
: Falls die pdfTEX-spezifischen Register `\pdfpagewidth` und `\pdfpageheight` vorhanden sind, wird der Ausgabetreiber `pdftex` aktiviert. Zusätzlich wird auch der Ausgabetreiber `dvips` verwendet.

`automedia`
: Dies entspricht dem Ausgabetreiber `auto`. Allerdings werden zusätzlich auch noch die VTEX-spezifischen Register `\mediawidth` und `\mediaheight` gesetzt, falls diese definiert sind.

`false`, `no`, `off`
: Die Papiergröße wird nicht an den Ausgabetreiber gemeldet.

`dvipdfmx`
v3.05a
: Die Papiergröße wird als `\special{pagesize=`*`Breite`*`,`*`Höhe`*`}` in die DVI-Datei geschrieben. Der Name des Ausgabetreibers kommt daher, dass das Programm `dvipdfmx` eine Papierformatumschaltung über diese Anweisung auch innerhalb des Dokuments erlaubt.

`dvips`
: Bei Verwendung innerhalb der Dokumentpräambel wird die Papiergröße über `\special{pagesize=`*`Breite`*`,`*`Höhe`*`}` in das Dokument geschrieben. Da das Programm `dvips` keine Papierformatumschaltung innerhalb des Dokuments unterstützt, wird bei Bedarf im Dokument ein recht unsauberer Hack verwendet, um die Umschaltung nach Möglichkeit dennoch zu erreichen. Papierformatumschaltung nach der Dokumentprämbel bei gleichzeitiger Verwendung des Ausgabetreibers `dvips` erfolgen daher auf eigene Gefahr!

`pdftex`
: Die Papiergröße wird über die pdfTEX-spezifischen Register `\pdfpagewidth` und `\pdfpageheight` in gesetzt. Dies ist auch jederzeit innerhalb des Dokuments problemlos möglich.

Beispiel: Angenommen, es soll ein Dokument sowohl als DVI-Datei verwendet werden, als auch eine Online-Version im PDF-Format erstellt werden. Dann könnte die Präambel beispielsweise so beginnen:

```
\documentclass{article}
\usepackage[paper=A4,pagesize]{typearea}
```

Wird nun für die Bearbeitung pdfTEX verwendet *und* die PDF-Ausgabe aktiviert, so werden die beiden Spezialgrößen `\pdfpagewidth` und `\pdfpageheight` entsprechend gesetzt. Wird jedoch eine DVI-Datei erzeugt – egal ob mit LATEX oder pdfLATEX –, so wird ein `\special` an den Anfang dieser Datei geschrieben.

Es wird empfohlen, die Option `pagesize` immer anzugeben. In der Regel ist dabei die Methode ohne *`Ausgabetreiber`* oder mit `auto` oder `automedia` günstig.

2.8. Tipps

Insbesondere für die Erstellung von schriftlichen Arbeiten während des Studiums findet man häufig Vorschriften, die einer typografischen Begutachtung nicht nur in keiner Weise standhalten, sondern massiv gegen alle Regeln der Typografie verstoßen. Ursache für solche Regeln ist oft typografische Inkompetenz derjenigen, die sie herausgeben. Manchmal ist die Ursache auch im Ausgangspunkt begründet, nämlich der Schreibmaschine. Mit einer Schreibmaschine oder einer Textverarbeitung von 1980 ist es ohne erheblichen Aufwand kaum möglich, typografisch perfekte Ergebnisse zu erzielen. Also wurden einst Vorschriften erlassen, die leicht erfüllbar schienen und dem Korrektor trotzdem entgegenkommen. Dazu zählen dann Randeinstellungen, die für einseitigen Druck mit einer Schreibmaschine zu brauchbaren Zeilenlängen führen. Um nicht extrem kurze Zeilen zu erhalten, die durch Flattersatz zudem verschlimmert werden, werden die Ränder schmal gehalten und für Korrekturen stattdessen ein großer Durchschuss in Form von eineinhalbzeiligem Satz vorgeschrieben. Bevor moderne Textverarbeitungssysteme verfügbar wurden, wäre – außer mit TEX – einzeiliger Satz die einzige Alternative gewesen. Dabei wäre dann selbst das Anbringen von Korrekturzeichen schwierig geworden. Als die Verwendung von Computern für die Erstellung schriftlicher Arbeiten üblicher wurde, hat sich manches Mal auch der Spieltrieb des einen oder anderen Studenten gezeigt, der durch Verwendung einer Schmuckschrift seine Arbeit aufpeppen und so eine bessere Note mit weniger Einsatz herausschinden wollte. Nicht bedacht hat er dabei, dass solche Schriften schlechter zu lesen und deshalb für den Zweck ungeeignet sind. Damit hielten zwei Brotschriften Einzug in die Vorschriften, die weder zusammenpassen noch im Falle von Times wirklich gut geeignet sind. Times ist eine relativ enge Schrift, die Anfang des 20. Jahrhunderts speziell für schmale Spalten im englischen Zeitungssatz entworfen wurde. In modernen Schnitten ist dies etwas entschärft. Dennoch passt die häufig vorgeschriebene Times meist nicht zu den gleichzeitig gegebenen Randvorgaben.

LaTeX setzt bereits von sich aus mit ausreichendem Durchschuss. Gleichzeitig sind die Ränder bei sinnvollen Zeilenlängen groß genug, um Platz für Korrekturen zu bieten. Dabei wirkt die Seite trotz einer Fülle von Text großzügig angelegt.

Oft sind die typografisch mehr als fragwürdigen Satzvorschriften mit LaTeX auch außerordentlich schwierig umzusetzen. So kann eine feste Anzahl von »Anschlägen« nur dann eingehalten werden, wenn keine proportionale Schrift verwendet wird. Es gibt nur wenige gute nichtproportionale Schriften. Kaum ein Text, der mit einer derartigen Schrift gesetzt ist, wirkt wirklich gut. So wird häufig versucht, durch ausladende Serifen beispielsweise beim kleinen »i« oder »l« die unterschiedliche Breite der Zeichen auszugleichen. Dies kann nicht funktionieren. Im Ergebnis wirkt der Text unruhig und zerrissen. Außerdem verträgt sich eine solche Schrift kaum mit dem im deutschen Sprachraum üblichen und allgemein vorzuziehenden Blocksatz. Gewisse Vorgaben können daher bei Verwendung von LaTeX nur ignoriert oder großzügig ausgelegt werden, etwa indem man »60 Anschläge pro Zeile« nicht als feste, sondern als durchschnittliche oder maximale Angabe interpretiert.

Wie ausgeführt, sind Satzvorschriften meist dazu gedacht, ein brauchbares Ergebnis zu erhalten, auch wenn der Ausführende selbst nicht weiß, was dabei zu beachten ist. Brauchbar bedeutet häufig: lesbar und korrigierbar. Nach meiner Auffassung wird ein mit LaTeX und dem typearea-Paket gesetzter Text bezüglich des Satzspiegels diesen Anforderungen von vornherein gerecht. Wenn Sie also mit Vorschriften konfrontiert sind, die offensichtlich erheblich davon abweichen, so empfehle ich, dem Betreuer einen Textauszug vorzulegen und nachzufragen, ob es gestattet ist, die Arbeit trotz der Abweichungen in dieser Form zu liefern. Gegebenenfalls kann durch Veränderung der Option `DIV` der Satzspiegel moderat angepasst werden. Von der Verwendung von `\areaset` zu diesem Zweck rate ich jedoch ab. Schlimmstenfalls verwenden Sie das nicht zu KOMA-Script gehörende geometry-Paket (siehe [Ume00]) oder verändern Sie die Satzspiegelparameter von LaTeX selbst. Die von typearea ermittelten Werte finden Sie in der `log`-Datei Ihres Dokuments. Damit sollten moderate Anpassungen möglich sein. Achten Sie jedoch unbedingt darauf, dass die Proportionen des Textbereichs mit denen der Seite unter Berücksichtigung der Bindekorrektur annähernd übereinstimmen.

Sollte es unbedingt erforderlich sein, den Text eineinhalbzeilig zu setzen, so definieren Sie keinesfalls `\baselinestretch` um. Dieses Vorgehen wird zwar allzu häufig empfohlen, ist aber seit der Einführung von LaTeX 2_ε im Jahre 1994 obsolet. Verwenden Sie schlimmstenfalls den Befehl `\linespread`. Ich empfehle das Paket setspace, das nicht zu KOMA-Script gehört (siehe [Tob00]). Auch sollten Sie typearea nach der Umstellung des Zeilenabstandes den Satzspiegel für diesen Abstand berechnen lassen, jedoch für den Titel, besser auch für die Verzeichnisse – sowie das Literaturverzeichnis und den Index – wieder auf normalen Satz umschalten. Das setspace-Paket bietet dafür eine spezielle Umgebung und eigene Befehle.

Das typearea-Paket berechnet auch bei der Option `DIV=calc` einen sehr großzügigen Textbereich. Viele konservative Typografen werden feststellen, dass die resultierende Zeilenlänge noch zu groß ist. Der berechnete *DIV*-Wert ist ebenfalls in der `log`-Datei zum jeweiligen Dokument zu finden. Sie können also leicht nach dem ersten LaTeX-Lauf einen kleineren Wert

wählen.

Nicht selten wird mir die Frage gestellt, warum ich eigentlich kapitelweise auf einer Satzspiegelberechnung herumreite, während es sehr viel einfacher wäre, nur ein Paket zur Verfügung zu stellen, mit dem man die Ränder wie bei einer Textverarbeitung einstellen kann. Oft wird auch behauptet, ein solches Paket wäre ohnehin die bessere Lösung, da jeder selbst wisse, wie gute Ränder zu wählen seien, und die Ränder von KOMA-Script wären ohnehin nicht gut. Ich erlaube mir zum Abschluss dieses Kapitels ein passendes Zitat von Hans Peter Willberg und Friedrich Forssmann, zwei der angesehensten Typografen der Gegenwart (siehe [WF00]):

> *Das Selbermachen ist längst üblich, die Ergebnisse oft fragwürdig, weil Laien-Typografen nicht sehen, was nicht stimmt und nicht wissen können, worauf es ankommt. So gewöhnt man sich an falsche und schlechte Typografie.* [...] *Jetzt könnte der Einwand kommen, Typografie sei doch Geschmackssache. Wenn es um Dekoration ginge, könnte man das Argument vielleicht gelten lassen, da es aber bei Typografie in erster Linie um Information geht, können Fehler nicht nur stören, sondern sogar Schaden anrichten.*

Die Hauptklassen scrbook, scrreprt, scrartcl

Die Hauptklassen des KOMA-Script-Pakets sind als Äquivalent zu den LaTeX-Standardklassen angelegt. Das bedeutet, dass zu den drei Standardklassen book, report und article im KOMA-Script-Paket Entsprechungen zu finden sind. Daneben ist auch für die Standardklasse letter eine Entsprechung vorhanden. Der Briefklasse in KOMA-Script ist jedoch ein eigenes Kapitel gewidmet, da sie sich von den drei Hauptklassen grundsätzlich unterscheidet (siehe Kapitel 4).

Die einfachste Möglichkeit, an Stelle einer Standardklasse eine KOMA-Script-Klasse zu verwenden, ist das Ersetzen des Klassennamens in der Anweisung `\documentclass` entsprechend Tabelle 3.1. Man tauscht also beispielsweise `\documentclass{book}` gegen `\documentclass{scrbook}`. Der anschließende LaTeX-Lauf sollte lediglich einige Layoutänderungen mit sich bringen. Ein großer Teil der in den nachfolgenden Abschnitten beschriebenen vielfältigen Möglichkeiten und Optionen werden von den KOMA-Script-Klassen zusätzlich geboten.

Lassen Sie mich der Erläuterung der Klassen noch eine Bemerkung vorausschicken. Oft ist man sich am Anfang eines Dokuments unsicher, welche Einstellungen konkret zu wählen sind. Bei einigen Einstellungen, wie der Auswahl des Papierformats, mögen sie bereits vorab feststehen. Aber schon die Frage nach der Seitenaufteilung könnte im Voraus schwer zu beantworten sein. Andererseits sollten diese Angaben für die Haupttätigkeiten des Autors – Entwurf der Gliederung, Schreiben des Textes, Zusammenstellen von Abbildungen, Tabellen und Verzeichnissen – zunächst auch unerheblich sein. Konzentrieren Sie sich als Autor erst einmal auf den Inhalt. Wenn der dann steht, können Sie sich um die Feinheiten der Form kümmern. Neben der Auswahl der Optionen gehören dazu dann auch Dinge wie die Korrektur der Trennung und möglicherweise dezente Eingriffe in den Seitenumbruch oder die Verteilung von Abbildungen und Tabellen.

3.1. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde.

Tabelle 3.1.: Gegenüberstellung der Standardklassen und der KOMA-Script-Klassen

Standard-Klasse	KOMA-Script-Klasse
article	scrartcl
report	scrreprt
book	scrbook
letter	scrlttr2

3.2. Kompatibilität zu früheren Versionen von KOMA-Script

Es gilt sinngemäß, was in Abschnitt 2.5 geschrieben wurde. .

3.3. Entwurfsmodus

Viele Klassen und viele Pakete kennen neben dem normalen Satzmodus auch einen Entwurfsmodus. Die Unterschiede zwischen diesen beiden sind so vielfältig wie die Klassen und Pakete, die diese Unterscheidung anbieten.

`draft=`*`Ein-Aus-Wert`*

v3.00

Mit dieser Option wird zwischen Dokumenten im Entwurfsstadium und fertigen Dokumenten unterschieden. Als *`Ein-Aus-Wert`* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Bei Aktivierung der Option werden im Falle überlanger Zeilen am Zeilenende kleine, schwarze Kästchen ausgegeben. Diese Kästchen erleichtern dem ungeübten Auge, Absätze ausfindig zu machen, die manueller Nachbearbeitung bedürfen. Demgegenüber erscheinen in der Standardeinstellung `draft=false` keine solchen Kästchen. Solche Zeilen verschwinden übrigens häufig durch Verwendung des Pakets microtype [Sch10].

3.4. Seitenaufteilung

Eine Dokumentseite besteht aus unterschiedlichen Teilen, wie den Rändern, dem Kopf, dem Fuß, dem Textbereich, einer Marginalienspalte und den Abständen zwischen diesen Elementen. KOMA-Script unterscheidet dabei auch noch zwischen der Gesamtseite oder dem Papier und der sichtbaren Seite. Ohne Zweifel gehört die Aufteilung der Seite in diese unterschiedlichen Teile zu den Grundfähigkeiten einer Klasse. Bei KOMA-Script wird diese Arbeit an das Paket typearea delegiert. Dieses Paket kann auch zusammen mit anderen Klassen verwendet werden. Die KOMA-Script-Klassen laden typearea jedoch selbständig. Es ist daher weder notwendig noch sinnvoll, das Paket bei Verwendung einer KOMA-Script-Klasse auch noch explizit per `\usepackage` zu laden. Siehe hierzu auch Abschnitt 3.1.

Einige Einstellungen der KOMA-Script-Klassen haben auch Auswirkungen auf die Seitenaufteilung und umgekehrt. Diese Auswirkungen werden bei den entsprechenden Einstellungen dokumentiert.

Für die weitere Erklärung zur Wahl des Papierformats, der Aufteilung der Seite in Ränder und Satzspiegel und die Wahl von ein- oder zweispaltigem Satz sei auf die Anleitung des Pakets typearea verwiesen. Diese ist in Kapitel 2 ab Seite 25 zu finden.

`\flushbottom`
`\raggedbottom`

Insbesondere bei doppelseitigen Dokumenten ist es wünschenswert, wenn nicht nur alle ersten Zeilen eines Satzspiegels mit ihrer Grundlinie auf der gleichen Höhe liegen, sondern auch die letzten

Zeilen einer Doppelseite. Enthält eine Seite nur Text ohne Absätze und Überschriften, so hat man das automatisch. Aber bereits dann, wenn Absätze mit einem halben Grundlinienabstand markiert werden, genügt es, wenn die Anzahl der Absätze auf der einen Seite um eine ungerade Zahl von der auf der anderen Seite abweicht, damit dieses Ziel nicht mehr erreicht werden kann. Es ist dann notwendig, dass man zumindest einige der vertikalen Abstände etwas dehnt oder staucht, um das Ziel wieder zu erreichen. TEX kennt zu diesem Zweck dehn- und stauchbare Abstände und LATEX bietet dann die Möglichkeit, diesen *vertikalen Ausgleich* automatisch durchzuführen.

Wird über die Option `twoside` (siehe Abschnitt 2.4, Seite 40) doppelseitiger Satz angefordert, so wird der vertikale Ausgleich ebenfalls eingeschaltet. Man kann ihn aber auch mit `\flushbottom` jederzeit ab der aktuellen Seite explizit fordern. Umgekehrt ist es möglich, mit `\raggedbottom` den vertikalen Ausgleich ab der aktuellen Seite explizit abzuschalten. Dies entspricht der Voreinstellung bei einseitigem Satz.

KOMA-Script verwendet übrigens einen leicht modifizierten Verzicht auf den vertikalen Ausgleich.

3.5. Wahl der Schriftgröße für das Dokument

Die Grundschrift und deren Größe sind zentrale Elemente der Gestaltung eines Dokuments. Wie in Kapitel 2 ausgeführt wurde, hängt die Aufteilung zwischen Satzspiegel und Rändern wesentlich davon ab. Die Grundschrift ist dabei die Schrift, die für die Masse des Textes eines Dokuments verwendet wird. Alle davon abweichenden Einstellungen, sei es in der Form, der Dicke, der Neigung oder der Größe, stehen in einer Beziehung zur Grundschrift.

`fontsize=`*`Größe`*

Während von den Standardklassen und den meisten anderen Klassen nur eine sehr beschränkte Anzahl an Schriftgrößen unterstützt wird, bietet KOMA-Script die Möglichkeit, jede beliebige *`Größe`* für die Grundschrift anzugeben. Dabei kann als Einheit für die *`Größe`* auch jede bekannte TEX-Einheit verwendet werden. Wird die *`Größe`* ohne Einheit angegeben, so wird `pt` als Einheit angenommen.

Wird die Option innerhalb des Dokuments gesetzt, so werden ab diesem Punkt die Grundschriftgröße und alle davon abhängigen Größen geändert. Das kann beispielsweise dann nützlich sein, wenn der Anhang insgesamt in einer kleineren Schriftgröße gesetzt werden soll. Es wird darauf hingewiesen, dass bei Verwendung nach dem Laden der Klasse die Aufteilung zwischen Satzspiegel und Rändern nicht automatisch neu berechnet wird (siehe `\recalctypearea`, Abschnitt 2.4, Seite 38). Wird diese Neuberechnung jedoch vorgenommen, so erfolgt sie auf Basis der jeweils gültigen Grundschriftgröße. Die Auswirkungen des Wechsels der Grundschriftgröße auf zusätzlich geladene Pakete sind von diesen Paketen abhängig. Es können also Fehler auftreten, die nicht als Fehler von KOMA-Script angesehen werden.

Diese Option sollte keinesfalls als Ersatz für `\fontsize` (siehe [Tea05a]) missverstanden

werden. Sie sollte auch nicht an Stelle einer der von der Grundschrift abhängigen Schriftgrößenanweisungen, `\tiny` bis `\Huge`, verwendet werden!

Voreingestellt ist bei scrbook, scrreprt und scrartcl `fontsize=11pt`. Demgegenüber ist übrigens bei den Standardklassen `10pt` voreingestellt. Dies ist bei einem Wechsel von den Standardklassen zu den KOMA-Script-Klassen gegebenenfalls zu beachten.

3.6. Textauszeichnungen

LaTeX verfügt über eine ganze Reihe von Anweisungen zur Textauszeichnung. Neben der Wahl der Schriftart gehören dazu auch Befehle zur Wahl einer Textgröße oder der Textausrichtung. Näheres zu den normalerweise definierten Möglichkeiten ist [SKPH99], [Tea05b] und [Tea05a] zu entnehmen.

```
\textsuperscript{Text}
\textsubscript{Text}
```

Im LaTeX-Kern ist bereits die Anweisung `\textsuperscript` definiert, mit der *Text* höher gestellt werden kann. Leider bietet LaTeX selbst keine entsprechende Anweisung, um Text tief statt hoch zu stellen. KOMA-Script definiert dafür `\textsubscript`.

Beispiel: Sie schreiben einen Text über den menschlichen Stoffwechsel. Darin kommen hin und wieder einfache chemische Summenformeln vor. Dabei sind einzelne Ziffern tief zu stellen. Im Sinne des logischen Markups definieren Sie zunächst in der Dokumentpräambel oder einem eigenen Paket:

```
\newcommand*{\Molek}[2]{#1\textsubscript{#2}}
```

Damit schreiben Sie dann:

```
Die Zelle bezieht ihre Energie unter anderem aus
der Reaktion von \Molek C6\Molek H{12}\Molek O6
und \Molek O2 zu \Molek H2\Molek O{} und
\Molek C{}\Molek O2. Arsen (\Molek{As}{}) wirkt
sich auf den Stoffwechsel sehr nachteilig aus.
```

Das Ergebnis sieht daraufhin so aus:

> Die Zelle bezieht ihre Energie unter anderem aus der Reaktion von $C_6H_{12}O_6$ und O_2 zu H_2O und CO_2. Arsen (As) wirkt sich auf den Stoffwechsel sehr nachteilig aus.

Etwas später entscheiden Sie, dass Summenformeln grundsätzlich serifenlos geschrieben werden sollen. Nun zeigt sich, wie gut die Entscheidung für konsequentes logisches Markup war. Sie müssen nur die `\Molek`-Anweisung umdefinieren:

```
\newcommand*{\Molek}[2]{%
  \textsf{#1\textsubscript{#2}}%
}
```

Schon ändert sich die Ausgabe im gesamten Dokument:

> Die Zelle bezieht ihr Energie unter anderem aus der Reaktion von $C_6H_{12}O_6$ und O_2 zu H_2O und CO_2. Arsen (As) wirkt sich auf den Stoffwechsel sehr nachteilig aus.

```
\setkomafont{Element}{Befehle}
\addtokomafont{Element}{Befehle}
\usekomafont{Element}
```

v2.8p

Mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` ist es möglich, die *Befehle* festzulegen, mit denen die Schrift eines bestimmten *Element*s umgeschaltet wird. Theoretisch könnten als *Befehle* alle möglichen Anweisungen einschließlich Textausgaben verwendet werden. Sie sollten sich jedoch unbedingt auf solche Anweisungen beschränken, mit denen wirklich nur ein Schriftattribut umgeschaltet wird. In der Regel werden dies Befehle wie `\normalfont`, `\rmfamily`, `\sffamily`, `\ttfamily`, `\mdseries`, `\bfseries`, `\upshape`, `\itshape`, `\slshape`, `\scshape` oder einer der Befehle `\Huge`, `\huge`, `\LARGE`, `\Large`, `\large`, `\normalsize`, `\small`, `\footnotesize`, `\scriptsize` und `\tiny` sein. Die Erklärung zu diesen Befehlen entnehmen Sie bitte [SKPH99], [Tea05b] oder [Tea05a]. Auch Farbumschaltungen wie `\normalcolor` sind möglich (siehe [Car99d] und [Ker07]). Die Verwendung anderer Anweisungen, inbesondere solcher, die Umdefinierungen vornehmen oder zu Ausgaben führen, ist nicht vorgesehen. Seltsames Verhalten ist in diesen Fällen möglich und stellt keinen Fehler dar.

Mit `\setkomafont` wird die Schriftumschaltung eines Elements mit einer völlig neuen Definition versehen. Demgegenüber wird mit `\addtokomafont` die existierende Definition lediglich erweitert. Es wird empfohlen, beide Anweisungen nicht innerhalb des Dokuments, sondern nur in der Dokumentpräambel zu verwenden. Beispiele für die Verwendung entnehmen Sie bitte den Abschnitten zu den jeweiligen Elementen. Namen und Bedeutung der einzelnen Elemente sind in Tabelle 3.2 aufgelistet. Die Voreinstellungen sind den jeweiligen Abschnitten zu entnehmen.

Mit der Anweisung `\usekomafont` kann die aktuelle Schriftart auf diejenige umgeschaltet werden, die für das angegebene *Element* definiert ist.

Beispiel: Angenommen, für das Element `captionlabel` soll dieselbe Schriftart wie für `descriptionlabel` verwenden. Das erreichen Sie einfach mit:

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

Weitere Beispiele finden Sie in den Abschnitten zu den jeweiligen Elementen.

Tabelle 3.2.: Elemente, deren Schrift bei scrbook, scrreprt oder scrartcl mit `\setkomafont` und `\addtokomafont` verändert werden kann

`caption`
: Text einer Abbildungs- oder Tabellenunter- oder -überschrift (siehe Abschnitt 3.20, Seite 123)

`captionlabel`
: Label einer Abbildungs- oder Tabellenunter- oder -überschrift; Anwendung erfolgt nach dem Element `caption` (siehe Abschnitt 3.20, Seite 123)

`chapter`
: Überschrift der Ebene `\chapter` (siehe Abschnitt 3.16, Seite 93)

`chapterentry`
: Inhaltsverzeichniseintrag der Ebene `\chapter` (siehe Abschnitt 3.9, Seite 70)

`chapterentrypagenumber`
: Seitenzahl des Inhaltsverzeichniseintrags der Ebene `\chapter` abweichend vom Element `chapterentry` (siehe Abschnitt 3.9, Seite 70)

`chapterprefix`
: Kapitelnummernzeile bei Einstellung `chapterprefix=true` oder `appendixprefix=true` (siehe Abschnitt 3.16, Seite 89)

`descriptionlabel`
: Label, also das optionale Argument der `\item`-Anweisung, in einer `description`-Umgebung (Abschnitt 3.18, Seite 112)

`dictum`
: mit `\dictum` gesetzter schlauer Spruch (siehe Abschnitt 3.17, Seite 107)

`dictumauthor`
: Urheber eines schlauen Spruchs; Anwendung erfolgt nach dem Element `dictum` (siehe Abschnitt 3.17, Seite 107)

`dictumtext`
: alternative Bezeichnung für `dictum`

`disposition`
: alle Gliederungsüberschriften, also die Argumente von `\part` bis `\subparagraph` und `\minisec` sowie die Überschrift der Zusammenfassung; die Anwendung erfolgt vor dem Element der jeweiligen Gliederungsebene (siehe Abschnitt 3.16 ab Seite 88)

...

Tabelle 3.2.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`footnote`
: Marke und Text einer Fußnote (siehe Abschnitt 3.14, Seite 83)

`footnotelabel`
: Marke einer Fußnote; Anwendung erfolgt nach dem Element `footnote` (siehe Abschnitt 3.14, Seite 83)

`footnotereference`
: Referenzierung der Fußnotenmarke im Text (siehe Abschnitt 3.14, Seite 83)

`footnoterule`
v3.07
: Linie über dem Fußnotenapparat (siehe Abschnitt 3.14, Seite 86)

`labelinglabel`
: Label, also das optionale Argument der `\item`-Anweisung, und Trennzeichen, also das optionale Argument der `labeling`-Umgebung, in einer `labeling`-Umgebung (Abschnitt 3.18, Seite 113)

`labelingseparator`
: Trennzeichen, also das optionale Argument der `labeling`-Umgebung, in einer `labeling`-Umgebung; Anwendung erfolgt nach dem Element `labelinglabel` (Abschnitt 3.18, Seite 113)

`minisec`
: mit `\minisec` gesetzte Überschrift (siehe Abschnitt 3.16 ab Seite 99)

`pagefoot`
: wird nur verwendet, wenn das Paket scrpage2 geladen ist (siehe Kapitel 5, Seite 220)

`pagehead`
: alternative Bezeichnung für `pageheadfoot`

`pageheadfoot`
: Seitenkopf und Seitenfuß bei allen von KOMA-Script definierten Seitenstilen (siehe Abschnitt 3.12 ab Seite 74)

`pagenumber`
: Seitenzahl im Kopf oder Fuß der Seite (siehe Abschnitt 3.12)

`pagination`
: alternative Bezeichnung für `pagenumber`

...

Tabelle 3.2.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`paragraph`
: Überschrift der Ebene `\paragraph` (siehe Abschnitt 3.16, Seite 93)

`part`
: Überschrift der Ebene `\part`, jedoch ohne die Zeile mit der Nummer des Teils (siehe Abschnitt 3.16, Seite 93)

`partentry`
: Inhaltsverzeichniseintrag der Ebene `\part`

`partentrypagenumber`
: Seitenzahl des Inhaltsverzeichniseintrags der Ebene `\part` abweichend vom Element `partentry` (siehe Abschnitt 3.9, Seite 70)

`partnumber`
: Zeile mit der Nummer des Teils in Überschrift der Ebene `\part` (siehe Abschnitt 3.16, Seite 93)

`section`
: Überschrift der Ebene `\section` (siehe Abschnitt 3.16, Seite 93)

`sectionentry`
: Inhaltsverzeichniseintrag der Ebene `\section` (nur bei scrartcl verfügbar, siehe Abschnitt 3.9, Seite 70)

`sectionentrypagenumber`
: Seitenzahl des Inhaltsverzeichniseintrags der Ebene `\section` abweichend vom Element `sectionentry` (nur bei scrartcl verfügbar, siehe Abschnitt 3.9, Seite 70)

`sectioning`
: alternative Bezeichnung für `disposition`

`subject`
: Typisierung des Dokuments, also das Argument von `\subject` auf der Haupttitelseite (siehe Abschnitt 3.7, Seite 63)

`subparagraph`
: Überschrift der Ebene `\subparagraph` (siehe Abschnitt 3.16, Seite 93)

`subsection`
: Überschrift der Ebene `\subsection` (siehe Abschnitt 3.16, Seite 93)

...

Tabelle 3.2.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`subsubsection`	Überschrift der Ebene `\subsubsection` (siehe Abschnitt 3.16, Seite 93)
`subtitle`	Untertitel des Dokuments, also das Argument von `\subtitle` auf der Haupttitelseite (siehe Abschnitt 3.7, Seite 63)
`title`	Haupttitel des Dokuments, also das Argument von `\title` (bezüglich der Größe des Haupttitels siehe die ergänzenden Bemerkungen im Text von Abschnitt 3.7 ab Seite 63)

3.7. Dokumenttitel

Bei Dokumenten wird zwischen zwei Arten von Titeln für das gesamte Dokument unterschieden. Zum einen gibt es die Titelseiten. Hierbei steht der Dokumenttitel zusammen mit einigen zusätzlichen Informationen wie dem Autor auf einer eigenen Seite. Neben der Haupttitelseite kann es dabei weitere Titelseiten, etwa Schmutztitel, Verlagsinformationen, Widmung oder ähnliche, geben. Zum anderen gibt es den Titelkopf. Dabei erscheint der Titel lediglich am Anfang einer neuen – in der Regel der ersten – Seite. Unterhalb dieser Titelzeilen wird das Dokument beispielsweise mit der Zusammenfassung, einem Vorwort oder dem Inhaltsverzeichnis fortgesetzt.

`titlepage=`*`Ein-Aus-Wert`*

v3.00 Mit dieser Option wird ausgewählt, ob für die mit `\maketitle` (siehe Seite 61) gesetzte Titelei eigene Seiten verwendet werden, oder stattdessen die Titelei von `\maketitle` als Titelkopf gesetzt wird. Als *`Ein-Aus-Wert`* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden.

Mit `titlepage=true` oder `titlepage` wird die Titelei in Form von Titelseiten ausgewählt. Die Anweisung `\maketitle` verwendet `titlepage`-Umgebungen zum Setzen dieser Seiten, die somit normalerweise weder Seitenkopf noch Seitenfuß erhalten. Bei KOMA-Script wurde die Titelei gegenüber den Standardklassen stark erweitert.

Demgegenüber wird mit `titlepage=false` erreicht, dass ein Titelkopf (engl.: *in-page title*) gesetzt wird. Das heißt, die Titelei wird lediglich speziell hervorgehoben, auf der Seite mit dem Titel kann aber nachfolgend weiteres Material, beispielsweise eine Zusammenfassung oder ein Abschnitt gesetzt werden.

Bei den Klassen scrbook und scrreprt sind Titelseiten voreingestellt. Demgegenüber verwendet scrartcl in der Voreinstellung einen Titelkopf.

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

Grundsätzlich werden bei den Standardklassen und bei KOMA-Script alle Titelseiten in einer speziellen Umgebung, der `titlepage`-Umgebung, gesetzt. Diese Umgebung startet immer mit einer neuen Seite – im zweiseitigen Layout sogar mit einer neuen rechten Seite – im einspaltigen Modus. Für eine Seite wird der Seitenstil mit `\thispagestyle{empty}` geändert, so dass weder Seitenzahl noch Kolumnentitel ausgegeben werden. Am Ende der Umgebung wird die Seite automatisch beendet. Sollten Sie nicht das automatische Layout der Titelei, wie es das nachfolgend beschriebene `\maketitle` bietet, verwenden können, ist zu empfehlen, eine eigene Titelei mit Hilfe dieser Umgebung zu entwerfen.

Beispiel: Angenommen, Sie wollen eine Titelseite, auf der lediglich oben links möglichst groß und fett das Wort »Me« steht – kein Autor, kein Datum, nichts weiter. Folgendes Dokument ermöglicht das:

```
\documentclass{scrbook}
\begin{document}
  \begin{titlepage}
    \textbf{\Huge Me}
  \end{titlepage}
\end{document}
```

Einfach? Stimmt.

`\maketitle[`*Seitenzahl*`]`

Während bei den Standardklassen nur maximal eine Titelseite mit den drei Angaben Titel, Autor und Datum existiert, können bei KOMA-Script mit `\maketitle` bis zu sechs Titelseiten gesetzt werden. Im Gegensatz zu den Standardklassen kennt `\maketitle` bei KOMA-Script außerdem noch ein optionales nummerisches Argument. Findet es Verwendung, so wird die Nummer als Seitenzahl der ersten Titelseite benutzt. Diese Seitenzahl wird jedoch nicht ausgegeben, sondern beeinflusst lediglich die Zählung. Sie sollten hier unbedingt eine ungerade Zahl wählen, da sonst die gesamte Zählung durcheinander gerät. Meiner Auffassung nach gibt es nur zwei sinnvolle Anwendungen für das optionale Argument. Zum einen könnte man dem Schmutztitel die logische Seitenzahl -1 geben, um so die Seitenzählung erst ab der Haupttitelseite mit 1 zu beginnen. Zum anderen könnte man mit einer höheren Seitenzahl beginnen, beispielsweise 3, 5 oder 7, um so weitere Titelseiten zu berücksichtigen, die erst vom Verlag hinzugefügt werden. Wird ein Titelkopf verwendet, wird das optionale Argument ignoriert. Dafür kann der Seitenstil einer solchen Titelei durch Umdefinierung des Makros `\titlepagestyle` (siehe Abschnitt 3.12, Seite 78) verändert werden.

Die folgenden Anweisungen führen nicht unmittelbar zum Setzen der Titelei. Das Setzen der Titelei erfolgt immer mit `\maketitle`. Es sei an dieser Stelle auch darauf hingewiesen, dass `\maketitle` nicht innerhalb einer `titlepage`-Umgebung zu verwenden ist. Wie in den Beispielen angegeben, sollte man nur entweder `\maketitle` oder `titlepage` verwenden.

Mit den nachfolgend erklärten Anweisungen werden lediglich die Inhalte der Titelei festgelegt. Sie müssen daher auch unbedingt vor `\maketitle` verwendet werden. Es ist jedoch nicht notwendig und bei Verwendung des babel-Pakets (siehe [Bra01]) auch nicht empfehlenswert, diese Anweisungen in der Dokumentpräambel vor `\begin{document}` zu verwenden. Beispieldokumente finden Sie am Ende des Abschnitts.

`\extratitle{`*Schmutztitel*`}`

Früher war der Buchblock oftmals nicht durch einen Buchdeckel vor Verschmutzung geschützt. Diese Aufgabe übernahm dann die erste Seite des Buches, die meist einen Kurztitel, eben den *Schmutztitel* trug. Auch heute noch wird diese Extraseite vor dem eigentlichen Haupttitel gerne verwendet und enthält dann Verlagsangaben, Buchreihennummer und ähnliche Angaben.

Bei KOMA-Script ist es möglich, vor der eigentlichen Titelseite eine weitere Seite zu setzen. Als *Schmutztitel* kann dabei beliebiger Text – auch mehrere Absätze – gesetzt werden. Der Inhalt von *Schmutztitel* wird von KOMA-Script ohne zusätzliche Beeinflussung der Formatierung ausgegeben. Dadurch ist dessen Gestaltung völlig dem Anwender überlassen. Die Rückseite des Schmutztitels bleibt leer. Der Schmutztitel ergibt auch dann eine eigene Titelseite, wenn mit Titelköpfen gearbeitet wird. Die Ausgabe des mit `\extratitle` definierten Schmutztitels erfolgt als Bestandteil der Titelei mit `\maketitle`.

Beispiel: Kommen wir auf das Beispiel von oben zurück und gehen davon aus, dass das spartanische »Me« nur den Schmutztitel darstellt. Nach dem Schmutztitel soll noch der Haupttitel folgen. Dann kann wie folgt verfahren werden:

```
\documentclass{scrbook}
\begin{document}
  \extratitle{\textbf{\Huge Me}}
  \title{It's me}
  \maketitle
\end{document}
```

Sie können den Schmutztitel aber auch horizontal zentriert und etwas tiefer setzen:

```
\documentclass{scrbook}
\begin{document}
  \extratitle{\vspace*{4\baselineskip}
    \begin{center}\textbf{\Huge Me}\end{center}}
  \title{It's me}
  \maketitle
\end{document}
```

Tabelle 3.3.: Voreinstellungen der Schrift für die Elemente des Titels

Element-Name	Voreinstellung
subject	\normalfont\normalcolor\bfseries\Large
title	\usekomafont{disposition}
subtitle	\usekomafont{title}\large

Die Anweisung \title ist beim Setzen der Titelei mit Hilfe von \maketitle grundsätzlich notwendig, damit die Beispiele fehlerfrei sind. Sie wird nachfolgend erklärt.

```
\titlehead{Kopf}
\subject{Typisierung}
\title{Titel}
\subtitle{Untertitel}
\author{Autor}
\date{Datum}
\publishers{Verlag}
\and
\thanks{Fußnote}
```

Für den Inhalt der Haupttitelseite stehen sieben Elemente zur Verfügung. Die Ausgabe der Haupttitelseite erfolgt als Bestandteil der Titelei mit \maketitle, während die hier aufgeführten Anweisungen lediglich der Definition der entsprechenden Elemente dienen.

Der *Kopf* des Haupttitels wird mit der Anweisung \titlehead definiert. Er wird über die gesamte Textbreite in normalem Blocksatz am Anfang der Seite ausgegeben. Er kann vom Anwender frei gestaltet werden.

v2.95 Die *Typisierung* wird unmittelbar über dem *Titel* ausgegeben. Dabei findet die Schriftumschaltung für das Element subject Anwendung. Die Voreinstellung, die in Tabelle 3.3 zu finden ist, kann mit Hilfe der Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 3.6, Seite 56) verändert werden.

v2.8p Der *Titel* wird in einer sehr großen Schrift ausgegeben. Dabei findet abgesehen von der Größe auch die Schriftumschaltung für das Element title Anwendung. Voreingestellt ist die gleiche Schrift, die für das Element disposition verwendet wird (siehe Tabelle 3.2, Seite 57). Die Voreinstellungen können mit Hilfe der Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 3.6, Seite 56) verändert werden. Die Größe ist jedoch davon unabhängig (siehe Tabelle 3.4, Seite 64).

v2.97c Der *Untertitel* wird in der über das Element subtitle eingestellten Schrift knapp unter dem Titel ausgegeben. Die Voreinstellung, die Tabelle 3.3 zu entnehmen ist, kann mit Hilfe der Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 3.6, Seite 56) verändert werden.

Tabelle 3.4.: Schriftgröße und Ausrichtung der Elemente der Haupttitelseite bei Verwendung von `\maketitle`

Element	Anweisung	Schrift	Satz
Seitenkopf	`\titlehead`	`\normalsize`	Block-
Typisierung	`\subject`	`\usekomafont{subject}`	zentriert
Titel	`\title`	`\huge\usekomafont{title}`	zentriert
Untertitel	`\subtitle`	`\usekomafont{subtitle}`	zentriert
Autoren	`\author`	`\Large`	zentriert
Datum	`\date`	`\Large`	zentriert
Verlag	`\publishers`	`\Large`	zentriert

Unter dem *Untertitel* folgt der *Autor*. Es kann auch durchaus mehr als ein Autor innerhalb des Arguments von `\author` angegeben werden. Die Autoren sind dann mit `\and` voneinander zu trennen.

Unter dem Autor oder den Autoren folgt das Datum. Dabei ist das aktuelle Datum, `\today`, voreingestellt. Es kann jedoch mit `\date` eine beliebige Angabe – auch ein leere – erreicht werden.

Als Letztes folgt schließlich der *Verlag*. Selbstverständlich kann diese Anweisung auch für andere Angaben geringer Wichtigkeit verwendet werden. Notfalls kann durch Verwendung einer `\parbox` über die gesamte Seitenbreite auch erreicht werden, dass diese Angabe nicht zentriert, sondern im Blocksatz gesetzt wird. Sie ist dann als Äquivalent zum Kopf zu betrachten. Dabei ist jedoch zu beachten, dass sie oberhalb von eventuell vorhandenen Fußnoten ausgegeben wird.

Fußnoten werden auf der Titelseite nicht mit `\footnote`, sondern mit der Anweisung `\thanks` erzeugt. Sie dienen in der Regel für Anmerkungen bei den Autoren. Als Fußnotenzeichen werden dabei Symbole statt Zahlen verwendet. Es ist zu beachten, dass `\thanks` innerhalb des Arguments einer der übrigen Anweisungen, beispielsweise im Argument *Autor* der Anweisung `\author` zu verwenden ist.

Bis auf den *Kopf* und eventuelle Fußnoten werden alle Ausgaben horizontal zentriert. Diese Angaben sind noch einmal kurz zusammengefasst in Tabelle 3.4 zu finden.

Beispiel: Nehmen wir nun einmal an, Sie schreiben eine Diplomarbeit. Dabei sei vorgegeben, dass die Titelseite oben linksbündig das Institut einschließlich Adresse und rechtsbündig das Semester wiedergibt. Wie üblich ist ein Titel einschließlich Autor und Abgabedatum zu setzen. Außerdem soll der Betreuer angegeben und zu erkennen sein, dass es sich um eine Diplomarbeit handelt. Sie könnten das wie folgt erreichen:

```
\documentclass{scrbook}
\usepackage[ngerman]{babel}
\begin{document}
\titlehead{{\Large Universit"at Schlauenheim
    \hfill SS~2001\\}
```

```
  Institut f"ur Raumkr"ummung\\
  Hochschulstra"se~12\\
  34567 Schlauenheim}
\subject{Diplomarbeit}
\title{Digitale Raumsimulation mit dem
  DSP\,56004}
\subtitle{Klein aber fein?}
\author{cand. stup. Uli Ungenau}
\date{30. Februar 2001}
\publishers{Betreut durch
  Prof. Dr. rer. stup. Naseweis}
\maketitle
\end{document}
```

Ein häufiges Missverständnis betrifft die Bedeutung der Haupttitelseite. Irrtümlich wird oft angenommen, es handle sich dabei um den Buchumschlag oder Buchdeckel. Daher wird häufig erwartet, dass die Titelseite nicht den Randvorgaben für doppelseitige Satzspiegel gehorcht, sondern rechts und links gleich große Ränder besitzt.

Nimmt man jedoch einmal ein Buch zur Hand und klappt es auf, trifft man sehr schnell auf mindestens eine Titelseite unter dem Buchdeckel innerhalb des so genannten Buchblocks. Genau diese Titelseiten werden mit `\maketitle` gesetzt.

Wie beim Schmutztitel handelt es sich also auch bei der Haupttitelseite um eine Seite innerhalb des Buchblocks, die deshalb dem Satzspiegel des gesamten Dokuments gehorcht. Überhaupt ist ein Buchdeckel, das *Cover*, etwas, was man in einem getrennten Dokument erstellt. Schließlich hat er oft eine sehr individuelle Gestalt. Es spricht auch nichts dagegen, hierfür ein Grafik- oder DTP-Programm zu Hilfe zu nehmen. Ein getrenntes Dokument sollte auch deshalb verwendet werden, weil es später auf ein anderes Druckmedium, etwa Karton, und möglicherweise mit einem anderen Drucker ausgegeben werden soll.

`\uppertitleback{`*`Titelrückseitenkopf`*`}`
`\lowertitleback{`*`Titelrückseitenfuß`*`}`

Im doppelseitigen Druck bleibt bei den Standardklassen die Rückseite des Blatts mit der Titelseite leer. Bei KOMA-Script lässt sich die Rückseite der Haupttitelseite hingegen für weitere Angaben nutzen. Dabei wird zwischen genau zwei Elementen unterschieden, die der Anwender frei gestalten kann: dem *`Titelrückseitenkopf`* und dem *`Titelrückseitenfuß`*. Dabei kann der Kopf bis zum Fuß reichen und umgekehrt. Nimmt man diese Anleitung als Beispiel, so wurde der Haftungsausschluss mit Hilfe von `\uppertitleback` gesetzt.

`\dedication{`*`Widmung`*`}`

KOMA-Script bietet eine eigene Widmungsseite. Diese Widmung wird zentriert und in etwas größerer Schrift gesetzt. Die Rückseite ist wie bei der Seite mit dem Schmutztitel grundsätzlich

leer. Die Widmungsseite wird zusammen mit der restlichen Titelei mit `\maketitle` ausgegeben und muss daher vor dieser Anweisung definiert sein.

Beispiel: Nehmen wir dieses Mal an, dass Sie einen Gedichtband schreiben, den Sie Ihrer Frau widmen wollen. Das könnte wie folgt aussehen:

```
\documentclass{scrbook}
\usepackage[ngerman]{babel}
\begin{document}
\extratitle{\textbf{\Huge In Liebe}}
\title{In Liebe}
\author{Prinz Eisenherz}
\date{1412}
\lowertitleback{%
  Dieser Gedichtband wurde mit Hilfe von
  {\KOMAScript} und {\LaTeX} gesetzt.}
\uppertitleback{Selbstverlach\par
  Auf"|lage: 1 Exemplar}
\dedication{Meinem Schnuckelchen\\
  in ewiger Liebe\\
  von Deinem Hasenboppelchen.}
\maketitle
\end{document}
```

Ich bitte, die Kosenamen nach eigenen Vorlieben zu ersetzen.

3.8. Zusammenfassung

Insbesondere bei Artikeln, seltener bei Berichten findet man unmittelbar unter der Titelei und noch vor dem Inhaltsverzeichnis eine Zusammenfassung. Bei Verwendung eines Titelkopfes ist die Zusammenfassung in der Regel ein rechts und links eingezogener Block. Im Vergleich dazu wird bei Verwendung von Titelseiten die Zusammenfassung eher als Kapitel oder Abschnitt gesetzt.

`abstract=`*`Ein-Aus-Wert`*

scrreprt, v3.00, scrartcl

Bei den Standardklassen setzt die `abstract`-Umgebung noch den zentrierten Titel »Zusammenfassung« vor die Zusammenfassung. Früher war dies durchaus üblich. Inzwischen sind wir durch das Zeitunglesen darin geübt, einen entsprechend hervorgehobenen Text am Anfang eines Artikels oder Berichts als Zusammenfassung zu erkennen. Dies gilt umso mehr, wenn dieser Text noch vor dem Inhaltsverzeichnis steht. Zudem verwundert es, wenn ausgerechnet diese Überschrift klein und zentriert ist. KOMA-Script bietet mit der Option `abstract` die Möglichkeit, die Überschrift über der Zusammenfassung ein- oder auszuschalten. Als *`Ein-Aus-Wert`* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Voreingestellt ist bei KOMA-Script `false`.

Bei Büchern wird in der Regel eine andere Art der Zusammenfassung verwendet. Dort setzt man ein entsprechendes Kapitel an den Anfang oder Ende des Werks. Oft wird diese Zusammenfassung entweder mit der Einleitung oder einem weiteren Ausblick verknüpft. Daher gibt es bei scrbook überhaupt keine `abstract`-Umgebung. Bei Berichten im weiteren Sinne, etwa einer Studien- oder Diplomarbeit, ist ebenfalls eine Zusammenfassung in dieser Form zu empfehlen.

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

scrartcl, scrreprt

Einige LaTeX-Klassen bieten eine spezielle Umgebung für diese Zusammenfassung, die `abstract`-Umgebung. Diese wird unmittelbar ausgegeben, ist also nicht Bestandteil der mit `\maketitle` gesetzten Titelei. Bitte beachten Sie unbedingt, dass es sich bei `abstract` um eine Umgebung und nicht um eine Anweisung handelt. Ob die Zusammenfassung mit einer Überschrift versehen wird oder nicht, wird über die Option `abstract` gesteuert (siehe oben).

Bei Büchern ist die Zusammenfassung häufig Bestandteil der Einleitung oder eines gesonderten Kapitels am Ende des Dokuments. Daher gibt es bei scrbook keine `abstract`-Umgebung. Bei Verwendung der Klasse scrreprt ist es sicher eine Überlegung wert, ob man nicht genauso verfahren sollte. Siehe hierzu in Abschnitt 3.16 die Anweisungen `\chapter*` und `\addchap` oder `\addchap*` (Seite 99).

Wird ein Titelkopf (siehe Option `titlepage`, Abschnitt 3.7, Seite 60) verwendet, so wird die Zusammenfassung intern mit Hilfe einer `quotation`-Umgebung (siehe Abschnitt 3.18, Seite 116) gesetzt. Dabei werden Absatzanfänge normalerweise mit Einzug gesetzt. Soll der erste Absatz nicht eingezogen werden, so kann dieser Einzug mit `\noindent` unmittelbar nach `\begin{abstract}` unterdrückt werden.

3.9. Inhaltsverzeichnis

Auf die Titelei und eine eventuell vorhandene Zusammenfassung folgt normalerweise das Inhaltsverzeichnis. Häufig findet man nach dem Inhaltsverzeichnis auch noch die Verzeichnisse der Gleitumgebungen, beispielsweise von Tabellen und Abbildungen (siehe Abschnitt 3.20).

`toc=`*`Einstellung`*

Neuerdings ist es fast schon üblich geworden Tabellen- und Abbildungsverzeichnis sowie das Literaturverzeichnis, seltener das Stichwortverzeichnis, ins Inhaltsverzeichnis aufzunehmen. Dies hat sicher auch mit der neuen Mode zu tun, Abbildungs- und Tabellenverzeichnis ans Buchende zu stellen. Beide Verzeichnisse haben von Aufbau und Intention eine deutliche Ähnlichkeit mit dem Inhaltsverzeichnis. Daher betrachte ich die Entwicklung skeptisch. Da es keinen Sinn hat, nur das Tabellen- oder nur das Abbildungsverzeichnis ohne das jeweils andere ins Inhaltsverzeichnis aufzunehmen, werden mit der *`Einstellung`* `listof` beide Verzeichnisse

v3.00

gemeinsam ins Inhaltsverzeichnis aufgenommen. Dabei werden auch Verzeichnisse berücksichtigt, die mit Hilfe des float-Pakets ab Version 1.2e (siehe [Lin01]) oder floatrow (siehe [Lap06]) erstellt werden. Als Verzeichnisse, die den Inhalt anderer Abschnitte des Werks aufführen, erhalten Tabellen-, Abbildungs- und die mit den genannten Paketen erzeugten Verzeichnisse grundsätzlich keine Kapitelnummer. Wer diesen Grundsatz ignorieren will, bedient sich der *Einstellung* `listofnumbered`.

Das Stichwortverzeichnis erhält mit `toc=index` einen Eintrag im Inhaltsverzeichnis. Da das Stichwortverzeichnis ebenfalls nur Verweise auf den Inhalt anderer Abschnitte enthält, wird auch dieser Eintrag grundsätzlich nicht nummeriert. Eine Abweichung von diesem Grundsatz wird von KOMA-Script nicht unterstützt.

Das Literaturverzeichnis stellt eine etwas andere Art von Verzeichnis dar. Hier wird nicht der Inhalt des vorliegenden Werks aufgelistet, sondern auf externe Inhalte verwiesen. Mit dieser Begründung könnte man argumentieren, dass das Literaturverzeichnis ein eigenes Kapitel bzw. einen eigenen Abschnitt darstelle und somit eine Nummer verdiene. Die Option `toc=bibliographynumbered` führt genau dazu, einschließlich des dann fälligen Eintrags im Inhaltsverzeichnis. Ich selbst bin allerdings der Meinung, dass bei dieser Argumentation auch ein klassisches, kommentiertes Quellenverzeichnis ein eigenes Kapitel wäre. Außerdem ist das Literaturverzeichnis letztlich nichts, was man selbst geschrieben hat. Deshalb erscheint mir allenfalls ein nicht nummerierter Eintrag im Inhaltsverzeichnis angemessen, was mit der Einstellung `toc=bibliography` erreicht wird.

v2.8q

Normalerweise wird das Inhaltsverzeichnis so formatiert, dass die Gliederungsebenen unterschiedlich weit eingezogen werden. Dabei wird für die Gliederungsnummer jeder Ebene ein Raum fester Breite vorgesehen, in dem die Nummer linksbündig gesetzt wird. Dies entspricht

v3.00

der Einstellung `toc=graduated`.

Werden sehr viele Gliederungspunkte verwendet, so werden die Gliederungsnummern sehr breit. Damit reicht der vorgesehene Platz nicht aus. In [Wik] wird für solche Fälle vorgeschlagen, die Erzeugung des Inhaltsverzeichnisses umzudefinieren. KOMA-Script bietet jedoch eine alternative Formatierung an, bei der das Problem nicht auftritt. Bei Verwendung der Option `toc=flat` werden die unterschiedlichen Gliederungsebenen nicht unterschiedlich weit eingezogen. Stattdessen wird eine tabellenartige Form gewählt, in der alle Gliederungsnummern und alle Gliederungstexte jeweils in einer Spalte linksbündig untereinander stehen. Der für die Gliederungsnummern benötigte Platz wird dabei automatisch ermittelt.

Einen Überblick über alle möglichen Werte für die *Einstellung* von `toc` ist in Tabelle 3.5 zu finden.

Tabelle 3.5.: Mögliche Werte für Option `toc` zur Einstellung von Form und Inhalt des Inhaltsverzeichnisses

`bibliography`, `bib`
: Das Literaturverzeichnis erhält einen Eintrag im Inhaltsverzeichnis, ohne dass es nummeriert wird.

`bibliographynumbered`, `bibnumbered`, `numberedbibliography`, `numberedbib`
: Das Literaturverzeichnis erhält einen Eintrag im Inhaltsverzeichnis und wird nummeriert.

`flat`, `left`
: Das Inhaltsverzeichnis erhält eine tabellarische Form. Die Gliederungsnummern sind dabei die erste Spalte, die Überschriften die zweite Spalte, die Seitenzahlen die dritte Spalte. Der Platz, der für die Gliederungsnummern reserviert wird, richtet sich nach dem benötigten Platz des vorherigen LaTeX-Laufs.

`graduated`, `indent`, `indented`
: Das Inhaltsverzeichnis erhält eine hierarchische Form. Es steht nur ein begrenzter Platz für die Gliederungsnummern zur Verfügung.

`index`, `idx`
: Das Stichwortverzeichnis erhält einen Eintrag im Inhaltsverzeichnis, ohne dass es nummeriert wird.

`listof`
: Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten einen Eintrag im Inhaltsverzeichnis, ohne dass sie nummeriert werden.

`listofnumbered`, `numberedlistof`
: Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten einen Eintrag im Inhaltsverzeichnis und werden nummeriert.

`nobibliography`, `nobib`
: Das Literaturverzeichnis erhält keinen Eintrag im Inhaltsverzeichnis.

`noindex`, `noidx`
: Das Stichwortverzeichnis erhält keinen Eintrag im Inhaltsverzeichnis.

...

Tabelle 3.6.: Voreinstellungen der Schrift für die Elemente des Inhaltsverzeichnisses

Element	Voreinstellung
`partentry`	`\usekomafont{disposition}\large`
`partentrypagenumber`	
`chapterentry`	`\usekomafont{disposition}`
`chapterentrypagenumber`	
`sectionentry`	`\usekomafont{disposition}`
`sectionentrypagenumber`	

Tabelle 3.5.: Mögliche Werte für Option `toc` (*Fortsetzung*)

`nolistof`	Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten keinen Eintrag im Inhaltsverzeichnis.

`\tableofcontents`

Die Ausgabe des Inhaltsverzeichnisses wird mit `\tableofcontents` erreicht. Um ein korrektes Inhaltsverzeichnis zu erhalten, sind nach jeder Änderung mindestens zwei LaTeX-Läufe notwendig. Mit der oben erläuterten Option `toc` kann der Umfang und die Form des Inhaltsverzeichnisses beeinflusst werden. Nach einer Umschaltung der Option sind ebenfalls mindestens zwei weitere LaTeX-Läufe notwendig.

Der Eintrag für die oberste Gliederungsebene unter `\part`, also `\chapter` bei scrbook und scrreprt beziehungsweise `\section` bei scrartcl, sowie der Gliederungsebene `\part` selbst wird nicht eingerückt. Gleichzeitig befinden sich zwischen dem Text der Gliederungsebene und der Seitenzahl keine Pünktchen. Die typografischen Gründe dafür liegen in der normalerweise anderen Schriftart sowie der erwünschten Hervorhebung. Das Inhaltsverzeichnis dieser Anleitung

v2.97c

ist mit den Voreinstellungen gesetzt und dient als Beispiel. Die Schrift dieser beiden oberen Inhaltsverzeichniseinträge ist außerdem über die Elemente `partentry` und für scrbook und scrreprt über `chapterentry` beziehungsweise `sectionentry` für scrartcl einstellbar. Die Schrift der Seitenzahlen kann jeweils davon abweichend über die Elemente `partentrypagenumber` und `chapterentrypagenumber` respektive `sectionentrypagenumber` eingestellt werden (siehe auch Abschnitt 3.6, Seite 56, sowie Tabelle 3.2, Seite 57). Die Voreinstellungen der Elemente ist Tabelle 3.6 zu entnehmen.

`tocdepth`

Normalerweise werden bei den Klassen scrbook und scrreprt die Gliederungsebenen `\part` bis `\subsection` und bei der Klasse scrartcl die Ebenen `\part` bis `\subsubsection` in das Inhaltsverzeichnis aufgenommen. Dies wid über den Zähler `tocdepth` gesteuert. Dabei steht

der Wert −1 für `\part`, 0 für `\chapter` und so weiter. Durch Setzen oder Erhöhen oder Verringern des Zählers kann bestimmt werden, bis zu welcher Gliederungsebene Einträge in das Inhaltsverzeichnis erfolgen sollen. Dies ist übrigens bei den Standardklassen ganz genauso.

Bei Verwendung des scrpage2-Pakets (siehe Kapitel 5) muss man sich nicht die nummerischen Werte der einzelnen Gliederungsebenen merken. Dann stehen dafür die Makros `\partlevel`, `\chapterlevel`, `\sectionlevel` und so weiter bis hinunter zu `\subparagraphlevel` zur Verfügung.

Beispiel: Angenommen, Sie setzen einen Artikel, bei dem die Gliederungsebene `\subsubsection` verwendet wird. Gehen wir weiter davon aus, dass Sie diese Gliederungsebene aber nicht im Inhaltsverzeichnis haben wollen. Dann könnte die Präambel Ihres Dokuments wie folgt aussehen:

```
\documentclass{scrartcl}
\setcounter{tocdepth}{2}
```

Sie setzen den Zähler `tocdepth` also auf 2, weil Sie wissen, dass dies der Wert für `\subsection` ist. Wissen Sie stattdessen nur, dass normalerweise bei scrartcl Einträge in das Inhaltsverzeichnis bis zur Ebene `\subsubsection` erfolgen, können Sie auch einfach vom voreingestellten Wert des Zählers `tocdepth` eins abziehen:

```
\documentclass{scrartcl}
\addtocounter{tocdepth}{-1}
```

Wie viel Sie von `tocdepth` subtrahieren oder dazu addieren müssen, können Sie natürlich auch einfach nach einem ersten LaTeX-Lauf im Inhaltsverzeichnis abzählen.

3.10. Absatzauszeichnung

Die Standardklassen setzen Absätze normalerweise mit Absatzeinzug und ohne Absatzabstand. Bei Verwendung eines normalen Satzspiegels, wie ihn typearea bietet, ist dies die vorteilhafteste Absatzauszeichnung. Würde man ohne Einzug und Abstand arbeiten, hätte der Leser als Anhaltspunkt nur die Länge der letzten Zeile. Im Extremfall kann es sehr schwer sein, zu erkennen, ob eine Zeile voll ist oder nicht. Des Weiteren stellt der Typograf fest, dass die Auszeichnung des Absatzendes am Anfang der nächsten Zeile leicht vergessen ist. Demgegenüber ist eine Auszeichnung am Absatzanfang einprägsamer. Der Absatzabstand hat den Nachteil, dass er in verschiedenem Zusammenhang leicht verloren geht. So wäre nach einer abgesetzten Formel nicht mehr festzustellen, ob der Absatz fortgesetzt wird oder ein neuer beginnt. Auch am Seitenanfang müsste zurückgeblättert werden, um feststellen zu können, ob mit der Seite auch ein neuer Absatz beginnt. All diese Probleme sind beim Absatzeinzug nicht gegeben. Eine Kombination von Absatzeinzug und Absatzabstand ist wegen der übertriebenen Redundanz abzulehnen. Der Einzug alleine ist deutlich genug. Der einzige Nachteil des Absatzeinzugs liegt

in der Verkürzung der Zeile. Damit gewinnt der Absatzabstand bei ohnehin kurzen Zeilen, etwa im Zeitungssatz, seine Berechtigung.

`parskip=`*`Methode`*

v3.00 Hin und wieder wird ein Layout mit Absatzabstand an Stelle des voreingestellten Absatzeinzugs gefordert. Die KOMA-Script-Klassen bietet mit der Option `parskip` eine Reihe von Möglichkeiten, um dies zu erreichen. Die *`Methode`* setzt sich dabei aus zwei Teilen zusammen. Der erste Teil ist entweder `full` oder `half`, wobei `full` für einen Absatzabstand von einer Zeile und `half` für einen Absatzabstand von einer halben Zeile steht. Der zweite Teil ist eines der Zeichen »`*`«, »`+`«, »`-`« und kann auch entfallen. Lässt man das Zeichen weg, so wird in der letzten Zeile des Absatzes am Ende mindestens ein Geviert, das ist 1 em, frei gelassen. Mit dem Pluszeichen wird am Zeilenende mindestens ein Drittel und mit dem Stern mindestens ein Viertel einer normalen Zeile frei gelassen. Mit der Minus-Variante werden keine Vorkehrungen für die letzte Zeile eines Absatzes getroffen.

v3.08 Die Einstellung kann jederzeit geändert werden. Wird sie innerhalb des Dokuments geändert, so wird implizit die Anweisung `\selectfont` ausgeführt. Änderungen der Absatzauszeichnung innerhalb eines Absatzes werden erst beim nächsten Absatz sichtbar.

Neben den sich so ergebenden acht Kombinationen ist es noch möglich, als *`Methode`* die Werte für einfache Schalter aus Tabelle 2.5, Seite 40 zu verwenden. Das Einschalten der Option entspricht dabei `full` ohne angehängtes Zeichen für den Freiraum der letzten Absatzzeile, also mit mindestens einem Geviert Freiraum am Ende des Absatzes. Das Ausschalten der Option schaltet hingegen wieder auf Absatzeinzug von einem Geviert um. Dabei darf die letzte Zeile eines Absatzes auch bis zum rechten Rand reichen. Einen Überblick über alle möglichen Werte für *`Methode`* bietet Tabelle 3.7.

Tabelle 3.7.: Mögliche Werte für Option `parskip` zur Auswahl der Kennzeichnung von Absätzen

`false`, `off`, `no`	Absätze werden durch einen Einzug der ersten Zeilen von einem Geviert (1 em) gekennzeichnet. Der erste Absatz eines Abschnitts wird nicht eingezogen.
`full`, `true`, `on`, `yes`	Absätze werden durch einen vertikalen Abstand von einer Zeile gekennzeichnet, Absatzenden durch einen Leerraum von mind. ein Geviert (1 em) der Grundschrift am Ende der letzten Zeile.
`full-`	Absätze werden durch einen vertikalen Abstand von einer Zeile gekennzeichnet. Absatzenden werden nicht gekennzeichnet.

...

Tabelle 3.7.: Mögliche Werte für Option `parskip` (*Fortsetzung*)

`full+`
: Absätze werden durch einen vertikalen Abstand von einer Zeile gekennzeichnet. Absatzenden werden durch einen Leerraum von mind. einem Drittel einer normalen Zeile gekennzeichnet.

`full*`
: Absätze werden durch einen vertikalen Abstand von einer Zeile gekennzeichnet. Absatzenden werden durch einen Leerraum von mind. einem Viertel einer normalen Zeile gekennzeichnet.

`half`
: Absätze werden durch einen vertikalen Abstand von einer halben Zeile gekennzeichnet. Absatzenden durch einen Leerraum von mind. ein Geviert (1 em) der normalen Schrift am Ende gekennzeichnet.

`half-`
: Absätze werden durch einen vertikalen Abstand von einer halben Zeile gekennzeichnet. Absatzenden werden nicht gekennzeichnet.

`half+`
: Absätze werden durch einen vertikalen Abstand von einer halben Zeile gekennzeichnet. Absatzenden werden durch einen Leerraum von mind. einem Drittel einer normalen Zeile gekennzeichnet.

`half*`
: Absätze werden durch einen vertikalen Abstand von einer Zeile gekennzeichnet. Absatzenden werden durch einen Leerraum von mind. einem Viertel einer normalen Zeile gekennzeichnet.

`never`
v3.08
: Es wird auch dann kein Abstand zwischen Absätzen eingefügt, wenn für den vertikalen Ausgleich der Einstellung `\flushbottom` zusätzlicher vertikaler Abstand verteilt werden muss.

Wird ein Absatzabstand verwendet, so verändert sich auch der Abstand vor, nach und innerhalb von Listenumgebungen. Dadurch wird verhindert, dass diese Umgebungen oder Absätze innerhalb dieser Umgebungen stärker vom Text abgesetzt werden als die Absätze des normalen Textes voneinander. Inhalts-, Abbildungs- und Tabellenverzeichnis werden immer ohne zusätzlichen Absatzabstand gesetzt.

Voreingestellt ist bei KOMA-Script `parskip=false`. Hierbei gibt es keinen Absatzabstand,

sondern einen Absatzeinzug von 1 em.

3.11. Erkennung von rechten und linken Seiten

Bei doppelseitigen Dokumenten wird zwischen linken und rechten Seiten unterschieden. Dabei hat eine linke Seite immer eine gerade Nummer und eine rechte Seite immer eine ungerade Nummer. Die Erkennung von rechten und linken Seiten ist damit gleichbedeutend mit der Erkennung von Seiten mit gerader oder ungerader Nummer. In dieser Anleitung ist vereinfachend von ungeraden und geraden Seiten die Rede.

Bei einseitigen Dokumenten existiert die Unterscheidung zwischen linken und rechten Seiten nicht. Dennoch gibt es natürlich auch bei einseitigen Dokumenten sowohl Seiten mit gerader als auch Seiten mit ungerader Nummer.

`\ifthispageodd{`*`Dann-Teil`*`}{`*`Sonst-Teil`*`}`

Will man bei KOMA-Script feststellen, ob ein Text auf einer geraden oder einer ungeraden Seite ausgegeben wird, so verwendet man die Anweisung `\ifthispageodd`. Dabei wird das Argument *`Dann-Teil`* nur dann ausgeführt, wenn man sich gerade auf einer ungeraden Seite befindet. Anderenfalls kommt das Argument *`Sonst-Teil`* zur Anwendung.

Beispiel: Angenommen, Sie wollen einfach nur ausgeben, ob ein Text auf einer geraden oder ungeraden Seite ausgegeben wird. Sie könnten dann beispielsweise mit der Eingabe

```
Dies ist eine Seite mit
\ifthispageodd{un}{}gerader Seitenzahl.
```

die Ausgabe

> Dies ist eine Seite mit gerader Seitenzahl.

erhalten. Beachten Sie, dass in diesem Beispiel das Argument *`Sonst-Teil`* leer geblieben ist.

Da die Anweisung `\ifthispageodd` mit einem Mechanismus arbeitet, der einem Label und einer Referenz darauf sehr ähnlich ist, werden nach jeder Textänderung mindestens zwei LaTeX-Durchläufe benötigt. Erst dann ist die Entscheidung korrekt. Im ersten Durchlauf wird für die Entscheidung eine Heuristik verwendet.

Näheres zur Problematik der Erkennung von linken und rechten Seiten oder geraden und ungeraden Seitennummern ist für Experten in Abschnitt 16.1, Seite 307 zu finden.

3.12. Kopf und Fuß bei vordefinierten Seitenstilen

Eine der allgemeinen Eigenschaften eines Dokuments ist der Seitenstil. Bei LaTeX versteht man unter dem Seitenstil in erster Linie den Inhalt der Kopf- und Fußzeilen.

`headsepline=`*`Ein-Aus-Wert`*
`footsepline=`*`Ein-Aus-Wert`*

v3.00

Mit diesen Optionen kann eingestellt werden, ob unter Kolumnentiteln oder über dem Fuß eine horizontale Linie gewünscht wird. Als *`Ein-Aus-Wert`* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Ein Aktivieren der Option `headsepline` oder die Verwendung der Option ohne Wertübergabe schaltet die Linie unter den Kolumnentiteln ein. Ein Aktivieren der Option `footsepline` oder die Verwendung der Option ohne Wertübergabe schaltet die Linie über der Fußzeile ein. Die Deaktivierung der Optionen schaltet die jeweilige Linie aus.

Bei den weiter unten erklärten Seitenstilen `empty` und `plain` hat die Option `headsepline` selbstverständlich keine Auswirkung, da hier auf einen Seitenkopf ausdrücklich verzichtet werden soll. Typografisch betrachtet hat eine solche Linie immer die Auswirkung, dass der Kopf optisch näher an den Text heranrückt. Dies bedeutet nun nicht, dass der Kopf räumlich weiter vom Textkörper weggerückt werden müsste. Stattdessen sollte der Kopf dann bei der Berechnung des Satzspiegels als zum Textkörper gehörend betrachtet werden. Dies wird bei KOMA-Script dadurch erreicht, dass bei Verwendung der Klassenoption `headsepline` automatisch die Paketoption `headinclude` mit gleichem Wert an das typearea-Paket weitergereicht wird. Entsprechendes gilt bei `footsepline` für `footinclude`. Im Gegensatz zu `headsepline` wirkt sich die Option `footsepline` auch beim Seitenstil `plain` aus, da `plain` eine Seitenzahl im Fuß ausgibt. Das Paket scrpage2 (siehe Kapitel 5) bietet weitere Einflussmöglichkeiten für Linien im Kopf und Fuß.

`\pagestyle{`*`Seitenstil`*`}`
`\thispagestyle{`*`lokaler Seitenstil`*`}`

Üblicherweise wird zwischen vier verschiedenen Seitenstilen unterschieden:

`empty` ist der Seitenstil, bei dem Kopf- und Fußzeile vollständig leer bleiben. Dies ist bei KOMA-Script vollkommen identisch zu den Standardklassen.

`headings` ist der Seitenstil für lebende Kolumnentitel. Das sind Kolumnentitel, bei denen Überschriften automatisch in den Seitenkopf übernommen werden. Im Internet oder in Beschreibungen zu LaTeX-Paketen findet man auch häufig die englische Bezeichnung »*running headline*«. Bei den Klassen scrbook und scrreprt werden dabei im doppelseitigen Layout die Überschriften der Kapitel und der Abschnitte in der Kopfzeile wiederholt – bei KOMA-Script jeweils außen, bei den Standardklassen innen. Die Seitenzahl wird bei KOMA-Script im Fuß außen, bei den Standardklassen im Kopf außen gesetzt. Im einseitigen Layout werden nur die Überschriften der Kapitel verwendet und bei KOMA-Script zentriert im Kopf ausgegeben. Die Seitenzahlen werden bei KOMA-Script dann zentriert im Fuß gesetzt. Bei scrartcl wird entsprechend verfahren, jedoch eine Ebene tiefer bei Abschnitt und Unterabschnitt angesetzt, da die Gliederungsebene Kapitel hier nicht existiert.

scrbook, scrreprt

scrartcl

Während die Standardklassen automatische Kolumnentitel immer in Versalien – also Großbuchstaben – setzen, verwendet KOMA-Script die Schreibweise, die in der Überschrift vorgefunden wurde. Dies hat verschiedene typografische Gründe. So sind Versalien als Auszeichnung eigentlich viel zu mächtig. Verwendet man sie trotzdem, sollten sie um einen Punkt kleiner gesetzt und leicht gesperrt werden (siehe hierzu beispielsweise [Tsc60]). All dies findet bei den Standardklassen keine Beachtung.

Darüber hinaus können bei den KOMA-Script-Klassen mit den Optionen `headsepline` und `footsepline` (siehe Seite 75) Linien unter dem Kopf und über dem Fuß gesetzt werden.

`myheadings` entspricht weitgehend dem Seitenstil `headings`, allerdings werden die Kolumnentitel nicht automatisch erzeugt, sondern liegen in der Verantwortung des Anwenders. Er verwendet dazu die Anweisungen `\markboth` und `\markright` (siehe Seite 77).

`plain` ist der Seitenstil, bei dem keinerlei Kolumnentitel verwendet, sondern nur eine Seitenzahl ausgegeben wird. Bei den Standardklassen wird diese Seitenzahl immer mittig im Fuß ausgegeben. Bei KOMA-Script erfolgt die Ausgabe im doppelseitigen Layout stattdessen außen im Fuß. Der einseitige Seitenstil entspricht bei KOMA-Script dem der Standardklassen.

Der Seitenstil kann jederzeit mit Hilfe der `\pagestyle`-Anweisung gesetzt werden. Verwendet man `\pagestyle` vor Anweisungen, die einen Seitenumbruch herbeiführen, und soll die Änderung erst ab der neuen Seite gelten, so ist ein `\cleardoublepage` davor nützlich. Üblicherweise verwendet man `\pagestyle` aber nur einmal zu Beginn des Dokuments oder in der Präambel.

Für eine Änderung des Seitenstils nur der aktuellen Seite verwendet man stattdessen die Anweisung `\thispagestyle`. Dies geschieht auch an einigen Stellen im Dokument automatisch. Beispielsweise wird bei allen Kapitelanfangsseiten implizit die Anweisung `\thispagestyle{\chapterpagestyle}` ausgeführt.

Bitte beachten Sie auch, dass die Umschaltung zwischen automatischen und manuellen Kolumnentiteln bei Verwendung des Pakets scrpage2 nicht mehr über den Seitenstil, sondern mit speziellen Anweisungen erfolgt. Die beiden Seitenstile `headings` und `myheadings` sollten nicht zusammen mit diesem Paket verwendet werden (siehe Kapitel 5, Seite 218).

v2.8p

Um die Schriftarten von Kopf und Fuß der Seite oder der Seitenzahl zu ändern, verwenden Sie die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56). Für den Kopf und den Fuß ist dabei das gleiche Element `pageheadfoot` zuständig. Das Element für die Seitenzahl innerhalb des Kopfes oder Fußes heißt `pagenumber`. Das ebenfalls in den KOMA-Script-Klassen bereitgestellte Element `pagefoot` wird nur verwendet, wenn man mit dem Paket scrpage2 (siehe Kapitel 5, Seite 220) einen Seitenstil definiert, bei dem auch der Fuß Text enthält.

Die Voreinstellungen sind in Tabelle 3.8 zu finden.

Tabelle 3.8.: Voreinstellungen der Schrift für die Elemente des Seitenstils

Element	Voreinstellung
`pagefoot`	
`pageheadfoot`	`\normalfont\normalcolor\slshape`
`pagenumber`	`\normalfont\normalcolor`

Beispiel: Angenommen, Sie wollen Kopf und Fuß einen Schriftgrad kleiner und kursiv setzen. Die Seitenzahl soll jedoch nicht kursiv, sondern fett gesetzt werden. Davon abgesehen, dass das Ergebnis grauenvoll aussehen wird, können Sie dies wie folgt erreichen:

```
\setkomafont{pageheadfoot}{%
  \normalfont\normalcolor\itshape\small
}
\setkomafont{pagenumber}{\normalfont\bfseries}
```

Wollen Sie hingegen lediglich, dass zusätzlich zur bereits voreingestellten schrägen Variante ebenfalls eine kleinere Schrift verwendet wird, so genügt:

```
\addtokomafont{pagehead}{\small}
```

Wie Sie sehen, wurde im letzten Beispiel das Element `pagehead` verwendet. Das gleiche Ergebnis erhalten Sie auch, wenn Sie stattdessen `pageheadfoot` verwenden (siehe Tabelle 3.2, Seite 57).

Es ist an dieser Stelle nicht möglich, Versalien für die automatischen Kolumnentitel zu erzwingen. Wenn Sie dies wünschen, verwenden Sie bitte das scrpage2-Paket (siehe Kapitel 5, Seite 227).

Werden eigene Seitenstile definiert, sind eventuell die Befehle `\usekomafont{pageheadfoot}`, `\usekomafont{pagenumber}` sowie `\usekomafont{pagefoot}` nützlich. Insbesondere falls Sie dafür nicht das KOMA-Script-Paket scrpage2 (siehe Kapitel 5), sondern beispielsweise das Paket fancyhdr (siehe [vO00]) einsetzen, können Sie diese Befehle in Ihren Definitionen verwenden. Dadurch bleiben Sie zu KOMA-Script möglichst kompatibel. Verwenden Sie diese Befehle in Ihren eigenen Definitionen nicht, so bleiben Schriftänderungen wie in den vorangehenden Beispielen unbeachtet. Das Paket scrpage2 sorgt selbst für maximale Kompatibilität.

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

Beim Seitenstil `myheadings` wird der Kolumnentitel nicht automatisch gesetzt. Stattdessen setzt man ihn mit Hilfe der Anweisungen `\markboth` und `\markright`. Dabei wird die *`linke Marke`* normalerweise im Kopf linker Seiten und die *`rechte Marke`* im Kopf rechter Seiten verwendet. Im einseitigen Satz existiert nur die rechte Marke. Bei Verwendung des Paketes scrpage2 steht außerdem die Anweisung `\markleft` zur Verfügung.

Tabelle 3.9.: Makros zur Festlegung des Seitenstils besonderer Seiten

`\titlepagestyle`	Seitenstil der Seite mit der Titelei bei Titelköpfen (siehe Abschnitt 3.7)
`\partpagestyle`	Seitenstil der Seiten mit `\part`-Titeln (siehe Abschnitt 3.16)
`\chapterpagestyle`	Seitenstil auf Kapitelanfangsseiten (siehe Abschnitt 3.16)
`\indexpagestyle`	Seitenstil der ersten Stichwortverzeichnisseite (siehe Abschnitt 3.24)

Die Anweisungen können auch zusammen mit anderen Seitenstilen verwendet werden. Bei Kombination mit automatischen Kolumnentiteln, etwa dem Seitenstil `headings`, ist der Wirkungsbereich allerdings bis zum nächsten automatischen Setzen der entsprechenden Marke begrenzt.

```
\titlepagestyle
\partpagestyle
\chapterpagestyle
\indexpagestyle
```

Auf einigen Seiten wird mit Hilfe von `\thispagestyle` automatisch ein anderer Seitenstil gewählt. Welcher Seitenstil dies ist, wird diesen vier Makros entnommen, wobei `\partpagestyle` und `\chapterpagestyle` nur bei den Klassen scrbook und scrreprt nicht jedoch bei scrartcl existieren. In der Voreinstellung ist der Seitenstil in allen vier Fällen `plain`. Die Bedeutung der einzelnen Makros entnehmen Sie bitte Tabelle 3.9. Die Seitenstile können mit Hilfe von `\renewcommand` umdefiniert werden.

scrbook, scrreprt

Beispiel: Angenommen, Sie wollen nicht, dass die Seiten mit der `\part`-Überschrift mit einer Nummer versehen werden. Dann setzen Sie folgende Anweisung beispielsweise in der Präambel Ihres Dokuments:

```
\renewcommand*{\partpagestyle}{empty}
```

Wie Sie auf Seite 75 erfahren haben, ist der Seitenstil `empty` genau das, was in diesem Beispiel verlangt wird. Natürlich können Sie auch einen selbst definierten Seitenstil verwenden.

Angenommen, Sie haben mit dem Paket scrpage2 (siehe Kapitel 5) einen eigenen Seitenstil für Kapitelanfangsseiten definiert. Diesem Seitenstil haben Sie den passenden Namen `chapter` gegeben. Um diesen nun auch tatsächlich zu verwenden, definieren Sie `\chapterpagestyle` entsprechend um:

```
\renewcommand*{\chapterpagestyle}{chapter}
```

Angenommen, Sie wollen das Inhaltsverzeichnis eines Buches insgesamt nicht mit Seitenzahlen versehen. Danach soll aber wieder mit dem Seitenstil `headings` gearbeitet werden, sowie mit `plain` auf den Kapitelanfangsseiten. Dann verwenden Sie beispielsweise:

```
\clearpage
\pagestyle{empty}
\renewcommand*{\chapterpagestyle}{empty}
\tableofcontents
\clearpage
\pagestyle{headings}
\renewcommand*{\chapterpagestyle}{plain}
```

Sie können die Umdefinierung des Seitenstils für Kapitelanfangsseiten aber auch lokal halten. Das hat den Vorteil, dass Sie dann keine Annahmen über die vor der Änderung gültige Einstellung treffen müssen. Die Änderung des Seitenstils selbst können Sie gleichermaßen lokal halten:

```
\clearpage
\begingroup
  \pagestyle{empty}
  \renewcommand*{\chapterpagestyle}{empty}
  \tableofcontents
  \clearpage
\endgroup
```

Beachten Sie jedoch, dass Sie niemals eine nummerierte Gliederungsüberschrift in eine Gruppe packen sollten. Anderenfalls können Anweisungen wie `\label` rasch zu unvorhergesehenen Ergebnissen führen.

Wer nun glaubt, er könne auf Kapitelanfangsseiten ebenfalls mit lebenden Kolumnentiteln arbeiten, indem er einfach eine Definition wie

```
\renewcommand*{\chapterpagestyle}{headings}
```

verwendet, sollte in Abschnitt 16.1, Seite 307 Näheres über die Hintergründe zu `\rightmark` nachlesen.

`\pagenumbering{`*Nummerierungsstil*`}`

Diese Anweisung funktioniert bei KOMA-Script in der gleichen Weise wie bei den Standardklassen. Genau genommen handelt es sich dabei weder um eine Fähigkeit der Standardklassen noch der KOMA-Script-Klassen, sondern um eine Anweisung des LaTeX-Kerns. Sie wird verwendet, um den *Nummerierungsstil* für die Seitenzahlen umzuschalten.

Die Umschaltung gilt ab sofort, also ab der Seite, auf der diese Anweisung aufgerufen wird. Gegebenenfalls sollte also zuvor mit `\clearpage` oder `\cleardoublepage` diese Seite erst beendet werden. Mögliche Angaben für den *Nummerierungsstil* sind Tabelle 3.10 zu entnehmen.

Tabelle 3.10.: Verfügbare Nummerierungsstile für Seitenzahlen

Nummerierungsstil	Beispiel	Bedeutung
`arabic`	8	arabische Zahlen
`roman`	viii	kleine römische Zahlen
`Roman`	VIII	große römische Zahlen
`alph`	h	Kleinbuchstaben
`Alph`	H	Großbuchstaben

Der Aufruf von `\pagenumbering` setzt immer die Seitenzahl zurück. Die aktuelle Seite bekommt also die Nummer 1 im gewählten *Nummerierungsstil*.

3.13. Vakatseiten

Vakatseiten sind Seiten, die beim Satz eines Dokuments absichtlich leer bleiben. Ursprünglich sind diese Seiten wirklich komplett leer. Bei LaTeX werden sie jedoch in der Voreinstellung mit dem aktuell gültigen Seitenstil gesetzt. KOMA-Script bietet hier diverse Erweiterungen.

Vakatseiten findet man hauptsächlich in Büchern. Da es bei Büchern üblich ist, dass Kapitel auf einer rechten Seite beginnen, muss in dem Fall, dass das vorherigen Kapitel ebenfalls auf einer rechten Seite endet, eine leere linke Seite eingefügt werden. Aus dieser Erklärung ergibt sich auch, dass Vakatseiten normalerweise nur im doppelseitigen Satz existieren. Die leeren Rückseiten im einseitigen Druck werden eher nicht als Vakatseiten bezeichnet, obwohl sie auf Druckbögen im Ergebnis als solche erscheinen.

`cleardoublepage=`*`Seitenstil`*
`cleardoublepage=current`

v3.00 Mit Hilfe dieser Option kann man den *Seitenstil* der Vakatseite bestimmen, die bei Bedarf von der Anweisung `\cleardoublepage` eingefügt wird, um bis zur gewünschten Seite zu umbrechen. Als *Seitenstil* sind dabei alle bereits definierten Seitenstile (siehe Abschnitt 3.12 ab Seite 74 und Kapitel 5 ab Seite 214) verwendbar. Daneben ist auch `cleardoublepage=current` möglich. Dieser Fall entspricht der Voreinstellung von KOMA-Script bis Version 2.98c und führt dazu, dass die Vakatseite mit dem Seitenstil erzeugt wird, der beim Einfügen gerade aktuell ist. Ab Version 3.00 werden in der Voreinstellung entsprechend der typografischen Gepflogenheiten Vakatseiten mit dem Seitenstil `empty` erzeugt, wenn man nicht Kompatibilität zu früheren KOMA-Script-Versionen eingestellt hat (siehe Option `version`, Abschnitt 3.2, Seite 31).

v3.00

Beispiel: Angenommen Sie wollen, dass die Vakatseiten bis auf die Paginierung leer sind, also mit Seitenstil `plain` erzeugt werden. Dies erreichen Sie beispielsweise mit

```
\KOMAoption{cleardoublepage=plain}
```

Näheres zum Seitenstil `plain` ist in Abschnitt 3.12, Seite 76 zu finden.

```
\clearpage
\cleardoublepage
\cleardoublepageusingstyle{Seitenstil}
\cleardoubleemptypage
\cleardoubleplainpage
\cleardoublestandardpage
\cleardoubleoddusingstyle{Seitenstil}
\cleardoubleoddemptypage
\cleardoubleoddplainpage
\cleardoubleoddstandardpage
\cleardoubleevenusingstyle{Seitenstil}
\cleardoubleevenemptypage
\cleardoubleevenplainpage
\cleardoubleevenstandardpage
```

Im LaTeX-Kern existiert die Anweisung `\clearpage`, die dafür sorgt, dass alle noch nicht ausgegebenen Gleitumgebungen ausgegeben werden und anschließend eine neue Seite begonnen wird. Außerdem existiert die Anweisung `\cleardoublepage`, die wie `\clearpage` arbeitet, durch die aber im doppelseitigen Layout (siehe Option `twoside` in Abschnitt 2.4, Seite 40) eine neue rechte Seite begonnen wird. Dazu wird gegebenenfalls eine linke Vakatseite im aktuellen Seitenstil ausgegeben.

v3.00 Bei KOMA-Script arbeitet `\cleardoubleoddstandardpage` genau in der soeben für die Standardklassen beschriebenen Art und Weise. Die Anweisung `\cleardoubleoddplainpage` ändert demgegenüber den Seitenstil der leeren linken Seite zusätzlich auf `plain`, um den Kolumnentitel zu unterdrücken. Analog dazu wird bei der Anweisung `\cleardoubleoddemptypage` der Seitenstil `empty` verwendet, um sowohl Kolumnentitel als auch Seitenzahl auf der leeren linken Seite zu unterdrücken. Die Seite ist damit vollständig leer. Will man für die Vakatseite einen eigenen *Seitenstil* vorgeben, so ist dieser als Argument von `\cleardoubleoddusingstyle` anzugeben. Dabei kann jeder bereits definierte Seitenstil (siehe auch Kapitel 5) verwendet werden.

Manchmal möchte man nicht, dass Kapitel mit neuen rechten Seiten, sondern links auf einer Doppelseite beginnen. Dies widerspricht zwar dem klassischen Buchdruck, kann jedoch seine Berechtigung haben, wenn die Doppelseite am Kapitelanfang einen ganz speziellen Inhalt hat. Bei KOMA-Script ist deshalb die Anweisung `\cleardoubleevenstandardpage` als Äquivalent zur Anweisung `\cleardoubleoddstandardpage` definiert, jedoch mit dem Unterschied, dass die nächste Seite eine linke Seite ist. Entsprechendes gilt für die Anweisungen `\cleardoubleevenplainpage`, `\cleardoubleevenemptypage` und `\cleardoubleevenusingstyle`.

Die Arbeitsweise der Anweisungen `\cleardoublestandardpage`, `\cleardoubleemptypage`, `\cleardoubleplainpage` und der ein Argument erwartenden Anweisung `\cleardoublepageusingstyle` ist ebenso wie die Standard-Anweisung `\cleardoublepage` von der zuvor erklärten Option `cleardoublepage` abhängig und entspricht je nach Einstellung einer der in den vorherigen Absätzen erläuterten Anweisungen.

Beispiel: Angenommen Sie wollen innerhalb eines Dokuments als nächstes eine Doppelseite setzen, bei der auf der linken Seite eine Abbildung in Größe des Satzspiegels platziert wird und rechts ein neues Kapitel beginnt. Falls das vorherige Kapitel mit einer linken Seite endet, muss also eine Vakatseite eingefügt werden. Diese soll komplett leer sein. Ebenso soll die linke Bildseite weder Kopf noch Fußzeile besitzen. Zunächst wird mit

```
\KOMAoptions{cleardoublepage=empty}
```

dafür gesorgt, dass Vakatseiten mit dem Seitenstil `empty`, also ohne Kopf- und Fußzeile gesetzt werden. Diese Einstellung können Sie bereits in der Dokumentpräambel vornehmen. Die Optionen können alternativ auch als optionale Argumente von `\documentclass` angegeben werden.

An der gewünschten Stelle im Dokument schreiben Sie nun:

```
\cleardoubleevenemptypage
\thispagestyle{empty}
\includegraphics[width=\textwidth,%
                 height=\textheight,%
                 keepaspectratio]%
                {bild}
\chapter{Kapitelüberschrift}
```

Die erste Zeile wechselt auf die nächste linke Seite und fügt zu diesem Zweck bei Bedarf eine komplett leere rechte Seite ein. Die zweite Zeile sorgt dafür, dass diese linke Seite ebenfalls mit dem Seitenstil `empty` gesetzt wird. Die dritte bis sechste Zeile lädt die Bilddatei mit dem Namen `bild` und bringt sie auf die gewünschte Größe, ohne sie dabei zu verzerren. Hierfür wird das Paket graphicx benötigt (siehe [Car99d]). Die letzte Zeile beginnt auf der nächsten – dann rechten – Seite ein neues Kapitel.

3.14. Fußnoten

Im Unterschied zu den Standardklassen bietet KOMA-Script die Möglichkeit, die Form von Fußnoten zu konfigurieren.

`footnotes=`*Einstellung*

v3.00

Fußnoten werden im Text in der Voreinstellung mit kleinen, hochgestellten Ziffern markiert. Werden zu einer Textstelle mehrere Fußnoten hintereinander gesetzt, so entsteht der Eindruck, dass es sich nicht um zwei einzelne Fußnoten, sondern um eine einzige Fußnote mit hoher Nummer handelt. Mit `footnotes=multiple` werden Fußnoten, die unmittelbar aufeinander folgen,

Tabelle 3.11.: Mögliche Werte für Option `footnotes` zur Einstellung der Fußnoten

`multiple`	Unmittelbar aufeinander folgende Fußnotenmarkierungen werden durch `\multfootsep` voneinander getrennt ausgegeben.
`nomultiple`	Unmittelbar aufeinander folgende Fußnotenmarkierungen werden auch unmittelbar aufeinander folgend ausgegeben.

stattdessen mit Trennzeichen aneinander gereiht. Das in `\multfootsep` definierte Trennzeichen ist mit einem Komma vorbelegt. Der gesamte Mechanismus ist kompatibel zu footmisc, Version 5.3d (siehe [Fai05]) implementiert. Er wirkt sich sowohl auf Fußnotenmarkierungen aus, die mit `\footnote` gesetzt wurden, als auch auf solche, die direkt mit `\footnotemark` ausgegeben werden.

Es ist jederzeit möglich, mit `\KOMAoptions` oder `\KOMAoption` auf die Voreinstellung `footnotes=nomultiple` zurückzuschalten. Bei Problemen mit anderen Paketen, die Einfluss auf die Fußnoten nehmen, sollte die Option jedoch nicht verwendet und die Einstellung auch nicht innerhalb des Dokuments umgeschaltet werden.

Eine Zusammenfassung möglicher Werte für die *Einstellung* von `footnotes` bietet Tabelle 3.11, Seite 83.

`\footnote[`*Nummer*`]{`*Text*`}`
`\footnotemark[`*Nummer*`]`
`\footnotetext[`*Nummer*`]{`*Text*`}`
`\multiplefootnoteseparator`
`\multfootsep`

Fußnoten werden bei KOMA-Script genau wie bei den Standardklassen mit der Anweisung `\footnote` oder den paarweise zu verwendenden Anweisungen `\footnotemark` und `\footnotetext` erzeugt. Genau wie bei den Standardklassen ist es möglich, dass innerhalb einer Fußnote ein Seitenumbruch erfolgt. Dies geschieht in der Regel dann, wenn die zugehörige Fußnotenmarkierung so weit unten auf der Seite gesetzt wird, dass keine andere Wahl bleibt, als die Fußnote auf die nächste Seite zu umbrechen. Im Unterschied zu den Standardklassen bietet KOMA-Script aber zusätzlich die Möglichkeit, Fußnoten, die unmittelbar aufeinander folgen, automatisch zu erkennen und durch ein Trennzeichen auseinander zu rücken. Siehe hierzu die zuvor dokumentierte Option `footnotes`.

v3.00

Will man dieses Trennzeichen stattdessen von Hand setzen, so erhält man es durch Aufruf von `\multiplefootnoteseparator`. Diese Anweisung sollten Anwender jedoch nicht umdefinieren, da sie neben dem Trennzeichen auch die Formatierung des Trennzeichen, beispielsweise die Wahl der Schriftgröße und das Hochstellen, enthält. Das Trennzeichen selbst ist in der Anweisung `\multfootsep` gespeichert. In der Voreinstellung ist dieses als

```
\newcommand*{\multfootsep}{,}
```

definiert. Dieses kann umdefiniert werden.

Beispiel: Angenommen, Sie wollen zu einem Wort zwei Fußnoten setzen. Im ersten Ansatz schreiben Sie dafür

```
Wort\footnote{Fußnote 1}\footnote{Fußnote 2}.
```

Nehmen wir weiter an, dass die Fußnoten mit 1 und 2 nummeriert werden. Da die beiden Fußnotennummern direkt aufeinander folgen, entsteht jedoch der Eindruck, dass das Wort nur eine Fußnote mit der Nummer 12 besitzt. Sie könnten dies nun dadurch ändern, dass Sie mit

```
\KOMAoptions{footnotes=multiple}
```

die automatische Erkennung von Fußnotenhäufungen aktivieren. Stattdessen können Sie aber auch

```
Wort\footnote{Fußnote 1}%
\multiplefootnoteseparator
\footnote{Fußnote 2}
```

verwenden. Das sollte auch dann noch funktionieren, wenn die automatische Erkennung aus irgendwelchen Gründen versagt oder nicht verwendet werden kann.

Nehmen wir nun an, dass Sie außerdem wollen, dass die Fußnotennummern nicht nur durch ein Komma, sondern durch ein Komma gefolgt von einem Leerzeichen getrennt werden sollen. In diesem Fall schreiben Sie

```
\renewcommand*{\multfootsep}{,\nobreakspace}
```

in Ihre Dokumentpräambel. `\nobreakspace` wurde hier an Stelle eines normalen Leerzeichens gewählt, damit innerhalb der Reihung der Fußnotenzeichen kein Absatz- oder Seitenumbruch erfolgen kann.

`\footref{`*Referenz*`}`

v3.00

Manchmal hat man in einem Dokument eine Fußnote, zu der es im Text mehrere Verweise geben soll. Die ungünstige Lösung dafür wäre die Verwendung von `\footnotemark` unter Angabe der gewünschten Nummer. Ungünstig an dieser Lösung ist, dass man die Nummer kennen muss und sich diese jederzeit ändern kann. KOMA-Script bietet deshalb die Möglichkeit, den `\label`-Mechanismus auch für Verweise auf Fußnoten zu verwenden. Man setzt dabei in der entsprechenden Fußnote eine `\label`-Anweisung und kann dann mit `\footref` alle weiteren Fußnotenmarken für diese Fußnote im Text setzen.

Beispiel: Sie schreiben einen Text, in dem sie bei jedem Auftreten eines Markennamens eine Fußnote setzen müssen, die darauf hinweist, dass es sich um einen geschützten Markennamen handelt. Sie schreiben beispielsweise:

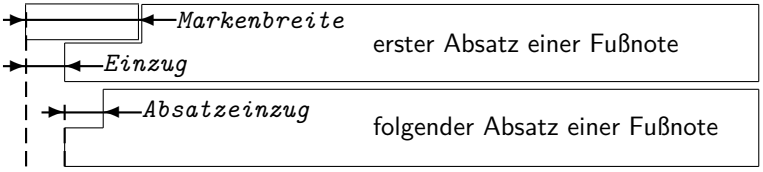


Abbildung 3.1.: Parameter für die Darstellung der Fußnoten

```
Die Firma SplischSplasch\footnote{Bei diesem
  Namen handelt es sich um eine registrierte
  Marke. Alle Rechte daran sind dem
  Markeninhaber vorbehalten.\label{refnote}}
stellt neben SplischPlumps\footref{refnote}
auch noch die verbesserte Version
SplischPlatsch\footref{refnote} her.
```

Es wird dann dreimal eine Marke auf dieselbe Fußnote gesetzt, einmal mit `\footnote` direkt und zweimal mit `\footref`.

Da die Fußnotenmarken mit Hilfe des `\label`-Mechanismus gesetzt werden, werden nach Änderungen, die sich auf die Fußnotennummerierung auswirken, gegebenenfalls zwei LaTeX-Durchläufe benötigt, bis die mit `\footref` gesetzten Marken korrekt sind.

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

Die KOMA-Script-Klassen setzen Fußnoten etwas anders als die Standardklassen. Die Fußnotenmarkierung im Text, also die Referenzierung der Fußnote, erfolgt wie bei den Standardklassen durch kleine hochgestellte Zahlen. Genauso werden die Markierungen auch in der Fußnote selbst wiedergegeben. Sie werden dabei rechtsbündig in einem Feld der Breite *Markenbreite* gesetzt. Die erste Zeile der Fußnote schließt direkt an das Feld der Markierung an.

Alle weiteren Zeilen werden um den Betrag von *Einzug* eingezogen ausgegeben. Wird der optionale Parameter *Markenbreite* nicht angegeben, dann entspricht er dem Wert von *Einzug*. Sollte die Fußnote aus mehreren Absätzen bestehen, dann wird die erste Zeile eines Absatzes zusätzlich mit dem Einzug der Größe *Absatzeinzug* versehen.

Abbildung 3.1 veranschaulicht die verschiedenen Parameter nochmals. Die Voreinstellung in den KOMA-Script-Klassen entspricht folgender Definition:

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

Dabei wird mit Hilfe von `\textsuperscript` sowohl die Hochstellung als auch die Wahl einer kleineren Schrift erreicht. Die Anweisung `\thefootnotemark` liefert die aktuelle Fußnotenmarke ohne jegliche Formatierung.

v2.8q Auf die Fußnote einschließlich der Markierung findet außerdem die für das Element `footnote` eingestellte Schriftart Anwendung. Die davon abweichende Schriftart der Markierung kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) für das Element `footnotelabel` eingestellt werden. Siehe hierzu auch Tabelle 3.2, Seite 57. Voreingestellt ist jeweils keine Umschaltung der Schrift.

Die Fußnotenmarkierung im Text wird getrennt von der Markierung vor der Fußnote definiert. Dies geschieht mit der Anweisung `\deffootnotemark`. Voreingestellt ist hier:

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

v2.8q Dabei findet die Schriftart für das Element `footnotereference` Anwendung (siehe Tabelle 3.2, Seite 57). Die Markierungen im Text und in der Fußnote selbst sind also identisch. Die Schriftart kann mit den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) jedoch geändert werden.

Beispiel: Relativ häufig wird gewünscht, dass die Markierung in der Fußnote selbst weder hochgestellt noch kleiner gesetzt wird. Dabei soll sie aber nicht direkt am Text kleben, sondern geringfügig davor stehen. Dies kann zum einen wie folgt erreicht werden:

```
\deffootnote{1em}{1em}{\thefootnotemark\ }
```

Die Fußnotenmarkierung und das folgende Leerzeichen wird also rechtsbündig in eine Box der Breite 1 em gesetzt. Die folgenden Zeilen der Fußnote werden gegenüber dem linken Rand ebenfalls um 1 em eingezogen.

Eine weitere, oft gefragte Formatierung sind linksbündige Fußnotenmarkierungen in der Fußnote. Diese können mit folgender Definition erhalten werden:

```
\deffootnote{1.5em}{1em}{%
    \makebox[1.5em][l]{\thefootnotemark}}
```

Sollen jedoch die Fußnoten insgesamt lediglich in einer anderen Schriftart, beispielsweise serifenlos gesetzt werden, so ist dies ganz einfach mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) zu lösen:

```
\setkomafont{footnote}{\sffamily}
```

Wie die Beispiele zeigen, ermöglicht KOMA-Script mit dieser einfachen Benutzerschnittstelle eine große Vielfalt unterschiedlicher Fußnotenformatierungen.

`\setfootnoterule[`*Höhe*`]{`*Länge*`}`

v3.06 Üblicherweise wird zwischen dem Textbereich und dem Fußnotenapparat eine Trennlinie gesetzt, die jedoch normalerweise nicht über die gesamte Breite des Satzspiegels geht. Mit Hilfe dieser Anweisung kann die genaue Länge und die Höhe oder Dicke der Linie bestimmt werden. Dabei werden *Höhe* und *Länge* erst beim Setzen der Linie selbst abhängig von `\normalsize`

ausgewertet. Der optionale Parameter *Höhe* kann komplett entfallen und wird dann nicht geändert. Ist das Argument *Höhe* oder *Länge* leer, so wird die jeweilige Größe ebenfalls nicht geändert. Es gibt sowohl beim Setzen als auch bei Verwendung der Größen für unplausible Werte eine Warnung.

v3.07 Die Farbe der Linie kann über das Element `footnoterule` mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) eingestellt werden. Voreingestellt ist hierbei keinerlei Änderung von Schrift oder Farbe. Um die Farbe ändern zu können, muss außerdem ein Farbpaket wie xcolor geladen sein.

scrbook

3.15. Abgrenzung

Bei Büchern gibt es teilweise die Grobaufteilung in *Vorspann*, *Hauptteil* und *Nachspann*. Auch KOMA-Script bietet für scrbook diese Möglichkeit.

```
\frontmatter
\mainmatter
\backmatter
```

Mit `\frontmatter` wird der Vorspann eingeleitet. Im Vorspann werden die nummerierten Seiten mit römischen Seitenzahlen versehen. Kapitelüberschriften sind im Vorspann nicht nummeriert. Abschnittsüberschriften wären jedoch nummeriert, gingen von Kapitelnummer 0 aus und wären außerdem über Kapitelgrenzen hinweg durchgehend nummeriert. Dies spielt jedoch keine Rolle, da der Vorspann allenfalls für die Titelei, das Inhalts-, Abbildungs- und Tabellenverzeichnis und ein Vorwort verwendet wird. Das Vorwort kann also als normales Kapitel gesetzt werden. Ein Vorwort sollte niemals in Abschnitte unterteilt, sondern möglichst kurz gefasst werden. Im Vorwort wird also keine tiefere Gliederungsebene als Kapitel benötigt.

v2.97e Für den Fall, dass der Anwender dies anders sieht und nummerierte Abschnitte in den Kapiteln des Vorspanns haben will, enthält ab Version 2.97e die Nummerierung der Abschnitte keine Kapitelnummer. Diese Änderung gibt es nur, wenn eine Kompatibilität ab Version 2.97e eingestellt ist (siehe Option `version`, Abschnitt 3.2, Seite 31). Es wird ausdrücklich darauf hingewiesen, dass dadurch bezüglich der Nummern eine Verwechslung mit Kapitelnummern gegeben ist! Die Verwendung von `\addsec` und `\section*` (siehe Abschnitt 3.16, Seite 98 und Seite 99) sind aus Sicht des Autors im Vorspann deshalb unbedingt vorzuziehen!

v2.97e Ab Version 2.97e enthalten auch die Nummern für Gleitumgebungen wie Tabellen und Abbildungen und die Gleichungsnummern im Vorspann keinen Kapitelanteil. Auch dies erfordert eine entsprechende Kompatibilitätseinstellung (siehe Option `version`, Abschnitt 3.2, Seite 31).

Mit `\mainmatter` wird der Hauptteil eingeleitet. Existiert kein Vorspann, so kann diese Anweisung auch entfallen. Im Hauptteil sind arabische Seitenzahlen voreingestellt. Die Seitenzählung beginnt im Hauptteil neu mit der 1.

Mit `\backmatter` wird der Nachspann eingeleitet. Was zum Nachspann gehört, ist unterschiedlich. Manchmal wird im Nachspann nur das Literaturverzeichnis, manchmal nur das

Tabelle 3.12.: Mögliche Werte für Option `open` zur Auswahl von Umbrüchen mit Vakatseiten

`any`	Die Anweisungen `\cleardoublepageusingstyle`, `\cleardoublestandardpage`, `\cleardoubleplainpage`, `\cleardoubleemptypage` und `\cleardoublepage` erzeugen einen Seitenumbruch und entsprechen damit `\clearpage`.
`left`	Die Anweisungen `\cleardoublepageusingstyle`, `\cleardoublestandardpage`, `\cleardoubleplainpage`, `\cleardoubleemptypage` und `\cleardoublepage` erzeugen einen Seitenumbruch und fügen ggf. eine Vakatseite ein, um im doppelseitigen Satz auf die nächste linke Seite zu gelangen.
`right`	Die Anweisungen `\cleardoublepageusingstyle`, `\cleardoublestandardpage`, `\cleardoubleplainpage`, `\cleardoubleemptypage` und `\cleardoublepage` erzeugen einen Seitenumbruch und fügen ggf. eine Vakatseite ein, um im doppelseitigen Satz auf die nächste rechte Seite zu gelangen.

Stichwortverzeichnis gesetzt. Manchmal erscheint der gesamte Anhang im Nachspann. Der Nachspann gleicht bezüglich der Gliederungsüberschriften dem Vorspann. Eine getrennte Seitennummerierung ist jedoch nicht vorgesehen. Falls Sie dies ebenfalls benötigen, bedienen Sie sich bitte der Anweisung `\pagenumbering` aus Abschnitt 3.12, Seite 79.

3.16. Gliederung

Unter der Gliederung versteht man die Einteilung eines Dokuments in Teile, Kapitel, Abschnitte und weitere Gliederungsebenen.

`open=`*`Methode`*

scrbook, scrreprt

Bei den KOMA-Script-Klassen scrbook und scrreprt kann gewählt werden, wo im doppelseitigen Satz neue Kapitel beginnen. In der Voreinstellung beginnen bei scrreprt neue Kapitel auf der nächsten neuen Seite. Dies entspricht der *`Methode`* `any`. Demgegenüber beginnen bei scrbook neue Kapitel auf der nächsten rechten Seite. Dies entspricht der *`Methode`* `right` und ist bei den meisten Büchern üblich. In einigen Fällen sollen neue Kapitel jedoch auf der linken Seite einer kompletten Doppelseite beginnen. Dies entspricht der *`Methode`* `left`. Eine Zusammenfassung der möglichen Werte findet sich noch einmal in Tabelle 3.12.

v3.00

Neben den Kapitelanfängen wirkt sich die Option auch auf die explizite Verwendung der Anweisung`\cleardoublepage` ebenso wie auf `\cleardoublepageusingstyle`, `\cleardoublestandardpage`, `\cleardoubleplainpage` und `\cleardoubleemptypage` aus.

Siehe dazu Abschnitt 3.12, Seite 81. Da im einseitigen Satz nicht zwischen linken und rechten Seiten unterschieden wird, hat die Option dort keine Wirkung.

Bei der Klasse scrartcl ist die oberste Gliederungsebene unter dem Teil der Abschnitt. Daher unterstützt scrartcl diese Option nicht.

`chapterprefix=`*`Ein-Aus-Wert`*
`appendixprefix=`*`Ein-Aus-Wert`*

scrbook, scrreprt

Bei den Standardklassen book und report werden Kapitelüberschriften in der Form ausgegeben, dass zunächst in einer Zeile »Kapitel«[1] gefolgt von der Kapitelnummer steht. Erst ab der nächsten Zeile wird dann die Überschrift in linksbündigem Flattersatz ausgegeben. Bei KOMA-Script kann dieses Verhalten mit der Klassenoption `chapterprefix` ebenfalls erreicht werden. Als *`Ein-Aus-Wert`* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Voreingestellt ist `chapterprefix=false`, während das Verhalten der Standardklassen `chapterprefix=true` entspricht. Die Optionen wirken sich außerdem auf das Aussehen der automatischen Kolumnentitel für Kapitel aus (siehe Abschnitt 3.12, Seite 75).

Zuweilen kommt es vor, dass man die Kapitelüberschriften im Hauptteil durchaus in der einfachen Form von `chapterprefix=false` setzen möchte. Gleichzeitig sollen die Überschriften im Anhang jedoch davon abweichend mit einer Präfixzeile, »Anhang« gefolgt vom Buchstaben des Anhangs, versehen werden. Dies ist mit der Einstellung `appendixprefix=true` möglich. Da sich jedoch dadurch ein inkonsistentes Layout ergibt, rate ich von der Verwendung ab.

v2.96a

Die Schriftart der Kapitelnummernzeile bei `chapterprefix=true` oder `appendixprefix=true` kann mit den beiden Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) für das Element `chapterprefix` geändert werden. Voreingestellt ist die Verwendung des Elements `chapter` (siehe Seite 93, sowie Tabelle 3.15, Seite 97)

`headings=`*`Einstellung`*

v3.00

Die Überschriften werden sowohl bei den Standardklassen als auch bei KOMA-Script normalerweise recht groß gesetzt. Dies gefällt nicht jedem und wirkt insbesondere bei kleinen Papiergrößen oft störend. Daher stehen bei KOMA-Script neben den mit der Option `headings=big` sehr groß voreingestellten Überschriften die beiden Möglichkeiten `headings=normal` und `headings=small` zur Verfügung, mit denen man insgesamt kleinere Überschriften erhält. Die aus den Optionen resultierenden Schriftgrößen sind für die Überschriften der Klassen scrbook und scrreprt Tabelle 3.15, Seite 97 zu entnehmen. Bei scrartcl werden generell etwas kleinere Überschriften verwendet. Die Abstände vor und nach Kapitelüberschriften werden von diesen Optionen ebenfalls beeinflusst.

scrbook, scrreprt

[1] Bei Verwendung einer anderen Sprache als Deutsch wird »Kapitel« selbstverständlich in der jeweiligen Sprache gesetzt.

Auf Kapitelüberschriften wirken sich außerdem die Optionen `headings=twolinechapter` und `headings=onelinechapter`, was `chapterprefix=true` und `chapterprefix=false` entspricht (siehe oben) aus. Für den Anhang stehen `headings=twolineappendix` und `headings=onelineappendix` zur Verfügung, die den Optionen `appendixprefix=true` und `appendixprefix=false` entspricht (siehe ebenfalls oben).

scrbook, scrreprt

Für Kapitel kann mit `headings=openany`, `headings=openright` und `headings=openleft` die Methode für Kapitelanfänge alternativ zur Verwendung der Option `open` mit den Werten `any`, `right` und `left` (siehe oben) gesetzt werden.

v3.10

Eine weitere Besonderheit von KOMA-Script betrifft die Behandlung des optionalen Arguments der Gliederungsbefehle. Sowohl dessen Funktion als auch dessen Bedeutung kann durch die Einstellungen `headings=optiontohead`, `headings=optiontotoc` und `headings=optiontoheadandtoc` beeinflusst werden.

Eine Zusammenfassung der möglichen Einstellungen für Option `headings` finden Sie in Tabelle 3.13. Beispiele zu Verwendung einiger der möglichen Einstellungen sind in den nachfolgenden Beschreibungen der Gliederungsbefehle enthalten.

Tabelle 3.13.: Mögliche Werte für Option `headings` zur Einstellung der Überschriften

`big`	Verwende große Überschriften mit großen Abständen darüber und darunter.
`normal`	Verwende mittelgroße Überschriften mit mittelgroßen Abständen darüber und darunter.
`onelineappendix`, `noappendixprefix`, `appendixwithoutprefix`, `appendixwithoutprefixline`	Kapitelüberschriften im Anhang werden wie andere Überschriften auch gesetzt.
`onelinechapter`, `nochapterprefix`, `chapterwithoutprefix`, `chapterwithoutprefixline`	Kapitelüberschriften werden wie andere Überschriften auch gesetzt.
`openany`	Teile, Kapitel, Index und Nachspann verwenden `\clearpage`, aber nicht `\cleardoublepage`.

...

Tabelle 3.13.: Mögliche Werte für Option `headings` (*Fortsetzung*)

`openleft`
: Die Anweisungen `\cleardoublepageusingstyle`, `\cleardoublestandardpage`, `\cleardoubleplainpage`, `\cleardoubleemptypage` und `\cleardoublepage` erzeugen einen Seitenumbruch und fügen ggf. eine Vakatseite ein, um im doppelseitigen Satz auf die nächste linke Seite zu gelangen. Teile, Kapitel, Index und Nachspann verwenden `\cleardoublepage`.

`openright`
: Die Anweisungen `\cleardoublepageusingstyle`, `\cleardoublestandardpage`, `\cleardoubleplainpage`, `\cleardoubleemptypage` und `\cleardoublepage` erzeugen einen Seitenumbruch und fügen ggf. eine Vakatseite ein, um im doppelseitigen Satz auf die nächste rechte Seite zu gelangen. Teile, Kapitel, Index und Nachspann verwenden `\cleardoublepage`.

`optiontohead`
v3.10
: Die erweiterte Funktion des optionalen Argument der Gliederungsbefehle wird aktiviert. In der Voreinstellung wird das optionale Argument ausschließlich für den Kolumnentitel verwendet.

`optiontoheadandtoc`, `optiontotocandhead`
v3.10
: Die erweiterte Funktion des optionalen Argument der Gliederungsbefehle wird aktiviert. In der Voreinstellung wird das optionale Argument sowohl für den Kolumnentitel als auch den Eintrag ins Inhaltsverzeichnis verwendet.

`optiontotoc`
v3.10
: Die erweiterte Funktion des optionalen Argument der Gliederungsbefehle wird aktiviert. In der Voreinstellung wird das optionale Argument ausschließlich für den Eintrag ins Inhaltsverzeichnis verwendet.

`small`
: Verwende kleine Überschriften mit kleinen Abständen darüber und darunter.

`twolineappendix`, `appendixprefix`, `appendixwithprefix`, `appendixwithprefixline`
: Kapitelüberschriften im Anhang werden mit einer Vorsatzzeile gesetzt, deren Inhalt von `\chapterformat` bestimmt wird.

`twolinechapter`, `chapterprefix`, `chapterwithprefix`, `chapterwithprefixline`
: Kapitelüberschriften werden mit einer Vorsatzzeile gesetzt, deren Inhalt von `\chapterformat` bestimmt wird.

`numbers=`*`Einstellung`*

Nach DUDEN steht in Gliederungen, in denen ausschließlich arabische Ziffern für die Nummerierung verwendet werden, am Ende der Gliederungsnummern kein abschließender Punkt (siehe [DUD96, R 3]). Wird hingegen innerhalb der Gliederung auch mit römischen Zahlen oder Groß- oder Kleinbuchstaben gearbeitet, so steht am Ende aller Gliederungsnummern ein abschließender Punkt (siehe [DUD96, R 4]). In KOMA-Script ist ein Automatismus eingebaut, der diese etwas komplexe Regel zu erfüllen versucht. Der Automatismus wirkt sich so aus, dass normalerweise bei Verwendung des Gliederungsbefehls `\part` oder eines Anhangs (`\appendix`) auf Gliederungsnummer mit abschließendem Punkt umgeschaltet wird. Diese Information wird in der `aux`-Datei gespeichert und wirkt sich dann beim nächsten LaTeX-Lauf auf das gesamte Dokument aus.

Manchmal versagt der mit `numbers=autoendperiod` voreingestellte Automatismus zum Setzen oder Weglassen des abschließenden Punktes in der Gliederungsnummer. Teilweise sehen andere Sprachen auch andere Regeln vor. Deshalb ist es beispielsweise mit der Einstellung `numbers=endperiod` möglich, den Punkt manuell vorzuschreiben oder mit `numbers=noendperiod` zu verbieten.

Es ist zu beachten, dass der Automatismus immer erst für den nächsten LaTeX-Lauf die Verwendung des abschließenden Punktes ein- oder ausschaltet. Bevor also versucht wird, die korrekte Darstellung über Verwendung einer der Optionen zu erzwingen, sollte grundsätzlich ein weiterer LaTeX-Lauf ohne Dokumentänderung durchgeführt werden.

Ein Zusammenfassung der möglichen Werte für die *`Einstellung`* von *`numbers`* bietet Tabelle 3.14. Im Unterschied zu den meisten anderen Einstellungen, kann diese Option nur in der Dokumentpräambel, also vor `\begin{document}` vorgenommen werden.

`chapteratlists`
`chapteratlists=`*`Wert`*

scrbook, scrreprt, ~~scrartv2.96a~~

Wie auch bei der Option `listof` in Abschnitt 3.20, Seite 134 erwähnt wird, fügt normalerweise jeder mit `\chapter` erzeugte Kapiteleintrag einen vertikalen Abstand in die Verzeichnisse der Gleitumgebungen ein. Seit Version 2.96a gilt dies auch für die Anweisung `\addchap`, wenn nicht eine Kompatibilitätseinstellung zu einer früheren Version gewählt wurde (siehe Option `version` in Abschnitt 3.2, Seite 31).

Außerdem kann mit der Option `chapteratlists` der Abstand verändert werden. Dazu gibt man als *`Wert`* den gewünschten Abstand an. Bei der Voreinstellung `listof=chaptergapsmall`sind dies 10 pt. Mit der Einstellung `chapteratlists=entry` oder bei Verwendung der Form `chapteratlists` ohne Angabe eines Wertes wird statt des Abstandes der Kapiteleintrag selbst in die Verzeichnisse eingetragen. Es wird darauf hingewiesen, dass ein solcher Eintrag auch dann erfolgt, wenn das Kapitel keine Gleitumgebung enthält.

Es ist zu beachten, dass sich eine Änderung der Einstellung je nach Art der Änderung erst nach zwei weiteren LaTeX-Läufen im Verzeichnis auswirkt.

Tabelle 3.14.: Mögliche Werte für Option `numbers` zur Konfigurierung des Abschlusspunktes in Gliederungsnummern

`autoendperiod`, `autoenddot`, `auto` KOMA-Script trifft die Entscheidung, ob am Ende von Gliederungsnummern und allen von Gliederungsnummern abhängigen Nummern ein Punkt gesetzt wird, selbst. Kommen in sämtlichen Gliederungsnummern nur arabische Ziffern vor, so wird kein Punkt gesetzt. Wird in einer Gliederungsnummer ein Buchstabe oder eine römische Zahl entdeckt, so wird der Punkt bei allen Nummern gesetzt. Referenzen auf diese Nummern werden jedoch ohne abschließenden Punkt gesetzt.
`endperiod`, `withendperiod`, `periodatend`, `enddot`, `withenddot`, `dotatend` Bei sämtlichen Gliederungsnummern und davon abhängigen Nummern wird am Ende ein Punkt gesetzt, der bei der Referenzierung entfällt.
`noendperiod`, `noperiodatend`, `noenddot`, `nodotatend` Gliederungsnummern und davon abhängige Nummern werden ohne abschließenden Punkt gesetzt.

```
\part[Kurzform]{Überschrift}
\chapter[Kurzform]{Überschrift}
\section[Kurzform]{Überschrift}
\subsection[Kurzform]{Überschrift}
\subsubsection[Kurzform]{Überschrift}
\paragraph[Kurzform]{Überschrift}
\subparagraph[Kurzform]{Überschrift}
```

Die Standardgliederungsbefehle funktionieren bei KOMA-Script im Grundsatz wie bei den Standardklassen. So kann in der Voreinstellung ganz normal über ein optionales Argument ein abweichender Text für den Kolumnentitel und das Inhaltsverzeichnis vorgegeben werden.

v3.10 Mit der Einstellung `headings=optiontohead` wird das optionale Argument bei KOMA-Script hingegen nur noch für den Kolumnentitel verwendet. Ein konkreter Eintrag in den Kolumnentitel erfolgt natürlich nur, wenn ein Seitenstil gewählt wird, bei dem die entsprechende Gliederungsebene überhaupt für den Kolumnentitel verwendet wird. Siehe hierzu Abschnitt 3.12 sowie Kapitel 5. Mit der Einstellung `headings=optiontotoc` wird das optionale Argument hingegen ausschließlich für den Eintrag ins Inhaltsverzeichnis verwendet. Auch dies selbstverständlich nur, wenn die Einstellung für den Zähler `tocdepth` (siehe Abschnitt 3.9, Seite 70) einen Eintrag der entsprechenden Ebene überhaupt vorsieht. Mit der Einstellung `headings=optiontoheadandtoc` findet schließlich das optionale Argument wieder sowohl für den Kolumnentitel als auch den Eintrag ins Inhaltsverzeichnis Verwendung. Allen drei Einstellungen ist gemeinsam, dass sie im Gegensatz zur Voreinstellung die erweiterte Interpretation des optionalen Arguments aktivieren.

v3.10

Bei der erweiterten Interpretation des optionalen Arguments wird geprüft, ob sich ein Gleichheitszeichen in *Kurzform* befindet. Ist dies der Fall, so wird das optionale Argument der Gliederungsbefehle selbst statt als *Kurzform* als *Optionenliste* interpretiert. Dabei werden die beiden Optionen `head=`*Kolumnentitel* und `tocentry=`*Inhaltsverzeichniseintrag* akzeptiert. Um ein Gleichheitszeichen oder ein Komma in einem der beiden Werte dieser Optionen unter zu bringen, muss dieses in zusätzliche geschweifte Klammern gesetzt werden.

Bitte beachten Sie, dass dieser Mechanismus nur funktioniert, solange KOMA-Script die Kontrolle über die Gliederungsbefehle besitzt. Wird hingegen ein Paket verwendet, dass die Gliederungsbefehle oder die internen LaTeX-Kern-Anweisungen für Gliederungsbefehle umdefiniert, so kann KOMA-Script diese erweiterte Funktionalität nicht mehr zur Verfügung stellen. Dies gilt auch für die immer aktive Erweiterung, dass leere Inhaltsverzeichniseinträge nicht zu einem Eintrag ohne Text führen, sondern gänzlich unterbleiben. Soll tatsächlich einmal ein leerer Eintrag ins Inhaltsverzeichnis erfolgen, so ist kann dies mit einem unsichtbaren Eintrag wie `\mbox{}` erreicht werden.

Beispiel: Angenommen, Sie haben ein Dokument mit teilweise sehr langen Kapitelüberschriften. Diese lange Kapitelüberschriften sollen auch im Inhaltsverzeichnis ausgegeben werden. Die Kolumnentitel wollen Sie jedoch auf einzeilige Kurztexte beschränken. Dazu stellen Sie mit

```
\documentclass[headings=optiontohead]{scrbook}
```

bereits beim Laden der Klasse ein, dass das optionale Argument der Gliederungsbefehle nur für die Kolumnentitel verwendet werden soll. Im Dokument nehmen Sie dann einen entsprechenden Eintrag über das optionale Argument von `\chapter` vor.

```
\chapter[Kurzformen für Kapitel]
        {Der Gliederungsbefehl für
          Kapitelüberschriften erlaubt neben dem
          Text für die eigentliche
          Kapitelüberschrift auch eine Kurzform
          mit steuerbarer Verwendung}
```

Etwas später wird Ihnen bewusst, dass diese lange Überschrift sehr ungünstig umbrochen wird. Sie wollen deshalb die Umbrüche für diese Überschrift selbst bestimmen. Im Inhaltsverzeichnis soll allerdings weiterhin automatisch umbrochen werden. Mit

```
\chapter[head={Kurzformen für Kapitel},
         tocentry={Der Gliederungsbefehl für
           Kapitelüberschriften erlaubt neben
           dem Text für die eigentliche
           Kapitelüberschrift auch eine Kurzform
           mit steuerbarer Verwendung}]
```

```
{Der Gliederungsbefehl für
  Kapitelüberschriften\\
  erlaubt neben dem\\
  Text für die eigentliche
  Kapitelüberschrift\\
  auch eine Kurzform\\
  mit steuerbarer Verwendung}
```

setzen Sie daher die Einträge für den Kolumnentitel und das Inhaltsverzeichnis unabhängig voneinander und von der Überschrift selbst. Die Argumente der beiden Optionen `head` und `tocentry` wurden dabei in geschweifte Klammern gesetzt, damit der Inhalt der Argumente beliebig sein kann.

Die Notwendigkeit der geschweiften Klammern im vorherigen Beispiel lässt sich am besten an einem weiteren Beispiel verdeutlichen. Angenommen, sie haben als Option `headings=optiontotoc` gewählt und setzen nun die Überschrift:

```
\section[head=\emph{Wert}]
        {Die Option head=\emph{Wert}}
```

Dies führt dazu, dass im Inhaltsverzeichnis der Eintrag »Die Option head= *Wert*« und im Kolumnentitel der Eintrag » *Wert*« verwendet wird. In Wirklichkeit wollten Sie aber, dass im Inhaltsverzeichnis der Eintrag »head= *Wert*« lautet und im Kolumnentitel der Text der Überschrift übernommen wird. Dies ist dadurch zu erreichen, dass das Gleichheitszeichen in geschweifte Klammern gesetzt wird:

```
\section[head{=}\emph{Wert}]
        {Die Option head=\emph{Wert}}
```

Ein ähnlicher Fall betrifft das Komma. Bei gleicher Voreinstellung der Optionen würde

```
\section[head=0, 1, 2, 3, \dots]
        {Die natürlichen Zahlen mit der Null}
```

zu einer Fehlermeldung führen, weil die Kommata als Trennzeichen zwischen den einzelnen Optionen der Optionenliste »`head=0, 1, 2, 3, \dots`« interpretiert würden. Schreibt man hingegen

```
\section[head={0, 1, 2, 3, \dots}]
        {Die natürlichen Zahlen mit der Null}
```

So ist »`0, 1, 2, 3, \dots`« das Argument der Option `head`.

Die Überschrift der Teile-Ebene (`\part`) unterscheidet sich von den anderen Gliederungsebenen dadurch, dass sie unabhängig von den übrigen Ebenen nummeriert wird. Das bedeutet, dass die Kapitel-Ebene (bei scrbook oder scrreprt) bzw. die Abschnitt-Ebene (bei scrartcl) über alle Teile hinweg durchgehend nummeriert wird. Darüber hinaus steht bei den Klassen scrbook und scrreprt die Überschrift dieser Ebene zusammen mit ihrer Präambel (siehe Anweisung

scrbook, scrreprt

\setpartpreamble, Seite 105) alleine auf einer Seite.

scrbook, scrreprt

\chapter existiert nur bei Buch- und Berichtklassen, also bei book, scrbook, report und scrreprt, nicht jedoch bei den Artikelklassen article und scrartcl. \chapter unterscheidet sich bei KOMA-Script außerdem gravierend von der Version der Standardklassen. Bei den Standardklassen wird die Kapitelnummer mit dem Präfix »Kapitel« beziehungsweise dem Kapitelnamen in der gewählten Sprache in einer Zeile vor dem eigentlichen Text der Überschrift ausgegeben. Diese sehr mächtige Form wird bei KOMA-Script durch eine einfache Nummer vor dem Text abgelöst, lässt sich aber durch die Optionen chapterprefix und appendixprefix einstellen (siehe Seite 89).

scrbook, scrreprt

Bitte beachten Sie, dass \part und \chapter bei scrbook und scrreprt den Seitenstil für eine Seite umschaltet. Der jeweilige Seitenstil ist bei KOMA-Script in den Makros \partpagestyle und \chapterpagestyle abgelegt (siehe Abschnitt 3.12, Seite 78).

v2.8p

Die Schriftart aller sieben Gliederungsebenen kann mit den Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 3.6, Seite 56) bestimmt werden. Dabei wird zunächst generell das Element disposition und anschließend zusätzlich je Gliederungsebene ein spezifisches Element verwendet (siehe Tabelle 3.2, Seite 57). Die Schriftart für das Element disposition ist als \normalcolor\sffamily\bfseries vordefiniert. Die Voreinstellungen für die spezifischen Elemente sind mit einer Schriftgröße vorbelegt und daher von den Einstellungen big, normal und small für die Option headings abhängig (siehe Seite 89). Sie finden die Voreinstellungen in Tabelle 3.15.

Beispiel: Angenommen, Sie stellen bei Verwendung der Klassenoption headings=big fest, dass die sehr großen Überschriften von Teildokumenten zu fett wirken. Nun könnten Sie natürlich wie folgt verfahren:

```
\setkomafont{disposition}{\normalcolor\sffamily}
\part{\appendixname}
\addtokomafont{disposition}{\bfseries}
```

Auf diese Weise würden Sie nur für die eine Überschrift »Anhang« das Schriftattribut **Fett** abschalten. Sehr viel komfortabler und eleganter ist es aber, stattdessen generell für \part-Überschriften eine entsprechende Änderung vorzunehmen. Das ist wahlweise mit:

```
\addtokomafont{part}{\normalfont\sffamily}
\addtokomafont{partnumber}{\normalfont\sffamily}
```

oder einfach mit:

```
\addtokomafont{part}{\mdseries}
\addtokomafont{partnumber}{\mdseries}
```

möglich. Die Verwendung von \setkomafont wäre zwar grundsätzlich möglich, müsste aber auch die Auswahl der Schriftgröße enthalten und würde damit die Größenänderung über die Option headings verhindern.

Tabelle 3.15.: Voreinstellungen der Schrift für die Elemente der Gliederung bei scrbook und scrreprt

Klassenoption	Element	Voreinstellung
`headings=big`	`part`	`\Huge`
	`partnumber`	`\huge`
	`chapter`	`\huge`
	`section`	`\Large`
	`subsection`	`\large`
	`subsubsection`	`\normalsize`
	`paragraph`	`\normalsize`
	`subparagraph`	`\normalsize`
`headings=normal`	`part`	`\huge`
	`partnumber`	`\huge`
	`chapter`	`\LARGE`
	`section`	`\Large`
	`subsection`	`\large`
	`subsubsection`	`\normalsize`
	`paragraph`	`\normalsize`
	`subparagraph`	`\normalsize`
`headings=small`	`part`	`\LARGE`
	`partnumber`	`\LARGE`
	`chapter`	`\Large`
	`section`	`\large`
	`subsection`	`\normalsize`
	`subsubsection`	`\normalsize`
	`paragraph`	`\normalsize`
	`subparagraph`	`\normalsize`

Die zweite Version mit `\mdseries` ist vorzuziehen, da diese auch dann noch zum gewünschten Ergebnis führt, wenn Sie später das Element `disposition` wie folgt ändern:

```
\setkomafont{disposition}{\normalcolor\bfseries}
```

Mit dieser Änderung verzichten Sie darauf, für alle Gliederungsebenen serifenlose Schrift voreinzustellen.

Ich möchte eindringlich davon abraten, die Möglichkeit zur Schriftumschaltung zu missbrauchen, um wild Schriften, Schriftgrößen und Schriftattribute miteinander zu mischen. Die Auswahl der richtigen Schrift für die richtige Aufgabe ist eine Sache für Experten und hat sehr, sehr wenig mit dem persönlichem Geschmack eines Laien zu tun. Siehe hierzu auch das Zitat am Ende von Abschnitt 2.8, Seite 51 und die folgende Erklärung.

Unterschiedliche Schriften für unterschiedliche Gliederungsebenen sind mit KOMA-Script-Mitteln möglich. Der Laie sollte sie aber meiden wie der Teufel das Weihwasser. Dies hat typografische Gründe.

Eine Regel der Typografie besagt, dass man möglichst wenig Schriften miteinander mischen soll. Serifenlose für die Überschriften scheinen bereits ein Verstoß gegen diese Regel zu sein. Allerdings muss man wissen, dass fette, große, serifenbehaftete Buchstaben oft viel zu mächtig für eine Überschrift sind. Man müsste dann strenggenommen zumindest auf eine normale statt eine fette oder halbfette Schrift ausweichen. In tiefen Gliederungsebenen kann das aber wieder zu schwach sein. Andererseits haben Serifenlose in Überschriften eine sehr angenehme Wirkung und fast nur für Überschriften eine Berechtigung. Daher wurde diese Voreinstellung für KOMA-Script mit gutem Grund gewählt.

Größere Vielfalt sollte aber vermieden werden. Schriftenmischung ist etwas für Profis. Aus den genannten Gründen sollten Sie bei Verwendung anderer als der Standard-TEX-Fonts – egal ob CM-, EC- oder LM-Fonts – bezüglich der Verträglichkeit der serifenlosen und serifenbehafteten Schrift einen Experten zu Rate ziehen oder die Schrift für das Element `disposition` vorsichtshalber wie in obigem Beispiel umdefinieren. Die häufig anzutreffenden Kombinationen Times mit Helvetica oder Palatino mit Helvetica werden vom Autor als ungünstig betrachtet.

```
\part*{Überschrift}
\chapter*{Überschrift}
\section*{Überschrift}
\subsection*{Überschrift}
\subsubsection*{Überschrift}
\paragraph*{Überschrift}
\subparagraph*{Überschrift}
```

Bei den Sternvarianten der Gliederungsbefehle erfolgt keine Nummerierung, wird kein Kolumnentitel gesetzt und kein Eintrag im Inhaltsverzeichnis vorgenommen. Der Verzicht auf den Kolumnentitel hat übrigens einen oftmals unerwünschten Effekt. Geht beispielsweise ein mit `\chapter*` gesetztes Kapitel über mehrere Seiten, so taucht plötzlich der Kolumnentitel des letzten Kapitels wieder auf. KOMA-Script bietet dafür aber eine Lösung, die im Anschluss beschrieben wird. `\chapter*` existiert selbstverständlich nur bei Buch- und Berichtklassen, also bei book, scrbook, report und scrreprt, nicht jedoch bei den Artikelklassen article und scrartcl.

scrbook, scrreprt

Bitte beachten Sie, dass `\part*` und `\chapter*` den Seitenstil für eine Seite umschalten. Während die Standardklassen dabei den Seitenstil `plain` verwenden, ist bei KOMA-Script der zu verwendende Seitenstil in den Makros `\partpagestyle` und `\chapterpagestyle` abgelegt (siehe Abschnitt 3.12, Seite 78).

v2.8p

Bezüglich der Möglichkeiten der Schriftumschaltung gilt das Gleiche wie zuvor in der Erklärung zu den sternlosen Varianten geschrieben. Die Elemente tragen die gleichen Namen, da sie nicht Varianten, sondern Gliederungsebenen bezeichnen.

Bei den Standardklassen gibt es keine weiteren Gliederungsbefehle. Es existieren also insbesondere keine Anweisungen, mit denen man nicht nummerierte Kapitel oder nicht nummerierte

Abschnitte erzeugen kann, die ins Inhaltsverzeichnis aufgenommen werden und bei denen ein automatischer Kolumnentitel erzeugt wird.

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

KOMA-Script bietet über die Gliederungsbefehle der Standardklassen hinaus die Anweisungen `\addpart`, `\addchap` und `\addsec`. Diese ähneln bis auf die fehlende Nummerierung sehr den Standardanweisungen `\part`, `\chapter` und `\section`. Sie erzeugen also sowohl einen automatischen Kolumnentitel als auch einen Eintrag im Inhaltsverzeichnis, wobei auch die Einstellungen von Option `headings` beachtet werden.

Die Sternvarianten `\addchap*` und `\addsec*` gleichen hingegen den Standardanweisungen `\chapter*` und `\section*` mit einem winzigen, aber wichtigen Unterschied: Die Kolumnentitel werden gelöscht. Dadurch wird der oben erwähnte Effekt veralteter Kolumnentitel ausgeschlossen. Stattdessen bleibt der Kolumnentitel auf Folgeseiten leer. `\addchap` und `\addchap*` existieren selbstverständlich nur bei Buch- und Berichtklassen, also bei scrbook und scrreprt, nicht jedoch bei der Artikelklasse scrartcl.

scrbook, scrreprt

Die Anweisung `\addpart` erstellt entsprechend einen nicht nummerierten Dokumentteil mit einem Eintrag im Inhaltsverzeichnis. Da bereits `\part` und `\part*` den Kolumnentitel löschen, ergibt sich hier nicht das oben genannte Problem mit veralteten Kolumnentiteln. Die Sternvariante `\addpart*` ist daher identisch mit der Sternvariante `\part*` und wurde nur aus Konsistenzgründen definiert.

Bitte beachten Sie, dass `\addpart` und `\addchap` und deren Sternvarianten den Seitenstil für eine Seite umschalten. Der jeweilige Seitenstil ist in den Makros `\partpagestyle` und `\chapterpagestyle` abgelegt (siehe Abschnitt 3.12, Seite 78).

v2.8p

Bezüglich der Möglichkeiten der Schriftumschaltung gilt das gleiche wie zuvor in der Erklärung zu `\part*`, `\chapter*` und `\section*` geschrieben. Die Elemente tragen die gleichen Namen, da sie nicht Varianten, sondern Gliederungsebenen bezeichnen.

```
\minisec{Überschrift}
```

Manchmal ist eine Art Überschrift wünschenswert, die zwar hervorgehoben wird, ansonsten aber eng mit dem nachfolgenden Text zusammenhängt. Eine solche Überschrift soll dann ohne große Abstände gesetzt werden.

Der Befehl `\minisec` bewirkt genau eine derartige Überschrift. Diese Überschrift ist keiner Gliederungsebene zugeordnet. Eine solche *Mini-Section* wird nicht in das Inhaltsverzeichnis aufgenommen und erhält auch keine Nummerierung.

v2.96a

Die Schriftart des Gliederungsbefehls `\minisec` kann über die Elemente `disposition` und `minisec` beeinflusst werden (siehe Tabelle 3.2, Seite 57). Die Voreinstellung für das Element `minisec` ist leer, so dass in der Voreinstellung nur das Element `disposition` wirkt.

Beispiel: Sie haben einen Bausatz für eine Mausefalle entwickelt und wollen diesen getrennt nach den benötigten Materialien und der Anleitung für die Montage beschreiben. Das könnte so gemacht werden:

```
\documentclass{scrartcl}
\usepackage[ngerman]{babel}
\usepackage{selinput}
\SelectInputMappings{
  adieresis={ä},
  germandbls={ß}
}

\begin{document}

\title{Selbstbauprojekte}
\author{Zwei Linke Daumen}
\date{\today}
\maketitle

\section{Mausefalle}
Das erste Projekt ist auch für Anfänger geeignet
und benötigt lediglich einige wenige Bauteile,
die in jedem Haushalt zu finden sein sollten.

\minisec{Bauteile}

\begin{flushleft}
  1 Brett ($100\times 50 \times 12$)\\
  1 Bierflaschenschnappverschluss\\
  1 Kugelschreiberfeder\\
  1 Reißzwecke\\
  2 Schrauben\\
  1 Hammer\\
  1 Messer
\end{flushleft}

\minisec{Montage}

Zunächst suche man das Mauseloch. Dann lege man die
Reißzwecke innen unmittelbar hinter das Loch, damit
bei den folgenden Aktionen die Maus nicht
entschlüpfen kann.
```

```
Anschließend klopfe man mit dem Hammer den
Bierflaschenschnappverschluss in das Mauseloch.
Sollte der Verschluss nicht groß genug sein, um das
Loch vollständig und dauerhaft zu verschließen,
nehme man stattdessen das Brett und schraube es
unter Zuhilfenahme der beiden Schrauben und des
Messers vor das Loch. Statt des Messers kann
selbstverständlich auch ein Schraubendreher
verwendet werden.
Die Kugelschreiberfeder ist dem Tierschutz zum
Opfer gefallen.
\end{document}
```

Der wesentliche Teil aber der Überschrift »Bauteile« sieht anschließend wie folgt aus:

Bauteile
1 Brett (100 × 50 × 12)
1 Bierflaschenschnappverschluss
1 Kugelschreiberfeder
1 Reißzwecke
2 Schrauben
1 Hammer
1 Messer

Montage
Zunächst suche man das Mauseloch. Dann lege man die Reißzwecke innen unmittelbar hinter das Loch, damit bei den folgenden Aktionen die Maus nicht entschlüpfen kann. Anschließend klopfe man mit dem Hammer den Bierflaschenschnappverschluss in das Mauseloch. Sollte der Verschluss nicht groß genug sein, um das Loch vollständig und dauerhaft zu verschließen, nehme man stattdessen das Brett und schraube es unter Zuhilfenahme der beiden Schrauben und des Messers vor das Loch. Statt des Messers kann selbstverständlich auch ein Schraubendreher verwendet werden. Die Kugelschreiberfeder ist dem Tierschutz zum Opfer gefallen.

Zum Verständnis der Verwendung des Pakets selinput und der darin definierten Anweisung `\SelectInputMappings` sei auf [Obe07] verwiesen.

`\raggedsection`
`\raggedpart`

Bei den Standardklassen werden die Überschriften ganz normal im Blocksatz ausgegeben. Dadurch können in den Überschriften Trennungen auftreten und mehrzeilige Überschriften werden auf Textbreite gedehnt. Dieses Vorgehen ist in der Typografie eher unüblich. KOMA-Script setzt Überschriften von `\chapter` bis `\subparagraph` daher in linksbündigem Flattersatz mit hängendem Einzug. Verantwortlich ist dafür die Anweisung `\raggedsection`, die vordefiniert ist als:

```
\let\raggedsection\raggedright
```

Die Anweisung `\raggedsection` kann mit `\renewcommand` umdefiniert werden.

Beispiel: Sie wollen auch für Überschriften Blocksatz. Dazu schreiben Sie in die Präambel Ihres Dokuments:

```
\renewcommand*{\raggedsection}{}
```

oder kürzer:

```
\let\raggedsection\relax
```

Sie erreichen somit eine ähnliche Formatierung der Überschriften wie bei den Standardklassen. Noch ähnlicher wird es, wenn Sie diese Änderung mit der oben vorgestellten Änderung für das Element `disposition` kombinieren.

Die Überschriften von Teilen (`\part`) werden in der Voreinstellung als einzige nicht in linksbündigem Flattersatz, sondern zentriert gesetzt. Dafür ist die Anweisung `\raggedpart` verantwortlich, die als:

```
\let\raggedpart\centering
```

vordefiniert ist. Auch diese Anweisung kann mit `\renewcommand` umdefiniert werden.

Beispiel: Sie wollen, dass Überschriften mit `\part` in der gleichen Weise formatiert werden, wie alle anderen Ebenen. Dazu schreiben Sie

```
\renewcommand*{\raggedpart}{\raggedsection}
```

in die Präambel Ihres Dokuments. An dieser Stelle wurde im Gegensatz zu oben absichtlich nicht `\let` verwendet, da mit `\let` der Anweisung `\raggedpart` die aktuelle Bedeutung von `\raggedsection` zugewiesen würde. Spätere Änderungen von `\raggedsection` blieben also bei `\raggedpart` unberücksichtigt. Bei der Umdefinierung mit `\renewcommand` erhält `\raggedpart` dagegen nicht die Bedeutung von `\raggedsection` zum Zeitpunkt dieser Umdefinierung, sondern zum Zeitpunkt der Verwendung von `\raggedpart`.

```
\partformat
\chapterformat
\othersectionlevelsformat{Gliederungsname}{}{Zählerausgabe}
\autodot
```

KOMA-Script fügt der Ausgabe der Gliederungsnummern oberhalb von `\the`*`Gliederungsname`* (siehe `\the`*`Zähler`*, Seite 307) eine weitere logische Ebene hinzu. Die Zähler werden für die jeweilige Überschrift nicht einfach nur ausgegeben. Sie werden mit Hilfe der parameterlosen Anweisungen `\partformat`, `\chapterformat` und der Anweisung `\othersectionlevelsformat`, die drei Parameter erwartet, formatiert. Die Anweisung

scrbook, scrreprt

\chapterformat existiert wie bereits \thechapter selbstverständlich nicht in der Klasse scrartcl, sondern nur in den Klassen scrbook und scrreprt.

Wie bereits bei Option numbers am Anfang dieses Abschnitts (siehe Seite 92) erläutert wurde, müssen gemäß [DUD96] die Gliederungsnummern je nach Gliederung mit einem nachfolgenden Punkt versehen werden oder dieser hat zu entfallen. Die Anweisung \autodot ist bei KOMA-Script für die Einhaltung dieser Regel verantwortlich. Auf den Punkt folgt bei allen Gliederungsebenen außer \part noch ein \enskip. Dies entspricht einem Leerraum von 0,5 em, also einem Halbgeviert.

Die Anweisung \othersectionlevelsformat erwartet als ersten Parameter den Namen der Gliederungsebene, also section, subsection, subsubsection, paragraph oder subparagraph. Als drittes Argument wird die Ausgabe des Zählers dieser Gliederungsebene, also \thesection, \thesubsection, \theparagraph oder \thesubparagraph, erwartet.

In der Voreinstellung gibt es nur für die beiden Ebenen \part und \chapter eigene Formatierungsanweisungen, während alle anderen Gliederungsebenen mit Hilfe einer einzigen Formatierungsanweisung abgedeckt werden. Dies ist allein historisch begründet. Als Werner Lemberg für sein CJK-Paket (siehe [Lem08]) eine entsprechende Erweiterung von KOMA-Script angeregt hat, wurde nur diese Unterscheidung benötigt. Anwender sollten bei der Umdefinierung der Anweisung das zweite Argument komplett ignorieren.

Mit Hilfe von \renewcommand kann jeder der drei Formatierungsanweisungen umdefiniert werden, um sie eigenen Anforderungen anzupassen. Nachfolgend finden Sie die Originaldefinitionen aus den KOMA-Script-Klassen:

```
\newcommand*{\partformat}{\partname~\thepart\autodot}
\newcommand*{\chapterformat}{%
  \mbox{\chapappifchapterprefix{\nobreakspace}\thechapter
    \autodot\enskip}}
\newcommand*{\othersectionlevelsformat}[3]{%
  #3\autodot\enskip}
```

Bitte beachten Sie, dass bei der Umdefinierung \newcommand durch \renewcommand zu ersetzen ist.

Beispiel: Angenommen, Sie wollen, dass bei \part das Wort »Teil« vor der Nummer nicht ausgegeben wird. Dann können Sie beispielsweise folgende Anweisung in die Präambel Ihres Dokuments schreiben:

```
\renewcommand*{\partformat}{\thepart\autodot}
```

Genau genommen könnten Sie an dieser Stelle auch auf \autodot verzichten und stattdessen einen festen Punkt setzen. Da \part mit römischen Zahlen nummeriert wird, muss der Punkt laut [DUD96] folgen. Allerdings bringen Sie sich dann um die Möglichkeit, die Option numbers einzusetzen und so von der Regel abzuweichen. Näheres zu der Option siehe Seite 92.

Eine weitere Möglichkeit besteht darin, die Nummerierung der Abschnitte so in

den Rand zu platzieren, dass der Überschriftentext links mit dem umgebenden Text abschließt. Dies erreicht man mit:

```
\renewcommand*{\othersectionlevelsformat}[3]{%
  \llap{#3\autodot\enskip}}
```

Die wenig bekannte TEX-Anweisung `\llap` sorgt dabei dafür, dass das Argument links von der aktuellen Position ausgegeben wird, ohne dass dabei die Position verändert wird. Eine bessere LATEX-Lösung für dieses Problem wäre:

```
\renewcommand*{\othersectionlevelsformat}[3]{%
  \makebox[0pt][r]{#3\autodot\enskip}}
```

Näheres zu den optionalen Argumenten von `\makebox` ist [Tea05b] zu entnehmen.

`\chapappifchapterprefix{`*Zusatztext*`}`
`\chapapp`

scrbook, scrreprt (2.96)

Diese beiden Anweisungen werden nicht nur intern von KOMA-Script verwendet, sondern stehen auch dem Anwender zur Verfügung. Nachfolgend werden sie beispielsweise für die Umdefinierung anderer Anweisungen verwendet.

Bei Verwendung von Option `chapterprefix=true` (siehe Seite 89) setzt `\chapappifchapterprefix` im Hauptteil des Dokuments das Wort »Kapitel« in der aktuellen Sprache gefolgt vom *Zusatztext*. Im Anhang wird stattdessen das Wort »Anhang« in der aktuellen Sprache, ebenfalls gefolgt vom *Zusatztext*, ausgegeben. Bei der Einstellung `chapterprefix=false` wird hingegen nichts ausgegeben.

Die Anweisung `\chapapp` setzt immer das Wort »Kapitel« beziehungsweise »Anhang«. Dabei spielt die Einstellung der Option `chapterprefix` keine Rolle.

Da es Kapitel nur bei den Klassen scrbook und scrreprt gibt, existieren die beiden Anweisungen auch nur bei diesen Klassen.

`\chaptermark{`*Kolumnentitel*`}`
`\sectionmark{`*Kolumnentitel*`}`
`\subsectionmark{`*Kolumnentitel*`}`
`\chaptermarkformat`
`\sectionmarkformat`
`\subsectionmarkformat`

Wie bereits in Abschnitt 3.12 erwähnt, arbeitet der Seitenstil `headings` mit automatischen Kolumnentiteln. Dazu werden die Anweisungen `\chaptermark` und `\sectionmark` beziehungsweise `\sectionmark` und `\subsectionmark` entsprechend definiert. Gliederungsbefehle (`\chapter`, `\section` ...) führen automatisch eine entsprechende `\...mark`-Anweisung aus. Der übergebene Parameter beinhaltet dabei den Text der Gliederungsüberschrift. Die zugehörige Gliederungsnummer wird automatisch in der `\...mark`-Anweisung hinzugefügt. Die Formatierung erfolgt dabei je nach Gliederungsebene mit einer der drei Anweisungen `\chaptermarkformat`, `\sectionmarkformat` und `\subsectionmarkformat`.

scrbook, scrreprt

scrartcl

Während bei scrartcl `\chaptermark` und `\chaptermarkformat` nicht existieren, gibt es die Anweisungen `\subsectionmark` und `\subsectionmarkformat` nur bei scrartcl. Dies ändert sich allerdings bei Verwendung des scrpage2-Pakets (siehe Kapitel 5).

So wie mit `\chapterformat` und `\othersectionlevelsformat` die Nummern der Gliederungsüberschriften formatiert ausgegeben werden, werden mit den Anweisungen `\chaptermarkformat`, `\sectionmarkformat` und `\subsectionmarkformat` die Nummern der Gliederungsebenen in den automatischen Kolumnentiteln formatiert ausgegeben. Mit `\renewcommand` können sie eigenen Anforderungen angepasst werden. Die Originaldefinitionen aus den KOMA-Script-Klassen sind:

```
\newcommand*{\chaptermarkformat}{%
  \chapappifchapterprefix{\ }\thechapter\autodot\enskip}
\newcommand*{\sectionmarkformat}{\thesection\autodot\enskip}
\newcommand*{\subsectionmarkformat}{%
  \thesubsection\autodot\enskip}
```

Beispiel: Angenommen, Sie wollen, dass der Kapitelnummer in den Kolumnentiteln das Wort »Kapitel« vorangestellt wird. Dann setzen Sie beispielsweise diese Definition in die Präambel Ihres Dokuments:

```
\renewcommand*{\chaptermarkformat}{%
  \chapapp~\thechapter\autodot\enskip}
```

Wie Sie sehen, finden hier bei der Umdefinierung die Anweisungen `\chapappifchapterprefix` und `\chapapp` Verwendung, die weiter oben erklärt wurden.

`secnumdepth`

Normalerweise werden bei den Klassen scrbook und scrreprt die Gliederungsebenen `\part` bis `\subsection` und bei der Klasse scrartcl die Ebenen `\part` bis `\subsubsection` nummeriert. Gesteuert wird dies über den LaTeX-Zähler `secnumdepth`. Dabei steht der Wert −1 für `\part`, 0 für `\chapter` und so weiter. Durch Setzen oder Erhöhen oder Verringern des Zählers kann bestimmt werden, bis zu welcher Gliederungsebene eine Nummerierung erfolgen soll. Dies ist übrigens bei den Standardklassen ganz genauso. Vergleichen Sie hierzu auch die Erklärung zum Zähler `tocdepth` in Abschnitt 3.9, Seite 70.

`\setpartpreamble[`*Position*`][`*Breite*`]{`*Präambel*`}`
`\setchapterpreamble[`*Position*`][`*Breite*`]{`*Präambel*`}`

scrbook, scrreprt

Teile und Kapitel können bei KOMA-Script mit einer *Präambel* versehen werden. Dies ist insbesondere im zweispaltigen Layout mit der Klassenoption `twocolumn` nützlich, da die *Präambel* zusammen mit der Überschrift einspaltig gesetzt wird. Die *Präambel* kann auch mehrere Absätze beinhalten. Die Anweisung zum Setzen der *Präambel* muss vor der jeweiligen `\part`- oder `\addpart`- bzw. `\chapter`- oder `\addchap`-Anweisung stehen.

Beispiel: Sie schreiben einen Bericht über den Zustand einer Firma. Dabei organisieren Sie den Bericht so, dass jeder Abteilung ein eigener Teilbericht spendiert wird. Jedem dieser Teile soll außerdem eine Zusammenfassung vorangestellt werden. Diese Zusammenfassung soll auf der Titelseite jedes Teils stehen. Das ist wie folgt möglich:

```
\setpartpreamble{%
  \begin{abstract}
    Dies ist ein Blindtext zur Demonstration.
  \end{abstract}
}
\part{Abteilung Grünschnitt}
```

Je nach Einstellung der Optionen für die Überschriftengröße (siehe Seite 89) und der Optionen für die Form der `abstract`-Umgebung (siehe Abschnitt 3.8, Seite 66), sieht das Ergebnis ungefähr wie folgt aus:

Bitte beachten Sie, dass Sie für die Abstände der Präambel zur Teilüberschrift bzw. zum Kapiteltext selbst verantwortlich sind. Bitte beachten Sie auch, dass die `abstract`-Umgebung bei der Klasse scrbook nicht existiert (siehe Abschnitt 3.8, Seite 67).

v2.8p Das erste optionale Argument *Position* bestimmt über ein bis zwei Buchstaben die Position, an der die Präambel ausgegeben wird. Für die vertikale Position existieren derzeit zwei Möglichkeiten:

- `o` – über der Überschrift
- `u` – unter der Überschrift

Es kann jeweils eine Präambel unter und eine Präambel über der Überschrift gesetzt werden.

Für die horizontale Position existieren derzeit drei Möglichkeiten:

- `l` – linksbündig
- `r` – rechtsbündig
- `c` – zentriert

Dabei wird allerdings nicht der Text der *Präambel* entsprechend angeordnet, sondern eine Box, deren Breite durch das zweite optionale Argument *Breite* bestimmt wird. Wird auf das

zweite optionale Argument verzichtet, so wird stattdessen die gesamte Textbreite verwendet. Damit bleibt die Option zur horizontalen Positionierung in diesem Fall wirkungslos. Es kann jeweils ein Buchstabe für die vertikale mit einem Buchstaben für die horizontale Anordnung kombiniert werden.

Eine häufigere Anwendung von `\setchapterpreamble` dürfte ein Spruch oder Zitat zu einer Überschrift sein. Die dazu verwendbare Anweisung `\dictum` ist im nachfolgenden Abschnitt erläutert. Dort findet sich auch ein entsprechendes Beispiel.

Bitte beachten Sie, dass *Präambel* bei der Platzierung über der Überschrift in den dort vorhandenen freien Platz gesetzt wird. Die Überschrift wird dabei nicht nach unten verschoben. Der Anwender ist also selbst dafür verantwortlich dass *Präambel* nicht zu groß ist und der vorhandene freie Platz über der Überschrift genügt. Siehe dazu auch `\chapterheadstartvskip` in Abschnitt 16.3 auf Seite 310.

3.17. Schlauer Spruch

Ein häufiger anzutreffendes Element ist ein Zitat oder eine Redewendung, die rechtsbündig unter oder über einer Überschrift gesetzt wird. Dabei werden der Spruch selbst und der Quellennachweis in der Regel speziell formatiert.

```
\dictum[Urheber]{Spruch}
\dictumwidth
\dictumauthorformat{Urheber}
\dictumrule
\raggeddictum
\raggeddictumtext
\raggeddictumauthor
```

Ein solcher Spruch kann mit Hilfe der Anweisung `\dictum` gesetzt werden. Bei KOMA-Script-Klassen wird für Kapitel oder Teile empfohlen, `\dictum` als obligatorisches Argument der Anweisung `\setchapterpreamble` beziehungsweise `\setpartpreamble` (siehe Abschnitt 3.16, Seite 105) zu verwenden. Dies ist jedoch nicht zwingend.

Der Spruch wird zusammen mit einem optional anzugebenden *Urheber* in einer `\parbox` (siehe [Tea05b]) der Breite `\dictumwidth` gesetzt. Dabei ist `\dictumwidth` keine Länge, die mit `\setlength` gesetzt wird. Es handelt sich um ein Makro, das mit `\renewcommand` umdefiniert werden kann. Vordefiniert ist `0.3333\textwidth`, also ein Drittel der jeweiligen Textbreite. Die Box selbst wird mit der Anweisung `\raggeddictum` ausgerichtet. Voreingestellt ist dabei `\raggedleft`, also rechtsbündig. `\raggeddictum` kann mit Hilfe von `\renewcommand` umdefiniert werden.

Innerhalb der Box wird der *Spruch* mit `\raggeddictumtext` angeordnet. Voreingestellt ist hier `\raggedright`, also linksbündig. Eine Umdefinierung ist auch hier mit `\renewcommand` möglich. Die Ausgabe erfolgt in der für Element `dictumtext` eingestellten Schriftart, die mit

Tabelle 3.16.: Voreinstellungen der Schrift für die Elemente des Spruchs

Element	Voreinstellung
`dictumtext`	`\normalfont\normalcolor\sffamily\small`
`dictumauthor`	`\itshape`

den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) geändert werden kann. Die Voreinstellung entnehmen Sie bitte Tabelle 3.16.

Ist ein *Urheber* angegeben, so wird dieser mit einer Linie über die gesamte Breite der `\parbox` vom *Spruch* abgetrennt. Diese Linie ist in `\dictumrule` definiert. Es handelt sich dabei um ein vertikales Objekt, das mit

v3.10

```
\newcommand*{\dictumrule}{\vskip-1ex\hrulefill\par}
```

vordefiniert ist.

Mit `\raggeddictumauthor` wird die Ausrichtung für die Linie und den Urheber vorgenommen. Voreingestellt ist `\raggedleft`. Auch diese Anweisung kann mit `\renewcommand` umdefiniert werden. Die Ausgabe erfolgt in der Form, die mit `\dictumauthorformat` festgelegt ist. Das Makro erwartet schlicht den `\Urheber` als Argument. In der Voreinstellung ist `\dictumauthorformat` als

```
\newcommand*{\dictumauthorformat}[1]{(#1)}
```

definiert. Der *Urheber* wird also in runde Klammern gesetzt. Für das Element `dictumauthor` kann dabei eine Abweichung der Schrift von der des Elementes `dictumtext` definiert werden. Die Voreinstellung entnehmen Sie bitte Tabelle 3.16. Eine Änderung ist mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) möglich.

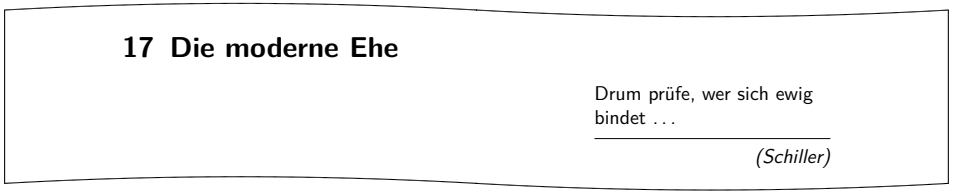
Wird `\dictum` innerhalb der Anweisung `\setchapterpreamble` oder `\setpartpreamble` (siehe Abschnitt 3.16, Seite 105) verwendet, so ist Folgendes zu beachten: Die horizontale Anordnung erfolgt immer mit `\raggeddictum`. Das optionale Argument zur horizontalen Anordnung, das die beiden Anweisungen vorsehen, bleibt daher ohne Wirkung. `\textwidth` ist nicht die Breite des gesamten Textkörpers, sondern wie bei `minipage` die aktuelle Textbreite. Ist also die Breite `\dictumwidth` als `.5\textwidth` definiert und bei `\setchapterpreamble` wird als optionales Argument für die Breite ebenfalls `.5\textwidth` angegeben, so erfolgt die Ausgabe in einer Box, deren Breite ein Viertel der Breite des Textkörpers ist. Es wird empfohlen, bei Verwendung von `\dictum` auf die optionale Angabe einer Breite bei `\setchapterpreamble` oder `\setpartpreamble` zu verzichten.

Sollen mehrere schlaue Sprüche untereinander gesetzt werden, so sollten diese durch einen zusätzlichen Abstand vertikal voneinander abgesetzt werden. Ein solcher kann leicht mit der Anweisung `\bigskip` gesetzt werden.

Beispiel: Sie schreiben ein Kapitel über die moderne Ehe. Dabei wollen sie in der Präambel zur Kapitelüberschrift einen schlauen Spruch setzen. Dieser soll unter der Überschrift erscheinen. Also schreiben Sie:

```
\setchapterpreamble[u]{%
  \dictum[Schiller]{Drum prüfe,
    wer sich ewig bindet \dots}}
\chapter{Die moderne Ehe}
```

Die Ausgabe erfolgt dann in der Form:



Wenn Sie wollen, dass nicht ein Drittel, sondern nur ein Viertel der verfügbaren Textbreite für den Spruch verwendet wird, so definieren Sie `\dictumwidth` wie folgt um:

```
\renewcommand*{\dictumwidth}{.25\textwidth}
```

An dieser Stelle sei noch auf das Paket [Sch09] hingewiesen, mit dem man Flattersatz mit Trennung erreichen kann.

3.18. Listen

LaTeX und die Standardklassen bieten verschiedene Umgebungen für Listen. All diese Umgebungen bietet KOMA-Script selbstverständlich auch, teilweise jedoch mit leichten Abwandlungen oder Erweiterungen. Grundsätzlich gilt, dass Listen – auch unterschiedlicher Art – bis zu einer Tiefe von vier Listen geschachtelt werden können. Eine tiefere Schachtelung wäre auch aus typografischen Gründen kaum sinnvoll, da genau genommen schon mehr als drei Ebenen nicht mehr überblickt werden können. Ich empfehle in solchen Fällen, die eine große Liste in mehrere kleinere Listen aufzuteilen.

```
\begin{itemize}
  \item
  ...
\end{itemize}
\labelitemi
\labelitemii
\labelitemiii
\labelitemiv
```

Die einfachste Form einer Liste ist die Stichpunkt- oder `itemize`-Liste. Bei den KOMA-Script-Klassen werden je nach Ebene folgende Aufzählungszeichen zur Einleitung eines Listenelements verwendet: »•«, »–«, »∗« und »·«. Die Definition der Zeichen für die einzelnen Ebenen sind in den Makros `\labelitemi`, `\labelitemii`, `\labelitemiii` und `\labelitemiv` abgelegt. Sie

können diese leicht mit `\renewcommand` umdefinieren. Die einzelnen Stichpunkte werden mit `\item` eingeleitet.

Beispiel: Sie haben eine einfache Aufzählung, die in mehreren Ebenen geschachtelt ist. Sie schreiben beispielsweise:

```
\minisec{Die Fahrzeuge im Spiel}
\begin{itemize}
  \item Flugzeuge
  \begin{itemize}
    \item Doppeldecker
    \item Jets
    \item Transportmaschinen
    \begin{itemize}
      \item einmotorig
      \begin{itemize}
        \item{düsengetrieben}
        \item{propellergetrieben}
      \end{itemize}
      \item zweimotorig
      \begin{itemize}
        \item{düsengetrieben}
        \item{propellergetrieben}
      \end{itemize}
    \end{itemize}
    \item Drehflügler
  \end{itemize}
  \item Motorräder
  \begin{itemize}
    \item historisch korrekt
    \item futurisch nicht real
  \end{itemize}
  \item Automobile
    \begin{itemize}
    \item Rennwagen
    \item Personenwagen
    \item Lastwagen
    \end{itemize}
   \item Fahrräder
  \end{itemize}
\end{itemize}
```

Anschließend erhalten Sie:

Die Fahrzeuge im Spiel

- Flugzeuge
  - Doppeldecker
  - Jets
  - Transportmaschinen
    - einmotorig
      - düsengetrieben
      - propellergetrieben
    - zweimotorig
      - düsengetrieben
      - propellergetrieben
  - Drehflügler
- Motorräder
  - historisch korrekt
  - futurisch nicht real
- Automobile
  - Rennwagen
  - Personenwagen
  - Lastwagen
- Fahrräder

```
\begin{enumerate}
  \item
  ...
\end{enumerate}
\theenumi
\theenumii
\theenumiii
\theenumiv
\labelenumi
\labelenumii
\labelenumiii
\labelenumiv
```

Die nummerierte Liste ist ebenfalls sehr häufig zu finden und bereits vom LaTeX-Kern vorgesehen. Die Nummerierung erfolgt je nach Ebene in unterschiedlicher Art: mit arabischen Zahlen, mit Kleinbuchstaben, mit kleinen römischen Zahlen und mit Großbuchstaben. Die Art der Nummerierung wird dabei über die Makros `\theenumi` bis `\theenumiv` festgelegt. Das Format der Ausgabe wird von den Makros `\labelenumi` bis `\labelenumiv` bestimmt. Dabei folgt auf den Wert der zweiten Ebene, der in Kleinbuchstaben ausgegeben wird, eine runde Klammer, während die Werte aller anderen Ebenen von einem Punkt gefolgt werden. Die einzelnen Stichpunkte werden wieder mit `\item` eingeleitet.

Beispiel: Verkürzen wir das vorherige Beispiel und verwenden statt der `itemize`- eine `enumerate`-Umgebung:

Die Fahrzeuge im Spiel

1. Flugzeuge
   a) Doppeldecker
   b) Transportmaschinen
      i. einmotorig
         A. düsengetrieben
         B. propellergetrieben
      ii. mehrmotorig
2. Motorräder
   a) historisch korrekt
   b) futurisch nicht real

Innerhalb der Aufzählung können ganz normal mit `\label` Marken gesetzt werden, auf die dann mit `\ref` zugegriffen werden kann. So wurde oben hinter den düsengetriebenen, einmotorigen Flugzeugen mit »`\label{bsp:duesen}`« ein Label gesetzt. Der `\ref`-Wert ist dann »1(b)iA«.

```
\begin{description}
  \item[Stichwort]
  ...
\end{description}
```

v2.8p

Eine weitere Listenform ist die Stichwortliste. Sie dient in erster Linie der Beschreibung einzelner Begriffe. Diese werden als optionale Parameter bei `\item` angegeben. Die Schriftart, die für die Hervorhebung des Stichworts verwendet wird, kann außerdem bei den KOMA-Script-Klassen mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) für das Element `descriptionlabel` (siehe Tabelle 3.2, Seite 57) geändert werden. In der Voreinstellung wird `\sffamily\bfseries` verwendet.

Beispiel: Sie wollen, dass die Stichworte statt serifenlos und fett lediglich fett, aber in der Standardschriftart ausgegeben werden. Mit

```
\setkomafont{descriptionlabel}{\normalfont
  \bfseries}
```

definieren Sie daher die Schrift entsprechend um.

Ein Beispiel für die Ausgabe einer Stichwortliste ist eine Aufzählung der Seitenstile. Der Quelltext dazu lautet beispielsweise:

```
\begin{description}
  \item[empty] ist der Seitenstil, bei dem Kopf-
    und Fußzeile vollständig leer bleiben.
```

```
    \item[plain] ist der Seitenstil, bei dem
      keinerlei Kolumnentitel verwendet wird.
    \item[headings] ist der Seitenstil für
      automatische Kolumnentitel.
    \item[myheadings] ist der Seitenstil für
      manuelle Kolumnentitel.
  \end{description}
```

Diese ergibt:

empty ist der Seitenstil, bei dem Kopf- und Fußzeile vollständig leer bleiben.

plain ist der Seitenstil, bei dem keinerlei Kolumnentitel verwendet wird.

headings ist der Seitenstil für automatische Kolumnentitel.

myheadings ist der Seitenstil für manuelle Kolumnentitel.

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

Eine andere Form der Stichwortliste ist nur bei den KOMA-Script-Klassen vorhanden: die `labeling`-Umgebung. Im Unterschied zur zuvor vorgestellten Umgebung `description` kann bei `labeling` ein Muster angegeben werden, dessen Länge die Einrücktiefe bei allen Stichpunkten ergibt. Darüber hinaus kann zwischen Stichpunkt und Beschreibungstext ein optionales *Trennzeichen* festgelegt werden. Die Schriftart, die für die Hervorhebung des Schlüsselworts verwendet wird, kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) für das Element `labelinglabel` (siehe Tabelle 3.2, Seite 57) geändert werden. Für die davon abweichende Schriftart der Trennzeichen ist das Element `labelingseparator` (siehe ebenfalls Tabelle 3.2, Seite 57) zuständig.

v3.02

Beispiel: Wir schreiben das Beispiel der `description`-Umgebung etwas um:

```
\setkomafont{labelinglabel}{\ttfamily}
\setkomafont{labelingseparator}{\normalfont}
\begin{labeling}[~--]{myheadings}
  \item[empty]
    Seitenstil für leere Seiten ohne Kopf und Fuß
  \item[plain]
    Seitenstil für Kapitelanfänge ganz ohne
    Kolumnentitel
  \item[headings]
    Seitenstil für automatische Kolumnentitel
  \item[myheadings]
```

```
    Seitenstil für manuelle Kolumnentitel
\end{labeling}
```

Als Ergebnis erhalten wir dann:

> `empty` – Seitenstil für leere Seiten ohne Kopf und Fuß
>
> `plain` – Seitenstil für Kapitelanfänge ganz ohne Kolumnentitel
>
> `headings` – Seitenstil für automatische Kolumnentitel
>
> `myheadings` – Seitenstil für manuelle Kolumnentitel

Wie in diesem Beispiel zu sehen ist, kann eine eventuell geforderte Schriftumschaltung in bei KOMA-Script gewohnter Weise erreicht werden. Da sich die Schriftumschaltung für das Schlüsselwort aber auch auf die Trennzeichen auswirkt, kann es eventuell erforderlich sein, die Schriftumschaltung dafür explizit aufzuheben.

Gedacht war die Umgebung ursprünglich für Strukturen wie »Voraussetzung, Aussage, Beweis« oder »Gegeben, Gesucht, Lösung«, wie man sie in Vorlesungsskripten häufiger findet. Inzwischen findet die Umgebung aber ganz unterschiedliche Anwendungen. So wurde die Umgebung für Beispiele in dieser Anleitung mit Hilfe der `labeling`-Umgebung definiert.

```
\begin{verse}
  ...
\end{verse}
```

Die `verse`-Umgebung wird normalerweise nicht als Listenumgebung wahrgenommen, da hier nicht mit `\item` gearbeitet wird. Stattdessen wird wie innerhalb der `flushleft`-Umgebung mit festen Zeilenumbrüchen gearbeitet. Intern handelt es sich jedoch sowohl bei den Standardklassen als auch bei KOMA-Script durchaus um eine Listenumgebung.

Die `verse`-Umgebung findet hauptsächlich für Gedichte Anwendung. Dabei werden die Zeilen links und rechts eingezogen. Einzelne Verse werden mit einem festen Zeilenumbruch, also mit `\\` beendet. Strophen werden ganz normal als Absatz gesetzt, also durch eine Leerzeile getrennt. Häufig findet stattdessen auch `\medskip` oder `\bigskip` Verwendung. Will man verhindern, dass am Ende eines Verses ein Seitenumbruch erfolgt, so verwendet man ganz normal `\\*` an Stelle von `\\`.

Beispiel: Als Beispiel ein kurzes Gedicht von Wilhelm Busch:

```
\begin{verse}
  Wenn einer, der mit Mühe kaum\\*
  Gekrochen ist auf einen Baum,\\*
  Schon meint, dass er ein Vogel wär,\\*
  So irrt sich der.
\end{verse}
```

Mit dem Resultat:

> Wenn einer, der mit Mühe kaum
> Gekrochen ist auf einen Baum,
> Schon meint, dass er ein Vogel wär,
> So irrt sich der.

Bei einem sehr langen Vers wie:

```
\begin{verse}
  Der Philosoph wie der Hausbesitzer hat
  immer Reparaturen.\\*
  \bigskip
  Wer dir sagt, er hätte noch nie gelogen,
  dem traue nicht, mein Sohn.
\end{verse}
```

bei dem ein Zeilenumbruch innerhalb des Verses erfolgt:

> Der Philosoph wie der Hausbesitzer hat immer Reparaturen.
>
> Wer dir sagt, er hätte noch nie gelogen, dem traue nicht, mein
> Sohn.

kann mit `\\*` allerdings nicht verhindert werden, dass am Zeilenumbruch auch ein Seitenumbruch erfolgt. Um dies zu erreichen, müsste innerhalb der ersten Zeile zusätzlich ein `\nopagebreak` eingefügt werden:

```
\begin{verse}
  Der Philosoph wie der Hausbesitzer\nopagebreak{}
  hat immer Reparaturen.\\
  \bigskip
  Wer dir sagt, er hätte noch nie\nopagebreak{}
  gelogen, dem traue nicht, mein Sohn.
\end{verse}
```

Hier noch zwei Sprüche, die man immer bedenken sollte, wenn man mit scheinbar seltsamen Fragen zu LaTeX oder den dazugehörigen Antworten konfrontiert ist:

```
\begin{verse}
  Wir mögen's keinem gerne gönnen,\\*
  Dass er was kann, was wir nicht können.\\
  \bigskip
  Wie klein ist das, was einer ist,\\*
  Wenn man's mit seinem Dünkel misst.
\end{verse}
```

> Wir mögen's keinem gerne gönnen,
> Dass er was kann, was wir nicht können.
>
> Wie klein ist das, was einer ist,
> Wenn man's mit seinem Dünkel misst.

In diesen Beispielen wurde übrigens jeweils `\bigskip` verwendet, um zwei Sprüche voneinander zu trennen.

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

Diese beiden Umgebungen sind intern ebenfalls Listenumgebungen und sowohl bei den Standardklassen als auch bei KOMA-Script zu finden. Beide Umgebungen setzen Blocksatz, der rechts und links eingezogen ist. Verwendet werden die Umgebungen häufig, um längere Zitate abzusetzen. Der Unterschied zwischen beiden liegt in der Art und Weise, wie Absätze abgesetzt werden. Während bei `quote` Absätze durch vertikalen Abstand gekennzeichnet werden, wird bei `quotation` mit horizontalem Einzug der ersten Zeile eines Absatzes gearbeitet. Dies gilt auch für den ersten Absatz einer `quotation`-Umgebung, außer es wird `\noindent` vorangestellt.

Beispiel: Sie wollen eine kleine Anekdote hervorheben. Also schreiben Sie folgende `quotation`-Umgebung:

```
Ein kleines Beispiel für eine Anekdote, die sich
einst in Schwaben zugetragen haben soll:
\begin{quotation}
  Es klingelt an der Tür eines Pfarrhauses in
  Stuttgart. Als die Haushälterin öffnet, steht
  ein unrasierter Mann in reichlich schäbigem
  Mantel vor der Tür und hält seine Strickmütze
  in der Hand.

  "‘Gute Frau,"’ verkündet der Mann in gequältem
  Ton, doch bestem Hochdeutsch,
  "‘ich habe seit drei Tagen nichts mehr
  gegessen."’

  Die Frau schüttelt mitleidig den Kopf und
  entgegnet im Brustton vollster Überzeugung:

  "‘Guda Moh, Sie missat sich halt zwinga!"’
\end{quotation}
```

Das Ergebnis ist dann:

Ein kleines Beispiel für eine Anekdote, die sich einst in Schwaben zugetragen haben soll:

> Es klingelt an der Tür eines Pfarrhauses in Stuttgart. Als die Haushälterin öffnet, steht ein unrasierter Mann in reichlich schäbigem Mantel vor der Tür und hält seine Strickmütze in der Hand.
>
> „Gute Frau," verkündet der Mann in gequältem Ton, doch bestem Hochdeutsch, „ich habe seit drei Tagen nichts mehr gegessen."
>
> Die Frau schüttelt mitleidig den Kopf und entgegnet im Brustton vollster Überzeugung:
>
> „Guda Moh, Sie missat sich halt zwinga!"

Zum Vergleich sei das Ganze an Stelle von `quotation` auch noch mit einer `quote`-Umgebung gezeigt:

Ein kleines Beispiel für eine Anekdote, die sich einst in Schwaben zugetragen haben soll:

> Es klingelt an der Tür eines Pfarrhauses in Stuttgart. Als die Haushälterin öffnet, steht ein unrasierter Mann in reichlich schäbigem Mantel vor der Tür und hält seine Strickmütze in der Hand.
>
> „Gute Frau," verkündet der Mann in gequältem Ton, doch bestem Hochdeutsch, „ich habe seit drei Tagen nichts mehr gegessen."
>
> Die Frau schüttelt mitleidig den Kopf und entgegnet im Brustton vollster Überzeugung:
>
> „Guda Moh, Sie missat sich halt zwinga!"

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

Wie bei `quote` und `quotation` handelt es sich bei `addmargin` um eine Umgebung, die den Rand verändert. Im Unterschied zu den beiden erstgenannten Umgebungen kann der Anwender jedoch bei `addmargin` wählen, um welchen Wert der Rand verändert werden kann. Des Weiteren verändert die Umgebung den Absatzeinzug und den Absatzabstand nicht. Es wird auch kein zusätzlicher vertikaler Abstand vor und nach der Umgebung eingefügt.

Ist nur das obligatorische Argument *Einzug* angegeben, so wird der Inhalt der Umgebung rechts und links um diesen Wert eingezogen. Ist das optionale Argument *linker Einzug* hingegen angegeben, so wird links abweichend von *Einzug* der Wert *linker Einzug* zum Rand addiert.

Die Sternvariante `addmargin*` unterscheidet sich nur im doppelseitigen Satz von der Variante ohne Stern, wobei der Unterschied auch nur dann auftritt, wenn das optionale Argument *innerer Einzug* verwendet wird. Dabei wird dann der Wert von *innerer Einzug* zum inneren Randanteil der Seite addiert. Dies ist bei rechten Seiten der linke Rand der Seite, bei linken Seiten jedoch der rechte Rand der Seite. *Einzug* gilt dann für den jeweils anderen Rand.

Bei beiden Varianten der Umgebung sind für alle Parameter auch negative Werte erlaubt. Damit kann man erreichen, dass die Umgebung in den Rand hineinragt.

Beispiel: Angenommen, Sie schreiben eine Anleitung mit kurzen Quellcode-Beispielen. Um diese sehr stark hervorzuheben, sollen sie mit horizontalen Linien vom Text abgesetzt und leicht in den äußeren Rand verschoben werden. Sie definieren sich dafür zunächst eine Umgebung:

```
\newenvironment{QuellcodeRahmen}{%
  \begin{addmargin*}[1em]{-1em}%
    \begin{minipage}{\linewidth}%
      \rule{\linewidth}{2pt}%
}{%
    \rule[.25\baselineskip]{\linewidth}{2pt}%
    \end{minipage}%
  \end{addmargin*}%
}
```

Zur Demonstration sei die Definition der Umgebung in der Umgebung selbst gesetzt:

```
\newenvironment{\QuellcodeRahmen}{%
  \begin{addmargin*}[1em]{-1em}%
    \begin{minipage}{\linewidth}%
      \rule{\linewidth}{2pt}%
}{%
    \rule[.25\baselineskip]{\linewidth}{2pt}%
    \end{minipage}%
  \end{addmargin*}%
}
```

Nicht zwingend schön, aber es zeigt den Nutzen.

Das optionale Argument der `addmargin*`-Umgebung sorgt dafür, dass der innere Rand um den Wert 1 em vergrößert wird. Dafür wird der äußere Rand um den negativen Wert vergrößert, also in Wirklichkeit um 1 em verkleinert. Dies resultiert in einer Verschiebung um 1 em nach außen. Selbstverständlich kann statt `1em` auch eine Länge, beispielsweise `2\parindent`, verwendet werden.

Ob eine Seite eine linke oder eine rechte Seite ist, kann übrigens beim ersten LaTeX-Durchlauf nicht zuverlässig festgestellt werden. Siehe dazu die Erklärungen zu den Anweisungen `\ifthispageodd` (Abschnitt 3.11, Seite 74) und `\ifthispagewasodd` (Abschnitt 16.1, Seite 307).

Im Zusammenspiel von Umgebungen wie Listen mit Absätzen ergeben sich für Laien häufiger Fragen. Daher widmet sich die weiterführende Erklärung zu Option `parskip` in Abschnitt 16.1, Seite 307 auch diesem Thema. Ebenfalls im Expertenteil in Abschnitt 16.1, Seite 307 wird die Problematik von mehrseitigen `addmargin*`-Umgebungen behandelt.

3.19. Mathematik

Die KOMA-Script-Klassen stellen keine eigenen Umgebungen für mathematische Formeln, Gleichungssysteme oder ähnliche Elemente der Mathematik bereit. Stattdessen stützt sich KOMA-Script im Bereich der Mathematik voll und ganz auf den LaTeX-Kern. Dies gilt auch für die Realisierung der beiden Optionen `leqno` und `fleqn`.

Auf eine Beschreibung der Mathematikumgebungen des LaTeX-Kerns, also `displaymath`, `equation` und `eqnarray`, wird an dieser Stelle verzichtet. Wer diese verwenden möchte, sei auf [SKPH99] verwiesen. Für mehr als nur einfachste mathematische Formeln und Gleichungen sei jedoch die Verwendung von Paket amsmath empfohlen (siehe [Ame02]).

`leqno`

Gleichungen werden normalerweise auf der rechten Seite nummeriert. Mit Hilfe der Standardoption `leqno` wird die Standardoptionsdatei `leqno.clo` geladen. Dadurch erfolgt die Nummerierung von Gleichungen links. Diese Option ist als optionales Argument von `\documentclass` zu verwenden. Eine Einstellung per `\KOMAoptions` oder `\KOMAoption` wird nicht unterstützt. Dies wäre auch deshalb nicht sinnvoll, weil das für den Mathematiksatz empfohlene Paket amsmath ebenfalls nur beim Laden, nicht aber zur Laufzeit auf diese Option reagieren kann.

`fleqn`

Gleichungen werden normalerweise horizontal zentriert ausgegeben. Mit Hilfe der Standardoption `fleqn` wird die Standardoptionsdatei `fleqn.clo` geladen. Dadurch erfolgt die Ausgabe von Gleichungen linksbündig. Diese Option ist als optionales Argument von `\documentclass` zu verwenden. Eine Einstellung per `\KOMAoptions` oder `\KOMAoption` wird nicht unterstützt. Dies wäre auch deshalb nicht sinnvoll, weil das für den Mathematiksatz empfohlene Paket amsmath ebenfalls nur beim Laden, nicht aber zur Laufzeit auf diese Option reagieren kann.

3.20. Gleitumgebungen für Tabellen und Abbildungen

LaTeX bietet mit den Gleitumgebungen einen sehr leistungsfähigen und komfortablen Mechanismus für die automatische Anordnung von Abbildungen und Tabellen. Genau genommen sollte von »Tafeln« statt von »Tabellen« die Rede sein. Dies wäre auch zur Unterscheidung der Umgebungen `table` und `tabular` von Vorteil. Es hat sich im Deutschen aber für beides

die Bezeichnung »Tabelle« eingebürgert. Das kommt vermutlich daher, dass man in `table`-Umgebungen üblicherweise `tabular`-Umgebungen setzt, auch wenn dies keineswegs zwingend ist.

Häufig werden die Gleitumgebungen von Anfängern nicht richtig verstanden. So wird oft die Forderung aufgestellt, eine Tabelle oder Abbildung genau an einer bestimmten Position im Text zu setzen. Dies ist jedoch nicht erforderlich, da auf Gleitumgebungen im Text über eine Referenz verwiesen wird. Es ist auch nicht sinnvoll, da ein solches Objekt an einer Stelle nur dann gesetzt werden kann, wenn auf der Seite noch genügend Platz für das Objekt vorhanden ist. Ist dies nicht der Fall, müsste das Objekt auf die nächste Seite umbrochen werden und auf der aktuellen Seite würde ein möglicherweise sehr großer leerer Raum bleiben.

Häufig findet sich in einem Dokument auch bei jedem Gleitobjekt das gleiche optionale Argument zur Platzierung des Objekts. Auch dies ist nicht sinnvoll. In solchen Fällen sollte man besser den Standardwert global ändern. Näheres dazu ist [Wik] zu entnehmen.

Ein wichtiger Hinweis sei diesem Abschnitt noch vorangestellt: Die meisten Mechanismen, die hier vorgestellt werden und über die Fähigkeiten der Standardklassen hinaus gehen, funktionieren nicht mehr, wenn Sie ein Paket verwenden, das in die Generierung von Tabellen- und Abbildungstiteln eingreift und deren Aussehen verändert. Dies sollte selbstverständlich sein, wird aber leider häufig nicht bedacht.

`captions=`*`Einstellung`*

Bei den Standardklassen werden die Titel von Gleitumgebungen, die mit der Anweisung `\caption` (siehe unten) gesetzt werden, als Unterschrift formatiert. Darüber hinaus wird zwischen ein- und mehrzeiligen Unterschriften unterschieden, wobei einzeilige Unterschriften horizontal zentriert werden.

Tabellen werden jedoch häufig mit Überschriften statt mit Unterschriften versehen. Das liegt daran, dass es auch Tabellen geben kann, die sich über mehrere Seiten erstrecken. Bei solchen Tabellen möchte man natürlich bereits auf der ersten Seite wissen, worum es sich handelt. Darüber hinaus werden Tabellen zeilenweise von oben nach unten gelesen. Damit gibt es mindestens zwei gute Gründe, in der Regel alle Tabellen mit Überschriften zu versehen. KOMA-Script bietet daher mit der Einstellung `captions=tableheading` die Möglichkeit, bei Tabellen die Formatierung auf Überschriften zu ändern.

Es sei an dieser Stelle darauf hingewiesen, dass mehrseitige Tabellen nicht als Gleitumgebungen gesetzt werden können. Für den automatischen Seitenumbruch von Tabellen werden außerdem Zusatzpakete wie longtable (siehe [Car04]) oder tabu (siehe [Che11]) benötigt.

Mit der Einstellung `captions=tablesignature` wird wieder die Voreinstellung der Formatierung als Tabellenunterschrift gewählt. Es sei an dieser Stelle darauf hingewiesen, dass die beiden Werte lediglich die Formatierung ändern. Der Ort, an dem die Über- oder Unterschrift gesetzt wird, hängt bei den Gleitumgebungen von KOMA-Script allein vom Ort ab, an dem die Anweisung `\caption` verwendet wird. Dies ändert sich jedoch bei Verwendung des Pakets float mit der Anweisung `\restylefloats` (siehe [Lin01]).

v3.09 Natürlich existiert mit den Optionen `captions=figureheading` und `captions=figuresignature` auch die entsprechende Funktion für Abbildungen. Allerdings werden Abbildungen in der Regel wie im Falle von Fotos eher als Ganzes und im Falle von Diagrammen oder Graphen eher von unten links her betrachtet. In den wenigsten Fällen dürfte es daher sinnvoll sein, nur bei Abbildungen die Formatierung von Unterschriften in Überschriften zu ändern.

Manchmal wird jedoch gewünscht, alle Gleitumgebungen mit Überschriften zu versehen. Daher gibt es bei KOMA-Script die Einstellung Optionen `captions=heading` und die umgekehrte Einstellung `captions=signature`, mit der man die Formatierungen aller Gleitumgebungen entsprechend ändern kann. Diese entfalten ihre Wirkung auch noch, wenn sie innerhalb einer Gleitumgebung verwendet werden.

float Bitte beachten Sie, dass bei Verwendung des float-Pakets die Einstellungen von Unterschriften oder Überschriften nicht mehr funktionieren, sobald Sie `\restylefloat` auf Tabellen oder Abbildungen anwenden. Näheres zum float-Paket und `\restylefloat` entnehmen Sie bitte [Lin01]. Die zusätzliche Unterstützung, die KOMA-Script bei Verwendung des float-Pakets bietet, finden Sie bei der Erklärung zu `komaabove` (siehe Seite 130).

Bei KOMA-Script besteht darüber hinaus auch die Möglichkeit, mit der Einstellung `captions=nooneline` die Unterscheidung zwischen ein- und mehrzeiligen Über- bzw. Unterschriften abzuschalten. Dies ist beispielsweise dann wichtig, wenn man keine Zentrierung wünscht. Die Voreinstellung, bei der einzeilige Über- und Unterschriften automatisch horizontal zentriert werden, entspricht der Einstellung `captions=oneline`.

Als Besonderheit bietet KOMA-Script innerhalb von Gleitumgebungen die Möglichkeit, den Titel statt als Über- oder Unterschrift neben den eigentlichen Inhalt zu setzen. Die dazu notwendige Umgebung `captionsbeside` ist ab Seite 127 erklärt. Die Einstellungen für diese Umgebung können ebenfalls mit der Option `captions` geändert werden. Die dabei möglichen Werte für die jeweilige *`Einstellung`* sind Tabelle 3.17 zu entnehmen.

Tabelle 3.17.: Mögliche Werte für Option `captions` zur Einstellung der Formatierung von Über- und Unterschriften in Gleitumgebungen

`bottombeside`, `besidebottom`	Gleitumgebungstitel der Umgebung `captionsbeside` (siehe Abschnitt 3.20, Seite 127) werden mit der untersten Grundlinie auf Höhe der untersten Grundlinie des Inhalts der Gleitumgebung ausgerichtet.
`centeredbeside`, `besidecentered`, `middlebeside`, `besidemiddle`	Gleitumgebungstitel der Umgebung `captionsbeside` (siehe Abschnitt 3.20, Seite 127) werden zum Inhalt der Gleitumgebung vertikal zentriert ausgerichtet.

...

Tabelle 3.17.: Mögliche Werte für Option `captions` (*Fortsetzung*)

`figureheading`, `figureabove`, `abovefigure`, `topatfigure`

v3.09

Bei Abbildungen wird (ggf. abweichend von `captions=signature`) die Formatierung als Überschrift gewählt.

`figuresignature`, `belowfigure`, `bottomatfiggure`

v3.09

Bei Abbildungen wird (ggf. abweichend von `captions=heading`) die Formatierung als Unterschrift gewählt.

`heading`, `above`, `top`

v3.09

Gleitumgebungstitel werden als Überschriften formatiert. Dies hat jedoch keinen Einfluss darauf ob sie über oder unter der Gleitumgebung platziert werden. Die Option impliziert auch `captions=tableheading` und `captions=figureheading`.

`innerbeside`, `besideinner`

Gleitumgebungstitel der Umgebung `captionsbeside` (siehe Abschnitt 3.20, Seite 127) werden bei doppelseitigem Satz in der Voreinstellung innen neben dem Inhalt der Gleitumgebung gesetzt. Dies entspricht im einseitigen Satz `captions=leftbeside`.

`leftbeside`, `besideleft`

Gleitumgebungstitel der Umgebung `captionsbeside` (siehe Abschnitt 3.20, Seite 127) werden in der Voreinstellung links neben dem Inhalt der Gleitumgebung gesetzt.

`nooneline`

Einzeilige Über- und Unterschriften von Gleitumgebungen werden nicht gesondert, sondern genau wie mehrzeilige behandelt.

`oneline`

Einzeilige Über- und Unterschriften von Gleitumgebungen werden gesondert behandelt, um sie horizontal zu zentrieren.

`outerbeside`, `besideouter`

Gleitumgebungstitel der Umgebung `captionsbeside` (siehe Abschnitt 3.20, Seite 127) werden bei doppelseitigem Satz in der Voreinstellung außen neben dem Inhalt der Gleitumgebung gesetzt. Dies entspricht im einseitigen Satz `captions=rightbeside`.

...

Tabelle 3.17.: Mögliche Werte für Option `captions` (*Fortsetzung*)

`rightbeside`, `besideright`
: Gleitumgebungstitel der Umgebung `captionsbeside` (siehe Abschnitt 3.20, Seite 127) werden in der Voreinstellung rechts neben dem Inhalt der Gleitumgebung gesetzt.

`signature`, `below`, `bot`, `bottom`
v3.09
: Gleitumgebungstitel werden als Unterschriften formatiert. Dies hat jedoch keinen Einfluss darauf ob sie über oder unter der Gleitumgebung platziert werden. Die Option impliziert auch `captions=tablesignature` und `captions=figuresignature`.

`tableheading`, `tableabove`, `abovetable`, `abovetabular`, `topattable`
: Bei Tabellen wird (ggf. abweichend von `captions=signature`) die Formatierung als Überschrift gewählt.

`tablesignature`, `belowtable`, `belowtabular`, `bottomattable`
: Bei Tabellen wird (ggf. abweichend von `captions=heading`) die Formatierung als Unterschrift gewählt.

`topbeside`, `besidetop`
: Gleitumgebungstitel der Umgebung `captionsbeside` (siehe Abschnitt 3.20, Seite 127) werden mit der obersten Grundlinie auf Höhe der obersten Grundlinie des Inhalts der Gleitumgebung ausgerichtet.

```
\caption[Verzeichniseintrag]{Titel}
\captionbelow[Verzeichniseintrag]{Titel}
\captionabove[Verzeichniseintrag]{Titel}
```

Tabellen und Abbildungen werden bei den Standardklassen mit Hilfe der Anweisung `\caption` mit einem *Titel* in Form einer Unterschrift versehen. Bei Abbildungen ist dies grundsätzlich korrekt. Bei Tabellen wird gestritten, ob der *Titel* als Überschrift über oder konsistent mit der Bildunterschrift unter die Tabelle gehört. Daher bietet KOMA-Script im Gegensatz zu den Standardklassen die Anweisungen `\captionbelow` für *Titel* in Form von Unterschriften und `\captionabove` für *Titel* in Form von Überschriften.

Sowohl bei Tabellen als auch bei Abbildungen oder generell für alle Gleitumgebungen lässt sich das Verhalten von `\caption` mit der Option `captions` steuern, die am Anfang dieses Abschnitts zu finden ist. Aus Gründen der Kompatibilität ist voreingestellt, dass sich `\caption` bei allen Gleitumgebungen wie `\captionbelow` verhält. Es wird jedoch empfohlen, Tabellenüberschriften zu verwenden und auf die Formatierung mit `captions=tableheading` entsprechend umzustellen oder bei Tabellen auf `\captionabove` zurück zu greifen.

Beispiel: Sie wollen mit Tabellenüberschriften statt mit Tabellenunterschriften arbeiten, weil Sie teilweise Tabellen haben, die über mehr als eine Seite gehen. Mit den Standardklassen bliebe Ihnen nur die Möglichkeit:

```
\begin{table}
  \caption{Dies ist nur eine Beispieltabelle}
  \begin{tabular}{llll}
    Dies & ist & ein & Beispiel.\\\hline
    Bitte & lassen & Sie & den \\
    Inhalt & dieser & Tabelle & unbeachtet.
  \end{tabular}
\end{table}
```

Damit hätten Sie das unschöne Ergebnis:

Tabelle 30.2: Dies ist nur eine Beispieltabelle

Dies	ist	ein	Beispiel.
Bitte	lassen	Sie	den
Inhalt	dieser	Tabelle	unbeachtet.

Bei KOMA-Script schreiben Sie hingegen:

```
\begin{table}
  \captionabove{Dies ist nur eine Beispieltabelle}
  \begin{tabular}{llll}
    Dies & ist & ein & Beispiel.\\\hline
    Bitte & lassen & Sie & den \\
    Inhalt & dieser & Tabelle & unbeachtet.
  \end{tabular}
\end{table}
```

Sie erhalten dann das gewünschte Ergebnis:

Tabelle 30.2: Dies ist nur eine Beispieltabelle

Dies	ist	ein	Beispiel.
Bitte	lassen	Sie	den
Inhalt	dieser	Tabelle	unbeachtet.

Da Sie konsequent nicht nur eine, sondern alle Tabellen mit Überschriften versehen, können Sie stattdessen auch die Option `captions=tableheading` setzen (siehe Seite 120). Dann genügt es, wenn Sie wie bei den Standardklassen `\caption` verwenden. Sie erhalten trotzdem das Ergebnis von `\captionabove`.

v2.8p Die Schriftart für die Beschreibung und das Label – »Abbildung« oder »Tabelle« gefolgt von der Nummer und einem Trennzeichen – kann über die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) verändert werden. Zuständig sind hier die Elemente `caption` und `captionlabel` (siehe Tabelle 3.2, Seite 57). Dabei wird die Schriftart für

Tabelle 3.18.: Voreinstellungen der Schrift für die Elemente der Tabellen- oder Abbildungsunterschrift bzw. -überschrift

Element	Voreinstellung
`caption`	`\normalfont`
`captionlabel`	`\normalfont`

das Element `caption` zunächst auch auf das Element `captionlabel` angewandt, bevor dessen spezifische Schriftart Anwendung findet. Die Vorbelegungen sind Tabelle 3.18 zu entnehmen.

Beispiel: Sie wollen, dass Tabellen- und Abbildungsbeschreibungen in einer kleineren Schriftart gesetzt werden. Also setzen Sie beispielsweise in der Präambel Ihres Dokuments:

```
\addtokomafont{caption}{\small}
```

Außerdem hätten Sie gerne, dass das Label serifenlos und fett gedruckt wird. Sie setzen also außerdem:

```
\setkomafont{captionlabel}{\sffamily\bfseries}
```

Das Ergebnis wäre bei Verwendung von `\addkomafont` in diesem Fall übrigens gleich.

```
\captionof{Objekttyp}[Verzeichniseintrag]{Titel}
\captionaboveof{Objekttyp}[Verzeichniseintrag]{Titel}
\captionbelowof{Objekttyp}[Verzeichniseintrag]{Titel}
```

v3.05 Ähnlich wie die Pakete caption und capt-of bietet auch KOMA-Script die Anweisung `\captionof` mit der man auch außerhalb einer Gleitumgebung oder in einer fremden Gleitumgebung einen entsprechenden Titel mit Eintrag in das jeweilige Verzeichnis setzen kann. Dabei muss im Gegensatz zu `\caption` die Art des Gleitobjekts als zusätzliches erstes Argument angegeben werden.

v3.09 Darüber hinaus bietet KOMA-Script zusätzlich auch die Anweisungen `\captionaboveof` und `\captionbelowof`. Diese dienen als Gegenstücke zu `\captionabove` und `\captionbelow`.

v3.09a Selbstverständlich berücksichtigt `\captionof` auch die Einstellungen von Option `captions` bezüglich der Formatierung des Titels als Über- oder Unterschrift. Diese Fähigkeit geht jedoch eventuell durch das Laden von Paketen wie capt-of oder caption verloren. Bei Verwendung von caption ist die Anleitung zu diesem Paket zu beachten!

Beispiel: Angenommen, Sie wollen ein Gleitobjekt erstellen, bei dem eine Tabelle und eine Abbildung nebeneinander stehen. Da es keine gemischten Gleitobjekte gibt, verwenden Sie primär eine `figure`-Umgebung:

```
\begin{figure}
  \begin{minipage}{.5\linewidth}
    \centering
    \rule{4cm}{3cm}
    \caption{Ein Rechteck}\label{fig:rechteck}
```

Tabelle 3.19.: Rechteckmaße

Breite:	4 cm
Höhe:	3 cm

Abbildung 3.2.: Ein Rechteck

Abbildung 3.3.: Verwendung von `\captionaboveof` innerhalb einer fremden Gleitumgebung

```
    \end{minipage}%
    \begin{minipage}{.5\linewidth}
      \centering
      \captionaboveof{table}
      [Maße des Rechtecks aus
        Abbildung~\ref{fig:rechteck}]%
      {Maße des Rechtecks}
      \label{tab:rechteck}
      \begin{tabular}{ll}
        Breite: & 4\,cm\\
        Höhe:   & 3\,cm
      \end{tabular}
    \end{minipage}
  \end{figure}
```

Um Abbildung und Tabelle nebeneinander zu setzen, wurden zwei `minipage`-Umgebungen verwendet. Wichtig ist hier das Prozentzeichen nach der ersten `minipage`, ohne das ein zusätzlicher Wortabstand zwischen die beiden `minipage`-Umgebungen gesetzt würde.

Die Abbildungsunterschrift wurde mit `\caption` gesetzt. Für die Tabellenüberschrift wurde `\captionaboveof` verwendet. Als erstes Argument wurde `table` angegeben. Dadurch weiß KOMA-Script, dass es sich trotz `figure`-Umgebung um eine Tabellenüberschrift handelt.

Das optionale Argument von `\captionaboveof` setzt den Eintrag in das Tabellenverzeichnis. Ohne das optionale Argument würde der als letztes Argument angegebene Titel ebenfalls in das Tabellenverzeichnis geschrieben. Während dieser Titel im Gleitobjekt selbst völlig ausreichend ist, wäre er jedoch im Tabellenverzeichnis wenig aussagekräftig. Daher wird hier für das Verzeichnis ein abweichender Titel über das optionale Argument verwendet. Das Ergebnis der Bemühungen zeigt Abbildung 3.3.

In gleicher Weise, wie in obigem Beispiel eine Tabelle innerhalb einer Abbildungsumgebung

gesetzt und mit einem Titel versehen wird, könnte man auch eine nicht gleitende Tabelle außerhalb jeder Gleitumgebung setzen. Auch dabei sollte in der Regel eine `minipage` verwendet werden, um zu verhindern, dass zwischen Überschrift und Tabelle ein Seitenumbruch erfolgen kann. Zusätzlich sollte man die `minipage` dann noch in eine `flushleft`-Umgebung einbetten, um einerseits einen gefälligen Abstand zum Text davor und dahinter zu erreichen, und andererseits den Absatzeinzug vor der `minipage` zu verhindern.

```
\begin{captionbeside}[Verzeichnistitel]{Titel}%
                     [Anordnung][Breite][Offset]
  ...
\end{captionbeside}
\begin{captionbeside}[Verzeichnistitel]{Titel}%
                     [Anordnung][Breite][Offset]*
  ...
\end{captionbeside}
```

v2.8q Neben den Unter- und Überschriften findet man insbesondere bei kleineren Abbildungen häufiger Beschreibungen, die neben der Abbildung gesetzt werden. Dabei schließt normalerweise die Unterkante der Beschreibung mit der Unterkante der Abbildung ab. Natürlich kann man mit etwas Geschick und beispielsweise zwei `\parbox`-Anweisungen dergleichen auch in den Standardklassen erreichen. KOMA-Script bietet jedoch eine spezielle Umgebung. Diese Umgebung kann innerhalb der Gleitumgebungen verwendet werden. Der erste optionale Parameter *Verzeichnistitel* und der obligatorische Parameter *Titel* entsprechen dabei genau den gleichnamigen Parametern von `\caption`, `\captionabove` oder `\captionbelow`. Der *Titel* wird dabei neben den Inhalt der Umgebung gesetzt.

Ob der *Titel* rechts oder links daneben gesetzt wird, kann mit dem optionalen Parameter *Anordnung* bestimmt werden. Es darf genau einer der folgenden Buchstaben angegeben werden:

- `l` – links
- `r` – rechts
- `i` – innen: auf rechten Seiten links, auf linken Seiten rechts
- `o` – außen: auf rechten Seiten rechts, auf linken Seiten links

v3.00 Voreingestellt ist rechts neben dem Inhalt der Umgebung. Diese Voreinstellung kann jedoch mit der Option `captions` (siehe Seite 120) und deren Werte `innerbeside`, `leftbeside`, `outerbeside` und `rightbeside` verändert werden. Bei Verwendung der Anordnung außen oder innen werden unter Umständen zwei LaTeX-Läufe benötigt, um die korrekte Anordnung zu erreichen.

Normalerweise nehmen der Inhalt der Umgebung und der *Titel* die gesamte verfügbare Breite ein. Es besteht jedoch die Möglichkeit, mit dem optionalen Parameter *Breite* eine andere Breite anzugeben. Diese kann auch größer als die Breite des Textkörpers sein.

Bei Angabe einer *Breite* wird die genutzte Breite normalerweise bezüglich der Breite des Textkörpers zentriert. Mit dem optionalen Argument *Offset* kann stattdessen eine Verschiebung relativ zum linken Rand angegeben werden. Ein positiver Wert entspricht einer Verschiebung nach rechts, ein negativer Wert einer Verschiebung nach links. Mit einem *Offset* von 0 pt erfolgt die Ausgabe linksbündig.

Wird hinter den optionalen Parameter *Offset* noch ein Stern gesetzt, so stellt der *Offset* im doppelseitigen Druck auf linken Seiten eine Verschiebung relativ zum rechten Rand dar. Ein positiver Wert entspricht dann einer Verschiebung nach außen, während ein negativer Wert für eine Verschiebung nach innen steht. Ein *Offset* von 0 pt wäre dann also bündig zum inneren Rand. Diese Variante benötigt unter Umständen zwei LaTeX-Durchläufe, um die korrekte Verschiebung zu erreichen.

Vertikal erfolgt die Ausrichtung in der Voreinstellung unten. Das bedeutet, dass die untere Grundlinie des Inhalts des Gleitobjekts und die untere Grundlinie von *Titel* auf einer Höhe liegen. Diese Einstellung kann mit der Option `captions` (siehe Seite 120) und deren Werte `topbeside`, `centeredbeside` und `bottombeside` verändert werden. Bei der Einstellung `topbeside` werden die oberen Grundlinien von Gleitobjektinhalt und *Titel* auf einer Höhe ausgerichtet, während bei `centeredbeside` eine Zentrierung stattfindet. In diesem Zusammenhang sei erwähnt, dass Abbildungen normalerweise die Grundlinie unten haben. Dies kann beispielsweise mit der Anweisung `\raisebox` verändert werden.

v3.00

Beispiel: Ein Beispiel für die Verwendung der `captionbeside`-Umgebung ist in Abbildung 3.4 zu finden. Gesetzt wurde diese Abbildung mit:

```
\begin{figure}
  \begin{captionbeside}%
    [Beispiel: Bildbeschreibung daneben, unten]%
    {Eine Bildbeschreibung weder über noch unter
      der Abbildung, sondern unten daneben}%
    [i][\linewidth][%
      \dimexpr\marginparwidth+\marginparsep\relax]*
    \fbox{%
      \parbox[b][5\baselineskip][c]{.25\textwidth}
      {%
        \hspace*{\fill}\KOMAScript
        \hspace*{\fill}\par
      }%
    }
  \end{captionbeside}
  \label{fig:maincls.captionbeside}
\end{figure}
```

Die Gesamtbreite ist also die aktuell verfügbare Breite `\linewidth`. Diese wird jedoch mit `2em` nach außen verschoben. Der Titel oder die Beschreibung steht innen neben der Abbildung. Damit erscheint die Abbildung selbst um 2 em in den Rand

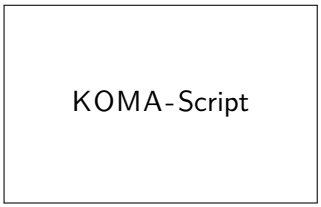


Abbildung 3.4.: Eine Bildbeschreibung weder über noch unter der Abbildung, sondern unten daneben

gerückt.

Mit `\dimexpr` wird hier eine ε-TEX-Anweisung verwendet, da KOMA-Script dessen Verwendung voraussetzt und bei jeder halbwegs aktuellen LATEX-Distribution ohnehin ε-TEX verwendet wird.

Die Zentrierung der Bildbeschreibung mit

```
\KOMAoption{captions}{centeredbeside}
```

zeigt Abbildung 3.5. Auch der typografisch weniger bewanderte Anfänger wird sofort erkennen, dass von dieser Ausrichtung normalerweise abzusehen ist.

Demgegenüber kann die Ausrichtung oben, die bei Abbildung 3.6 zu sehen ist, durchaus verwendet werden. Zur Verdeutlichung, wie man `\raisebox` zur Verschiebung der Grundlinie nutzen kann, sei hier ein komplettes Beispiel angegeben. Eine solche Verschiebung kann man nicht nur bei einer Ersatzgrafik wie angegeben, sondern auch beispielsweise auf `\includegraphics` (siehe [Car99d]) anwenden.

```
\documentclass[captions=topbeside]{scrbook}
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\chapter{Ein Beispiel}
\begin{figure}
  \begin{captionbeside}%
    [Beispiel: Bildbeschreibung daneben, oben]%
    {Eine Bildbeschreibung weder über noch unter
      der Abbildung, sondern oben daneben}%
    [i][\linewidth][%
      \dimexpr\marginparwidth+\marginparsep\relax
    ]*
```

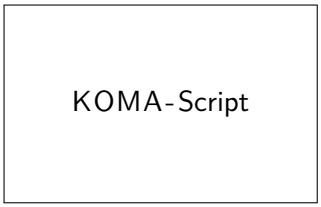
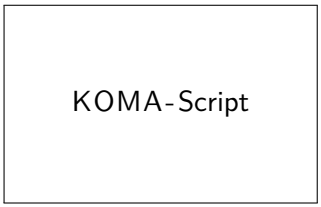


Abbildung 3.5.: Eine Bildbeschreibung weder über noch unter der Abbildung, sondern mittig daneben

Abbildung 3.6.: Eine Bildbeschreibung weder über noch unter der Abbildung, sondern oben daneben




```
    \raisebox{%
      \dimexpr\baselineskip-\totalheight\relax
    }{%
      \includegraphics{beispielbild}%
    }%
  \end{captionbeside}
  \label{fig:maincls.captionbesidetop}
\end{figure}
\end{document}
```

```
\begin{captionofbeside}{Objekttyp}[Verzeichnistitel]{Titel}%
                        [Anordnung][Breite][Offset]
  ...
\end{captionofbeside}
\begin{captionofbeside}{Objekttyp}[Verzeichnistitel]{Titel}%
                        [Anordnung][Breite][Offset]*
  ...
\end{captionofbeside}
```

v3.10 Wie zu `\caption` mit `\captionof` eine Variante existiert, bei der der *Objekttyp* nicht durch die Verwendung innerhalb einer Gleitumgebung dieses Typs bestimmt wird, so gibt es passend zur Umgebung `captionbeside` mit `captionofbeside` auch eine entsprechende Umgebung. Im Unterschied zu `captionbeside` ist auch hier, dass der *Objekttyp* als zusätzliches, erstes Argument anzugeben ist.

`komaabove`
`komabelow`

float Bei Verwendung des float-Pakets wird das Aussehen der damit definierten Gleitumgebungen allein vom *float*-Stil bestimmt. Dies schließt auch die Frage ein, ob mit Überschriften oder Unterschriften gearbeitet wird. Im float-Paket gibt es keinen vordefinierten Stil, der im Aussehen dem von KOMA-Script entspricht und dieselben Einstellmöglichkeiten (siehe unten) bietet. KOMA-Script definiert deshalb zusätzlich die beiden Stile `komaabove` und `komabelow`. Diese können bei Verwendung des float-Pakets wie die dort definierten Stile `plain`, `boxed` oder `ruled` aktiviert werden. Siehe dazu [Lin01]. Beim Stil `komaabove` werden `\caption`, `\captionabove` und `\captionbelow` als Überschrift, beim Stil `komabelow` als Unterschrift gesetzt.

`\captionformat`

Bei KOMA-Script gibt es verschiedene Eingriffsmöglichkeiten, um die Formatierung der Beschreibung zu ändern. Die Änderung der Schriftart wurde bereits erläutert. Das oder die Trennzeichen zwischen dem Label und dem eigentlichen Beschreibungstext sind im Makro `\captionformat` abgelegt. Abweichend von allen anderen `\...format`-Anweisungen ist hier also nicht der Zähler, sondern nur die auf den Zähler folgenden Angaben enthalten. Die Originaldefinition lautet:

```
\newcommand*{\captionformat}{:\ }
```

Auch diese kann mit `\renewcommand` geändert werden.

Beispiel: Aus mir unerfindlichen Gründen wollen Sie als Trennzeichen keinen Doppelpunkt gefolgt von einem Leerzeichen, sondern einen Gedankenstrich einschließlich der notwendigen Leerzeichen. Daher definieren Sie:

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

Diese Definition sollten Sie beispielsweise in die Präambel Ihres Dokuments stellen.

`\figureformat`
`\tableformat`

Es wurde schon darauf hingewiesen, dass `\captionformat` keine Formatierung für das Label selbst enthält. Dieses sollte nun keineswegs über Umdefinierung der Anweisungen für die Zählerausgabe, `\thefigure` oder `\thetable`, verändert werden. Eine solche Umdefinierung hätte nämlich auch Auswirkungen auf die Ausgabe von `\ref` oder der Verzeichnisse. Stattdessen bietet KOMA-Script auch hier zwei `\...format`-Anweisungen. Diese sind wie folgt vordefiniert:

```
\newcommand*{\figureformat}{\figurename~\thefigure\autodot}
\newcommand*{\tableformat}{\tablename~\thetable\autodot}
```

Sie können ebenfalls mit `\renewcommand` eigenen Anforderungen angepasst werden.

Beispiel: Hin und wieder wird gewünscht, dass die Beschreibungstexte ganz ohne Label und natürlich auch ohne Trennzeichen ausgegeben werden. Bei KOMA-Script genügen folgende Definitionen, um dies zu erreichen:

```
\renewcommand*{\figureformat}{}
\renewcommand*{\tableformat}{}
\renewcommand*{\captionformat}{}
```

Dabei ist jedoch zu beachten, dass die Nummerierung damit zwar nicht ausgegeben, aber dennoch fortgezählt wird. Dies ist insbesondere dann von Bedeutung, wenn die Umdefinierungen nur auf einzelne `figure`- oder `table`-Umgebungen angewendet werden.

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

Wie bereits erwähnt wurde, werden in den Standardklassen die Beschreibungen nicht hängend gesetzt. Das heißt: In mehrzeiligen Beschreibungen beginnt die zweite Zeile direkt unter dem Labeltext. Es gibt bei den Standardklassen auch keinen Mechanismus, dies direkt zu beeinflussen. Bei KOMA-Script werden hingegen alle Zeilen ab der zweiten so weit eingerückt, dass diese nicht mehr unter dem Label, »Abbildung . . . :« oder »Tabelle . . . :«, sondern unter dem eigentlichen Text der ersten Zeile beginnen.

Dieses Verhalten, das der Verwendung von `\setcaphanging` entspricht, kann bei KOMA-Script jederzeit durch Verwendung der Anweisung `\setcapindent` oder `\setcapindent*` geändert werden. Dabei gibt der Parameter *Einzug* an, wie weit ab der zweiten Zeile eingerückt werden soll. Soll nach dem Label und vor dem Beschreibungstext noch ein Zeilenumbruch erfolgen, so definieren Sie die Einrücktiefe *XEinzug* der Beschreibung stattdessen mit der Sternvariante der Anweisung: `\setcapindent*`. Mit einem negativen *Einzug* erreicht man hingegen, dass vor der Beschreibung ebenfalls ein Umbruch erfolgt und nur die erste Zeile der Beschreibung, nicht jedoch die folgenden, um den Betrag von *Einzug* eingerückt werden.

Ob einzeilige Beschreibungen wie mehrzeilige Beschreibungen gesetzt werden oder eine Sonderbehandlung erfahren, wird über die Option `captions` gewählt. Siehe hierzu die Erklärung zu den Werten `oneline` und `nooneline` dieser Option auf Seite 121.

Beispiel: Die Abbildungen 3.7 bis 3.10 zeigen die Auswirkungen unterschiedlicher Einstellungen. Dabei wird deutlich, dass bei geringer Spaltenbreite der komplett hängende Einzug unvorteilhaft ist. Der Quelltext der zweiten Abbildung sei hier mit abgewandelter Unterschrift beispielhaft wiedergegeben:

```
\begin{figure}
  \setcapindent{1em}
  \fbox{\parbox{.95\linewidth}{%
      \centering\KOMAScript}}
  \caption{Beispiel mit teilweise hängendem Einzug
    ab der zweiten Zeile}
\end{figure}
```

Wie zu sehen ist, kann die Formatierung also auch lokal innerhalb der `figure`-Umgebung geändert werden. Die Änderung gilt dann nur für die eine Abbildung. Nachfolgende Abbildungen werden wieder mit den Grundeinstellungen oder den globalen Einstellungen, die Sie beispielsweise in der Dokumentpräambel vorgenommen haben, gesetzt. Das gilt für Tabellen natürlich genauso.

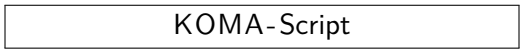


Abbildung 3.7.: Mit der Standardeinstellung, also wie bei Verwendung von `\setcaphanging`

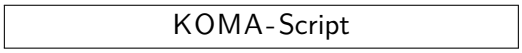


Abbildung 3.8.: Mit teilweise hängendem Einzug ab der zweiten Zeile durch Verwendung von `\setcapindent{1em}`

Abbildung 3.9.:
Mit hängendem Einzug ab der zweiten Zeile und Umbruch vor der Beschreibung durch Verwendung von `\setcapindent*{1em}`

Abbildung 3.10.:
Mit Einzug lediglich in der zweiten Zeile und einem Umbruch vor der Beschreibung durch Verwendung von `\setcapindent{-1em}`

`\setcapwidth[`*`Ausrichtung`*`]{`*`Breite`*`}`
`\setcapmargin[`*`Rand links`*`]{`*`Rand`*`}`
`\setcapmargin*[`*`Rand innen`*`]{`*`Rand`*`}`

v2.8q

Mit Hilfe dieser drei Befehle kann die Breite und Anordnung der Beschreibung beeinflusst werden. Normalerweise steht die gesamte Text- oder Spaltenbreite für den Text der Beschreibung zur Verfügung.

Mit der Anweisung `\setcapwidth` kann diese *`Breite`* reduziert werden. Dabei gibt das obligatorische Argument die maximale für die Beschreibung verwendete *`Breite`* an. Als optionales Argument kann genau ein Buchstabe übergeben werden, der die horizontale Ausrichtung der Beschreibung angibt. Die möglichen Ausrichtungen finden Sie in der folgenden Liste.

`l` – linksbündig

`c` – zentriert

`r` – rechtsbündig

`i` – innen: auf rechten Seiten linksbündig, auf linken Seiten rechtsbündig

`o` – außen: auf rechten Seiten rechtsbündig, auf linken Seiten linksbündig

Die Ausrichtung innen und außen entspricht im einseitigen Satz linksbündig und rechtsbündig. Innerhalb von longtable-Tabellen funktioniert die Ausrichtung innen und außen nicht korrekt. Insbesondere werden Beschreibungen von Folgeseiten bei diesen Tabellen immer nach den Beschreibungen der ersten Teiltabelle ausgerichtet. Dies ist ein konzeptionelles Problem des Paketes longtable.

Mit der Anweisung `\setcapmargin` kann statt der Breite der Beschreibung ein *`Rand`* angegeben werden, der neben der Beschreibung zusätzlich zum normalen Textrand eingehalten werden soll. Sollen der Rand rechts und links nicht identisch gewählt werden, kann mit dem

optionalen Argument ein von *Rand* abweichender *Rand links* von der Beschreibung eingestellt werden. Bei der Sternvariante `\setcapmargin*` wird statt *Rand links* im doppelseitigen Satz *Rand innen* abweichend definiert. Hier ergibt sich bei longtable-Tabellen das gleiche Problem wie bei der Ausrichtung außen oder innen bei der Anweisung `\setcapwidth`. Die Verwendung von `\setcapmargin` oder `\setcapmargin*` aktiviert außerdem die Einstellung `captions=nooneline` (siehe Seite 121) für die Beschreibungen, die mit dieser Randeinstellung gesetzt werden.

Man kann übrigens auch negative Werte für *Rand* und *Rand rechts* oder *Rand außen* angeben. Dadurch erreicht man, dass die Beschreibung in den entsprechenden Rand hineinragt.

Für Experten und versierte Anwender ist eine etwas trickreiche Anwendung für `\setcapwidth` in [KM08] zu finden.

`origlongtable`

Falls die Tabellenüberschriften des longtable-Pakets (siehe [Car04]) von den KOMA-Script-Klassen nicht umdefiniert werden sollen, kann die Option `origlongtable` gesetzt werden. Diese Option ist als optionales Argument von `\documentclass` zu verwenden. Eine Einstellung per `\KOMAoptions` oder `\KOMAoption` wird nicht unterstützt.

`listof=`*Einstellung*

v3.00

Normalerweise werden die Verzeichnisse der Gleitumgebungen – wie das Tabellen- und das Abbildungsverzeichnis – nicht nummeriert oder in das Inhaltsverzeichnis aufgenommen. In Abschnitt 3.9 wurde dies bereits näher ausgeführt. Alternativ zu den dort erwähnten Einstellungen `toc=nolistof`, `toc=listof` und `toc=listofnumbered`, kann dieses Verhalten auch aus Sicht der Verzeichnisse selbst gesehen werden. Daher kann man die gleichen Ergebnisse auch mit den Einstellungen `listof=notoc`, `listof=totoc` und `listof=numbered` erreichen.

Dabei werden in der Voreinstellung für die Überschriften der Verzeichnisse die oberste verfügbare Gliederungsebene unterhalb von `\part` verwendet. Bei scrbook und scrreprt ist das die Kapitelebene, bei scrartcl die Abschnittsebene. Mit Hilfe der Einstellung `listof=leveldown` kann hingegen die nächst tiefere Gliederungsebene verwendet werden.

Beispiel: Sie wollen in einem Buch das Abbildungs- und das Tabellenverzeichnis als Unterverzeichnisse eines gemeinsamen Verzeichnisses »Abbildungen und Tabellen« setzen. Dazu verwenden Sie einfach:

```
\KOMAoption{listof}{leveldown}
```

und dann an entsprechender Stelle Ihres Dokuments:

```
\addchap*{Abbildungs- und Tabellenverzeichnis}
\listoffigures
\listoftables
```

Näheres zur Anweisung `\addchap*` ist Abschnitt 3.16, Seite 99 zu entnehmen.

v2.8q

Normalerweise werden die Verzeichnisse der Gleitumgebungen so formatiert, dass für die Nummer ein Raum fester Breite verwendet wird. Gleichzeitig werden alle Einträge leicht eingezogen. Dies entspricht der Verwendung der Einstellung `listof=graduated`.

Werden die Nummern sehr breit, weil beispielsweise sehr viele Tabellen verwendet werden, so reicht der vorgesehene Platz irgendwann nicht mehr aus. Vergleichbar zur Einstellung `toc=flat` für das Inhaltsverzeichnis bietet KOMA-Script daher die Einstellung `listof=flat` für die Verzeichnisse der Gleitumgebungen. Dabei wird die Breite der Nummern automatisch ermittelt und der Platz entsprechend angepasst. Bezüglich der Nebenwirkungen und Funktionsweise gilt, was in Abschnitt 3.9, Seite 68 für die Einstellung `toc=flat` erklärt wurde. Es sei an dieser Stelle jedoch nochmals darauf hingewiesen, dass mit der Einstellung `listof=flat` mehrere LaTeX-Durchläufe benötigt werden, bis die Verzeichnisse ihre endgültige Form erhalten haben.

v3.06

Die Einstellung `listof=flat` wird automatisch aktiviert, falls die Einstellung `listof=entryprefix` verwendet wird. Normalerweise ist es nicht sinnvoll jeden Eintrag in eines der Verzeichnisse der Gleitumgebungen mit einem Präfix wie »Abbildung« oder »Tabelle« zu versenden, das natürlich im Abbildungsverzeichnis nur Abbildungen und im Tabellenverzeichnis nur Tabellen zu finden sind. Damit hat ein solcher Präfix keinen zusätzlichen Informationswert und wird in der Voreinstellung auch weggelassen. Mit der Einstellung `listof=entryprefix` wird ein solcher Präfix jedoch gesetzt. Dabei erhalten alle Einträge eines Verzeichnisses denselben Präfix. Dieser richtet sich nach dem Dateianhang der Hilfsdatei, die für das Verzeichnis verwendet wird. Für das Abbildungsverzeichnis, das den Dateianhang »`lof`« besitzt, wird beispielsweise `\listoflofentryname` verwendet, während für das Tabellenverzeichnis, das den Dateianhang »`lot`« besitzt, `\listoflotentryname` verwendet wird.

scrbook, scrreprt

v3.00

Bei den Klassen scrbook und scrreprt fügt KOMA-Script in der Voreinstellung bei jedem Kapitelanfang einen vertikalen Abstand in die Verzeichnisse der Gleitumgebungen ein. Dieses Verhalten, das es auch bei den Standardklassen gibt, dient dazu, diese Verzeichnisse nach Kapiteln zu gruppieren. Es entspricht bei KOMA-Script der Einstellung `listof=chaptergapsmall`. Dabei wird ein fester vertikaler Abstand von 10 pt verwendet. Mit der Einstellung `listof=chaptergapline` kann man stattdessen einen vertikalen Abstand von einer Zeile erreichen. Mit `listof=nochaptergap` kann man den vertikalen Abstand komplett abschalten. Eine Besonderheit stellt die Einstellung `listof=chapterentry` dar. Dabei wird statt des Abstandes der Inhaltsverzeichniseintrag für das Kapitel in das Verzeichnis der Gleitumgebungen eingefügt. Es wird darauf hingewiesen, dass ein solcher Eintrag auch dann erfolgt, wenn das Kapitel keine Gleitumgebung enthält. Eine noch direktere Beeinflussung, was in den Verzeichnissen der Gleitumgebungen bei neuen Kapiteln geschehen soll, ist mit der Option `chapteratlists` zu erreichen, die in Abschnitt 3.16 auf Seite 92 erläutert wird.

Einen Überblick über alle möglichen Werte für die *`Einstellung`* von `listof` ist in Tabelle 3.20 zu finden.

Tabelle 3.20.: Mögliche Werte für Option `listof` zur Einstellung von Form und Inhalt der Verzeichnisse der Gleitumgebungen

`chapterentry`, `withchapterentry`
: Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen durch einen Inhaltsverzeichniseintrag des Kapitels markiert.

`chaptergapline`, `onelinechaptergap`
: Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen durch einen Abstand von einer Zeile markiert.

`chaptergapsmall`, `smallchaptergap`
: Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen durch einen kleinen Abstand markiert.

`entryprefix`
v3.06
: Jeder Verzeichniseintrag wird mit einem vom Verzeichnis abhängenden Präfix vor der Nummer versehen. Der Präfix ist normalerweise sprachabhängig, beispielsweise bei deutschen Spracheinstellungen »Abbildung« für das Abbildungsverzeichnis und »Tabelle« für das Tabellenverzeichnis jeweils gefolgt von einem Leerzeichen.

`flat`, `left`
: Die Verzeichnisse der Gleitumgebungen erhalten eine tabellarische Form. Die Gleitumgebungsnummern sind dabei die erste Spalte, der Titel die zweite Spalte, die Seitenzahlen die dritte Spalte. Der Platz, der für die Gleitumgebungsnummern reserviert wird, richtet sich nach dem benötigten Platz des vorherigen LaTeX-Laufs.

`graduated`, `indent`, `indented`
: Die Verzeichnisse der Gleitumgebungen erhalten eine hierarchische Form. Es steht nur ein begrenzter Platz für die Gleitumgebungsnummern zur Verfügung.

`leveldown`
: Die Verzeichnisse werden um eine Gliederungsebene nach unten verschoben.

`nochaptergap`, `ignorechapter`
: Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen nicht markiert.

`notoc`, `plainheading`
: Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten keinen Eintrag im Inhaltsverzeichnis.

...

Tabelle 3.20.: Mögliche Werte für Option `listof` (*Fortsetzung*)

`numbered`, `totocnumbered`, `tocnumbered`, `numberedtotoc`	Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten einen Eintrag im Inhaltsverzeichnis und werden nummeriert.
`totoc`, `toc`, `notnumbered`	Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten einen Eintrag im Inhaltsverzeichnis, ohne dass sie nummeriert werden.

```
\listoftables
\listoffigures
```

Mit diesen Anweisungen kann ein Verzeichnis der Tabellen beziehungsweise der Abbildungen ausgegeben werden. Änderungen, die Auswirkungen auf diese Verzeichnisse haben, werden erst nach zwei LaTeX-Läufen sichtbar. Die Form der Verzeichnisse kann durch die Option `listof` mit den Werten `graduated` und `flat` beeinflusst werden (siehe Seite 134). Darüber hinaus wirken sich indirekt die Werte `listof` und `listofnumbered` für die Option `toc` (siehe Abschnitt 3.9, Seite 67), sowie die Werte `totoc` und `totocnumbered` der oben erläuterten Option `listof` auf die Verzeichnisse aus.

In der Regel findet man die Verzeichnisse der Gleitumgebungen, also das Tabellen- und das Abbildungsverzeichnis, unmittelbar nach dem Inhaltsverzeichnis. In einigen Dokumenten wandern diese auch in den Anhang. Der Autor bevorzugt jedoch die Platzierung unmittelbar nach dem Inhaltsverzeichnis.

3.21. Randnotizen

Außer dem eigentlichen Textbereich, der normalerweise den Satzspiegel ausfüllt, existiert in Dokumenten noch die so genannte Marginalienspalte. In dieser können Randnotizen gesetzt werden. In diesem Dokument wird davon ebenfalls Gebrauch gemacht.

```
\marginpar[Randnotiz links]{Randnotiz}
\marginline{Randnotiz}
```

Für Randnotizen ist bei LaTeX normalerweise Anweisung `\marginpar` vorgesehen. Die *`Randnotiz`* wird dabei im äußeren Rand gesetzt. Bei einseitigen Dokumenten wird der rechte Rand verwendet. Zwar kann bei `\marginpar` optional eine abweichende Randnotiz angegeben werden, falls die Randnotiz im linken Rand landet, jedoch werden Randnotizen immer im Blocksatz ausgegeben. Die Erfahrung zeigt, dass bei Randnotizen statt des Blocksatzes oft

je nach Rand linksbündiger oder rechtsbündiger Flattersatz zu bevorzugen ist. KOMA-Script bietet hierfür die Anweisung `\marginline`.

Beispiel: In diesem Kapitel ist an einigen Stellen die Klassenangabe scrartcl im Rand zu finden. Diese kann mit:

```
\marginline{\texttt{scrartcl}}
```

erreicht werden.

Statt der Anweisung `\marginline` wäre auch die Verwendung von `\marginpar` möglich gewesen. Tatsächlich wird bei obiger Verwendung von `\marginline` intern nichts anders gemacht als:

```
\marginpar[\raggedleft\texttt{scrartcl}]
  {\raggedright\texttt{scrartcl}}
```

Letztlich ist `\marginline` also nur eine abkürzende Schreibweise.

Für Experten sind in Abschnitt 16.1, Seite 307 Probleme bei der Verwendung von `\marginpar` dokumentiert. Diese gelten ebenso für `\marginline`.

3.22. Anhang

Der Anhang eines Dokuments besteht im Wesentlichen aus den Anlagen zu einem Dokument. Typische Teile eines Anhangs sind Literaturverzeichnis, Stichwortverzeichnis und Begriffsverzeichnis. Alleine für diese Teile würde man jedoch keinen Anhang beginnen, da diese Teile normalerweise schon von sich aus eine Auszeichnung besitzen, die sie als Anhang erkennbar macht. Enthält der Anhang aber weitere Teile wie beispielsweise zitierte Fremddokumente, Endnoten oder Tafeln, so werden die zuvor genannten Teile ebenfalls im Anhang gesetzt.

`\appendix`

Der Anhang wird in den Standardklassen und den KOMA-Script-Klassen mit der Anweisung `\appendix` eingeleitet. Diese Anweisung schaltet unter anderem die Kapitelnummerierung auf Großbuchstaben um und sorgt gleichzeitig dafür, dass die Regeln für die Nummerierung der Gliederungsebenen nach [DUD96] eingehalten werden. Diese Regeln sind in der Beschreibung der Option `numbers` in Abschnitt 3.16, Seite 92 näher erläutert.

scrbook, scrreprt

Die Form der Kapitelüberschriften im Anhang wird durch die Optionen `chapterprefix` und `appendixprefix` bestimmt. Näheres dazu ist Abschnitt 3.16, Seite 89 zu entnehmen.

Bitte beachten Sie, dass es sich bei `\appendix` um eine Anweisung und *nicht* um eine Umgebung handelt! Die Anweisung erwartet auch nicht etwa ein Argument. Die Kapitel beziehungsweise Abschnitte des Anhangs werden ganz normal mit `\chapter` und `\section` gesetzt.

3.23. Literaturverzeichnis

Das Literaturverzeichnis erschließt externe Quellen. In der Regel wird das Literaturverzeichnis mit Hilfe des Programms BibTeX aus einer Datei mit datenbankähnlicher Struktur erzeugt. Dabei kann über den BibTeX-Stil sowohl die Form der Einträge als auch deren Sortierung verändert werden. Wird zusätzlich ein Literaturpaket, beispielsweise natbib, babelbib oder biblatex verwendet, so schwindet der Einfluss von KOMA-Script auf das Literaturverzeichnis. In diesen Fällen ist unbedingt die Anleitung des verwendeten Pakets zu beachten! Zur generellen Verwendung eines Literaturverzeichnisses sei auf [SKPH99] verwiesen.

`bibliography=`*`Einstellung`*

v3.00

Als *Einstellung* kann zunächst einmal jeder definierte Formatierungsstil gewählt werden. Vordefiniert sind bei KOMA-Script zwei solche Formatierungsstile für das Literaturverzeichnis. Diese sind jedoch nicht zu verwechseln mit den unterschiedlichen Stilen für BibTeX, die man mit `\bibstyle` auswählt. Während BibTeX sowohl die Art der Sortierung als auch den Inhalt des Literaturverzeichnisses bestimmt, können über die Einstellungen von KOMA-Script nur grundlegende Eigenschaften des Literaturverzeichnisses oder einige wenige Eigenschaften der Formatierung der Einträge beeinflusst werden.

Mit `bibliography=oldstyle` wird die kompakte Formatierung gewählt. Dabei führt die Anweisung `\newblock` in den einzelnen Einträgen lediglich zu einem dehnbaren horizontalen Abstand. Der Name kommt daher, dass dies die häufigste klassische Form eines Literaturverzeichnisses ist. Demgegenüber erreicht man die etwas modernere, offene Form mit der Einstellung `bibliography=openstyle`. Der Name kommt daher, dass hier die Anweisung `\newblock` einen Absatz einfügt. Die Einträge im Literaturverzeichnis werden so stärker gegliedert. Sie sind weniger kompakt und deutlich aufgelockerter oder geöffnet. Bezüglich der Möglichkeit, neue Formatierungsstile zu definieren, sei auf `\newbibstyle`, Abschnitt 16.3, Seite 313 verwiesen.

Neben dem Formatierungsstil gibt es eine weitere Eigenschaft, die über *Einstellung* verändert werden kann. Das Literaturverzeichnis stellt eine Art von Verzeichnis dar, bei der nicht der Inhalt des vorliegenden Werks aufgelistet, sondern auf externe Inhalte verwiesen wird. Mit dieser Begründung könnte man argumentieren, dass das Literaturverzeichnis ein eigenes Kapitel bzw. einen eigenen Abschnitt darstellt und somit eine Nummer verdiene. Die Einstellung `bibliography=totocnumbered` führt genau dazu, einschließlich des dann fälligen Eintrags im Inhaltsverzeichnis. Ich selbst bin der Meinung, dass bei dieser Argumentation auch ein klassisches, kommentiertes Quellenverzeichnis ein eigenes Kapitel wäre. Außerdem ist das Literaturverzeichnis letztlich nichts, was man selbst geschrieben hat. Deshalb verdient es allenfalls einen nicht nummerierten Eintrag im Inhaltsverzeichnis, was mit der Einstellung `bibliography=totoc` erreicht wird. Die Voreinstellung, bei der das Literaturverzeichnis als nicht nummeriertes Kapitel ohne eigenen Inhaltsverzeichniseintrag gesetzt wird, entspricht

Tabelle 3.21.: Vordefinierte Werte für Option `bibliography` zur Einstellung der Form des Literaturverzeichnisses

`nottotoc`	Das Literaturverzeichnis erhält keinen Eintrag im Inhaltsverzeichnis und wird auch nicht nummeriert.
`oldstyle`	Es wird die klassische, kompakte Formatierung gewählt, bei der `\newblock` nur einen dehnbaren horizontalen Abstand darstellt.
`openstyle`	Es wird eine untergliederte, offene Formatierung gewählt, bei der `\newblock` einen Absatz darstellt.
`totoc`	Das Literaturverzeichnis erhält einen Eintrag im Inhaltsverzeichnis, ohne dass es nummeriert wird.
`totocnumbered`	Das Literaturverzeichnis erhält einen Eintrag im Inhaltsverzeichnis und wird nummeriert.

`bibliography=nottotoc`. Siehe hierzu auch Option `toc` in Abschnitt 3.9, insbesondere die Werte `bibliographynumbered`, `bibliography` und `nobibliography` ab Seite 68.

Eine Zusammenfassung der möglichen Werte für die *Einstellung* von `bibliography` ist in Tabelle 3.21 zu finden. Es ist jedoch zu beachten, dass mit `\newbibstyle` weitere Werte definiert werden können.

`\setbibpreamble{`*Präambel*`}`

Mit der Anweisung `\setbibpreamble` kann eine Präambel für das Literaturverzeichnis gesetzt werden. Bedingung dafür ist, dass die Präambel vor der Anweisung zum Setzen des Literaturverzeichnisses gesetzt wird. Dies muss nicht unmittelbar davor sein. Es kann also beispielsweise am Anfang des Dokuments erfolgen. Ebenso wie die Klassenoptionen `bibtotoc` und `bibtotocnumbered` kann die Anweisung aber nur erfolgreich sein, wenn nicht ein Paket geladen wird, das dies durch Umdefinierung der `thebibliography`-Umgebung verhindert. Obwohl das natbib-Paket nicht freigegebene interne Makros von KOMA-Script verwendet, konnte erreicht werden, dass `\setbibpreamble` auch mit der aktuellen Version von natbib funktioniert (siehe [Dal99]).

Beispiel: Sie wollen darauf hinweisen, dass das Literaturverzeichnis nicht in der Reihenfolge der Zitierung im Dokument, sondern alphabetisch sortiert ist. Daher setzen Sie

folgende Anweisung:

```
\setbibpreamble{Die Literaturangaben sind
  alphabetisch nach den Namen der Autoren
  sortiert. Bei mehreren Autoren wird nach dem
  ersten Autor sortiert.\par\bigskip}
```

Die Anweisung `\bigskip` sorgt dafür, dass zwischen der Präambel und der ersten Literaturangabe ein großer Zwischenraum gesetzt wird.

`\BreakBibliography{`*`Unterbrechung`*`}`

v3.00 Diese Anweisung existiert nur, wenn Umgebung `thebibliography` nicht durch ein Paket neu definiert wurde. In diesem Fall ist es möglich, mit dieser Anweisung das Literaturverzeichnis zu unterbrechen. Die *`Unterbrechung`* wird dann innerhalb einer Gruppe ausgegeben. Eine solche *`Unterbrechung`* könnte beispielsweise eine Überschrift mit Hilfe von `\minisec` sein. Leider gibt es bisher keine Möglichkeit, diese Anweisung beispielsweise mit Hilfe eines speziellen Eintrags in der Literaturdatenbank von BibTeX erzeugen zu lassen. Daher kann sie derzeit nur von Anwendern verwendet werden, die das Literaturverzeichnis selbst editieren. Ihr Nutzen ist damit sehr beschränkt.

`\AfterBibliographyPreamble{`*`Anweisungen`*`}`
`\AtEndBibliography{`*`Anweisungen`*`}`

v3.00 In einigen Fällen ist es nützlich, wenn man nach der Präambel des Literaturverzeichnisses oder unmittelbar vor dem Ende des Literaturverzeichnisses noch *`Anweisungen`* ausführen kann. Dies ist mit Hilfe dieser beiden Anweisungen möglich.

Beispiel: Sie wollen, dass das Literaturverzeichnis nicht im Blocksatz, sondern im linksbündigen Flattersatz ausgegeben wird. Dies ist einfach mit:

```
\AfterBibliographyPreamble{\raggedright}
```

zu erreichen. Sie können diese Anweisung an beliebiger Stelle vor dem Literaturverzeichnis verwenden. Es wird jedoch empfohlen, sie in die Präambel des Dokuments oder ein eigenes Paket zu schreiben.

Die Realisierung dieser Anweisung bedarf bei Verwendung eines Pakets, das die Umgebung für Literaturverzeichnisse umdefiniert, der Zusammenarbeit mit dem entsprechenden Paket (siehe Abschnitt 16.2, Seite 307).

3.24. Stichwortverzeichnis

Das Stichwortverzeichnis ist auch unter den Bezeichnungen Index oder Register bekannt. Zur generellen Verwendung eines Stichwortverzeichnisses sei auf [SKPH99] sowie auf [Lam87] und [Keh97] verwiesen. Wird ein Paket verwendet, das selbst Anweisungen und Umgebungen für

Tabelle 3.22.: Mögliche Werte für Option `index` zur Einstellung des Stichwortverzeichnisses

`totoc`, `toc`, `notnumbered`	Das Stichwortverzeichnis erhält einen Eintrag im Inhaltsverzeichnis, ohne dass er nummeriert wird.
`default`, `nottotoc`, `plainheading`	Das Stichwortverzeichnis erhält keinen Eintrag im Inhaltsverzeichnis.

das Stichwortverzeichnis zur Verfügung stellt, so schwindet eventuell der Einfluss, den KOMA-Script auf dieses Verzeichnis hat. Dies gilt beispielsweise bei Verwendung von index, nicht jedoch bei Verwendung von splitidx (siehe [Koh06]).

`index=`*`Einstellung`*

v3.00

In der Voreinstellung `index=default` ist das Stichwortverzeichnis ein nicht nummeriertes Kapitel ohne Eintrag im Inhaltsverzeichnis. Da das Stichwortverzeichnis normalerweise in einem Buch oder ähnlichen Dokument zuletzt steht, benötigt es eigentlich auch keinen Inhaltsverzeichniseintrag. Wird dieser dennoch gewünscht, beispielsweise weil wie in dieser Anleitung mit einem mehrgliedrigen Stichwortverzeichnis gearbeitet wird, so kann dies mit der Einstellung `index=totoc` erreicht werden. Siehe hierzu auch Option `toc` mit dem Wert `index` in Abschnitt 3.9 ab Seite 68.

Eine Zusammenfassung der möglichen Werte für die *`Einstellung`* von `index` ist in Tabelle 3.22 zu finden.

`\setindexpreamble{`*`Präambel`*`}`

Analog zur Präambel des Literaturverzeichnisses können Sie auch das Stichwortverzeichnis mit einer Präambel versehen. Dies findet häufig dann Anwendung, wenn es mehr als einen Index gibt oder im Index unterschiedliche Arten der Referenzierung durch unterschiedliche Hervorhebung der Seitenzahlen markiert werden.

Beispiel: Sie haben ein Dokument, in dem Begriffe sowohl definiert als auch verwendet werden. Die Seitenzahlen der Begriffsdefinitionen sind fett dargestellt. Natürlich möchten Sie gerne auf diesen Umstand hinweisen. Also setzen Sie eine entsprechende Präambel für den Index:

```
\setindexpreamble{Alle \textbf{fett} gedruckten
  Seitenzahlen sind Referenzen auf die Definition
  des jeweiligen Begriffs. Demgegenüber geben
  normal gedruckte Seitenzahlen die Seiten der
  Verwendung des jeweiligen Begriffs wieder.\par
  \bigskip}
```

Bitte beachten Sie, dass für die erste Seite des Index der Seitenstil umgeschaltet wird. Welcher Seitenstil hierbei Verwendung findet, ist im Makro `\indexpagestyle` abgelegt (siehe Abschnitt 3.12, Seite 78).

Für die Erstellung, Sortierung und Ausgabe des Stichwortverzeichnisses sind die üblichen Standard-LaTeX-Pakete und Zusatzprogramme zuständig. Von KOMA-Script werden genau wie von den Standardklassen lediglich die grundlegenden Makros und Umgebungen dafür zur Verfügung gestellt.

Die Briefklasse scrlttr2

Briefe sind in vielerlei Hinsicht etwas ganz anderes als Artikel, Berichte, Bücher oder Ähnliches. Schon allein deshalb gibt es für die Briefklasse ein eigenes Kapitel. Aber auch aus einem anderen Grund ist ein eigenes Kapitel für scrlttr2 gerechtfertigt. Die Klasse wurde von Grund auf neu entwickelt. Sie hat daher auch ein komplett anderes Bedienkonzept als alle anderen mir bekannten Klassen. Die neue Art der Bedienung ist möglicherweise etwas ungewohnt, bietet jedoch nicht nur dem geübten Anwender einige Vorteile.

4.1. Variablen

Neben Optionen, Anweisungen (oder Befehlen), Umgebungen, Zählern und Längen wurden in Kapitel 3 für KOMA-Script bereits zusätzlich Elemente eingeführt. Eine typische Eigenschaft eines Elements ist seine Schriftart und die Möglichkeit, diese zu ändern (siehe Abschnitt 4.9, Seite 56). An dieser Stelle werden nun zusätzlich Variablen eingeführt. Variablen haben einen Namen, über den sie angesprochen werden und einen Inhalt. Der Inhalt einer Variablen kann zeitlich bzw. räumlich getrennt von ihrer Verwendung gesetzt werden, so wie der Inhalt einer Anweisung getrennt von ihrer Ausführung definiert werden kann. Ein Hauptunterschied einer Variablen zu einer Anweisung besteht darin, dass eine Anweisung normalerweise eine Aktion auslöst, während der Inhalt einer Variablen normalerweise aus einem Text besteht, der dann von einer Anweisung ausgegeben wird. Außerdem kann eine Variable zusätzlich eine Bezeichnung besitzen, die ebenfalls gesetzt und ausgegeben werden kann.

Dieser Abschnitt beschränkt sich bewusst auf die Einführung des Begriffs der Variablen. Die zur Verdeutlichung verwendeten Beispiele sind ohne tiefere Bedeutung. Konkretere Anwendungsbeispiele gibt es bei der Erläuterung der in der Briefklasse bereits definierten und von ihr verwendeten Variablen in den nachfolgenden Abschnitten. Tabelle 4.1 gibt eine Übersicht über alle in scrlttr2 definierten Variablen.

Tabelle 4.1.: Von der Klasse scrlttr2 unterstützte Variablen

`addresseeimage`	Anweisungen, die zum Setzen des Port-Payé-Kopfes bei der Einstellung `addrfield=backgroundimage` oder der Port-Payé-Anschrift bei der Einstellung `addrfield=image`, verwendet werden (Abschnitt 4.10, Seite 181)
`backaddress`	Rücksendeadresse für Fensterbriefumschläge (Abschnitt 4.10, Seite 181)

...

Tabelle 4.1.: Von der Klasse scrlttr2 unterstützte Variablen (*Fortsetzung*)

`backaddressseparator`
: Trennzeichen innerhalb der Rücksendeadresse (Abschnitt 4.10, Seite 181)

`ccseparator`
: Trennzeichen zwischen Verteilertitel und Verteiler (Abschnitt 4.7, Seite 157)

`customer`
: Geschäftszeilenfeld »Kundennummer« (Abschnitt 4.10, Seite 187)

`date`
: Datum (Abschnitt 4.10, Seite 186)

`emailseparator`
: Trennzeichen zwischen E-Mail-Bezeichnung und E-Mail-Adresse (Abschnitt 4.10, Seite 174)

`enclseparator`
: Trennzeichen zwischen Anlagetitel und Anlagen (Abschnitt 4.7, Seite 158)

`firstfoot`
v3.08
: Seitenfuß des Briefbogens (Abschnitt 4.10, Seite 192)

`firsthead`
v3.08
: Kopf des Briefbogens (Abschnitt 4.10, Seite 179)

`faxseparator`
: Trennzeichen zwischen Faxbezeichner und Faxnummer (Abschnitt 4.10, Seite 174)

`fromaddress`
: Absenderadresse ohne Absendername (Abschnitt 4.10, Seite 169)

`frombank`
: Bankverbindung des Absenders (Abschnitt 4.10, Seite 192)

`fromemail`
: E-Mail-Adresse des Absenders (Abschnitt 4.10, Seite 174)

`fromfax`
: Faxnummer des Absenders (Abschnitt 4.10, Seite 174)

`fromlogo`
: Anweisungen zum Setzen des Absenderlogos (Abschnitt 4.10, Seite 177)

Tabelle 4.1.: Von der Klasse scrlttr2 unterstützte Variablen (*Fortsetzung*)

`fromname`
: vollständiger Absendername (Abschnitt 4.10, Seite 169)

`fromphone`
: Telefonnummer des Absenders (Abschnitt 4.10, Seite 174)

`fromurl`
: eine URL des Absenders (Abschnitt 4.10, Seite 174)

`fromzipcode`
: Postleitzahl des Absenders für den Port-Payé-Kopf bei `addrfield=PP` (Abschnitt 4.10, Seite 181)

`invoice`
: Geschäftszeilenfeld »Rechnungsnummer« (Abschnitt 4.10, Seite 187)

`location`
: erweiterte Absenderangabe (Abschnitt 4.10, Seite 184)

`myref`
: Geschäftszeilenfeld »Mein Zeichen« (Abschnitt 4.10, Seite 187)

`nextfoot`
v3.08
: Seitenfuß im Seitenstil `headings` oder `myheadings` (Abschnitt 4.13, Seite 198)

`nexthead`
v3.08
: Kopf im Seitenstil `headings` oder `myheadings` (Abschnitt 4.13, Seite 198)

`phoneseparator`
: Trennzeichen zwischen Telefonbezeichner und Telefonnummer (Abschnitt 4.10, Seite 174)

`place`
: Ort (Abschnitt 4.10, Seite 181)

`placeseparator`
: Trennzeichen zwischen Ort und Datum (Abschnitt 4.10, Seite 188)

`PPdatamatrix`
: Anweisungen zum Setzen einer Data-Matrix bei der Einstellung `addrfield=PP` (Abschnitt 4.10, Seite 181)

...

Tabelle 4.1.: Von der Klasse scrlttr2 unterstützte Variablen (*Fortsetzung*)

`PPcode`
: Code zur Identifizierung des Absenders bei Einstellung `addrfield=PP` (Abschnitt 4.10, Seite 181)

`signature`
: Signatur unter Unterschrift und Grußformel (Abschnitt 4.20, Seite 200)

`specialmail`
: Versandart (Abschnitt 4.10, Seite 181)

`subject`
: Betreff (Abschnitt 4.10, Seite 190)

`subjectseparator`
: Trennzeichen zwischen Betrefftitel und Betreff (Abschnitt 4.10, Seite 190)

`title`
: Brieftitel (Abschnitt 4.10, Seite 190)

`toaddress`
: Empfängeradresse ohne Empfängername (Abschnitt 4.10, Seite 181)

`toname`
: vollständiger Empfängername (Abschnitt 4.10, Seite 181)

`yourmail`
: Geschäftszeilenfeld »Ihr Schreiben« (Abschnitt 4.10, Seite 187)

`yourref`
: Geschäftszeilenfeld »Ihr Zeichen« (Abschnitt 4.10, Seite 187)

`zipcodeseparator`
: Trennzeichen zwischen der Bezeichnung und dem Inhalt der Variablen `fromzipcode` (Abschnitt 4.10, Seite 181)

`\setkomavar{`*`Name`*`}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`
`\setkomavar*{`*`Name`*`}{`*`Bezeichnung`*`}`

Mit der Anweisung `\setkomavar` wird der *`Inhalt`* der Variablen *`Name`* gesetzt. Dabei kann per optionalem Argument gleichzeitig auch die *`Bezeichnung`* der Variablen geändert werden. Demgegenüber kann mit der Sternvariante `\setkomavar*` auch nur die *`Bezeichnung`* der Variablen *`Name`* gesetzt werden.

Beispiel: In Briefen ist es üblich, den Absender im Briefkopf stehen zu haben. Dazu muss scrlttr2 den Absender aber erst einmal mit Namen kennen. Für »Peter Musterfrau« ginge das einfach mit:

```
\setkomavar{fromname}{Peter Musterfrau}
```

Die voreingestellte Bezeichnung für den Namen des Absenders ist »Von«. Angenommen, Herr Musterfrau will aber an den Stellen, an denen scrlttr2 diese Bezeichnung verwendet, lieber »Absender« haben, so müsste er zusätzlich

```
\setkomavar*{fromname}{Absender}
```

setzen oder aber die beiden Angaben zu einer Anweisung zusammenfassen:

```
\setkomavar{fromname}[Absender]{Peter Musterfrau}
```

Damit schlägt er sozusagen zwei Fliegen mit einer Klappe.

Übrigens kann mit einem leeren obligatorischen Argument *Inhalt* der Inhalt der Variable gelöscht werden. Selbstverständlich kann in gleicher Weise mit einem leeren Argument *Bezeichnung* auch die Bezeichnung der Variablen gelöscht werden.

Beispiel: Angenommen, Herr Musterfrau will gar keine Bezeichnung für den Namen des Absenders haben. Dann könnte er diese entweder für sich mit:

```
\setkomavar*{fromname}{}
```

löschen. Er könnte aber auch wieder zwei Fliegen mit einer Klappe schlagen und

```
\setkomavar{fromname}[]{Peter Musterfrau}
```

verwenden. Dadurch wird gleichzeitig der Inhalt der Variablen gesetzt und ihre Bezeichnung gelöscht.

`\usekomavar[`*Anweisung*`]{`*Name*`}`
`\usekomavar*[`*Anweisung*`]{`*Name*`}`

v2.9i

In manchen Fällen wird es notwendig sein, selbst auf den Inhalt oder die Bezeichnung einer Variablen zuzugreifen, dies also nicht allein der Klasse zu überlassen. Das gilt insbesondere dann, wenn Sie eigene Variablen definiert haben, die nicht zur Geschäftszeile hinzugefügt werden. Mit der Anweisung `\usekomavar` können Sie auf den Inhalt der Variablen *Name* zugreifen, während Sie mit der Sternvariante `\usekomavar*` ihre Bezeichnung erhalten. Näheres zur Definition eigener Variablen ist Abschnitt 17.2, Seite 331 zu entnehmen.

`\ifkomavar{`*Name*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

v3.03

Mit dieser Anweisung kann man feststellen, ob eine Variable definiert ist. Der *Dann-Teil* wird nur dann ausgeführt, wenn die Variable existiert. Dabei wird der Inhalt der Variablen nicht getestet, kann also auch leer sein. Der *Sonst-Teil* wird hingegen ausgeführt, wenn die Variable nicht existiert. Solche Tests können beispielsweise dann sinnvoll sein, wenn eigene

Variablen in einer lco-Datei (siehe Abschnitt 4.21 ab Seite 201) definiert werden und in einer anderen lco-Datei diese Variable nur dann verwendet werden soll, wenn sie existiert.

```
\ifkomavarempty{Name}{Dann-Teil}{Sonst-Teil}
\ifkomavarempty*{Name}{Dann-Teil}{Sonst-Teil}
```

v2.9i

Mit Hilfe dieser Anweisungen kann man feststellen, ob der Inhalt oder die Bezeichnung einer Variablen leer ist oder nicht. Der *Dann-Teil* wird nur dann ausgeführt, wenn der expandierte Inhalt oder die expandierte Bezeichnung der Variablen *Name* leer ist. Anderenfalls wird der *Sonst-Teil* ausgeführt. Die Sternvariante der Anweisung bezieht sich dabei auf die Bezeichnung der Variablen, während die normale Variante den Inhalt behandelt.

4.2. Pseudolängen

Längen werden bei LaTeX mit den Anweisungen \newlength, \setlength, \addtolength und \the*Länge* verarbeitet. Sehr viele Pakete nutzen aber auch Makros, also Anweisungen, um Längen zu speichern. KOMA-Script erweitert dieses Verfahren um die Möglichkeit, solche in Makros gespeicherten Längen mit ähnlichen Anweisungen zu verarbeiten wie echte Längen. Diese eigentlich in Makros abgelegten Längen heißen bei KOMA-Script daher Pseudolängen.

Eine Liste aller in scrlttr2 definierten Pseudolängen findet sich in Tabelle 17.1, Seite 317. Eine grafische Darstellung der Bedeutungen der wichtigsten Pseudolängen des Briefbogens ist Abbildung 17.1, Seite 321 zu entnehmen. Die verwendeten Maße sind dabei an die Voreinstellungen von scrlttr2 angelehnt. Nähere Beschreibungen zu den einzelnen Pseudolängen finden sich in den einzelnen Abschnitten dieses Kapitels.

Da der Anwender normalerweise keine eigenen Pseudolängen definieren muss, wird dieser Teil nicht hier, sondern im Expertenteil in Abschnitt 17.1, Seite 320 behandelt. Ebenso ist das Setzen von Pseudolängen eher dem fortgeschrittenen Anwender vorbehalten. Also wird auch dies im Abschnitt für Experten ab Seite 322 erklärt.

```
\useplength{Name}
```

Mit Hilfe dieser Anweisung wird auf den Wert der Pseudolänge mit dem angegebenen *Namen* zugegriffen. Dies ist eine der wenigen Benutzeranweisung rund um Pseudolängen. Natürlich kann diese Anweisung dennoch auch innerhalb einer lco-Datei (siehe Abschnitt 4.21 ab Seite 201) verwendet werden.

```
\setlengthtoplength[Faktor]{Länge}{Pseudolänge}
\addtolengthplength[Faktor]{Länge}{Pseudolänge}
```

Während man einer Länge einfach einen Faktor voranstellen kann, ist dies bei Pseudolängen nicht möglich. Angenommen, eine Länge \TestLength hat den Wert 2 pt, dann ergibt 3\TestLength den Wert 6 pt. Verwendet man stattdessen eine Pseudolänge, so würde aus 3\useplength{Test}

der Wert 32 pt. Dies ist insbesondere dann lästig, wenn man einer echten *Länge* den Wert einer *Pseudolänge* zuweisen will.

Mit der Anweisung `\setlengthtoplength` kann man einer echten *Länge* das Vielfache einer *Pseudolänge* zuweisen. Allerdings wird hier ein *Faktor* nicht direkt der *Pseudolänge* vorangestellt, sondern als optionales Argument übergeben. Man sollte diese Anweisung auch verwenden, wenn man einer *Länge* den negativen Wert einer *Pseudolänge* zuweisen will. Als *Faktor* kann dann wahlweise ein Minuszeichen oder `-1` verwendet werden. Die Anweisung `\addtolengthplength` arbeitet ganz ähnlich. Allerdings wird hier zur *Länge* das Vielfache der *Pseudolänge* addiert.

4.3. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde.

4.4. Kompatibilität zu früheren Versionen von KOMA-Script

Es gilt sinngemäß, was in Abschnitt 2.5 geschrieben wurde. Allerdings existiert diese Möglichkeit bei scrlttr2 bereits seit Version 2.9t.

4.5. Entwurfsmodus

Es gilt sinngemäß, was in Abschnitt 3.3 geschrieben wurde.

4.6. Seitenaufteilung

Eine Dokumentseite besteht aus unterschiedlichen Teilen, wie den Rändern, dem Kopf, dem Fuß, dem Textbereich, einer Marginalienspalte und den Abständen zwischen diesen Elementen. KOMA-Script unterscheidet dabei auch noch zwischen der Gesamtseite oder dem Papier und der sichtbaren Seite. Ohne Zweifel gehört die Aufteilung der Seite in diese unterschiedlichen Teile zu den Grundfähigkeiten einer Klasse. Bei KOMA-Script wird diese Arbeit an das Paket typearea delegiert. Dieses Paket kann auch zusammen mit anderen Klassen verwendet werden. Die KOMA-Script-Klassen laden typearea jedoch selbständig. Es ist daher weder notwendig noch sinnvoll, das Paket bei Verwendung einer KOMA-Script-Klasse auch noch explizit per `\usepackage` zu laden. Siehe hierzu auch Abschnitt 4.3.

Einige Einstellungen der KOMA-Script-Klassen haben auch Auswirkungen auf die Seitenaufteilung und umgekehrt. Diese Auswirkungen werden bei den entsprechenden Einstellungen dokumentiert.

Für die weitere Erklärung zur Wahl des Papierformats, der Aufteilung der Seite in Ränder und Satzspiegel und die Wahl von ein- oder zweispaltigem Satz sei auf die Anleitung des Pakets typearea verwiesen. Diese ist in Kapitel 2 ab Seite 25 zu finden.

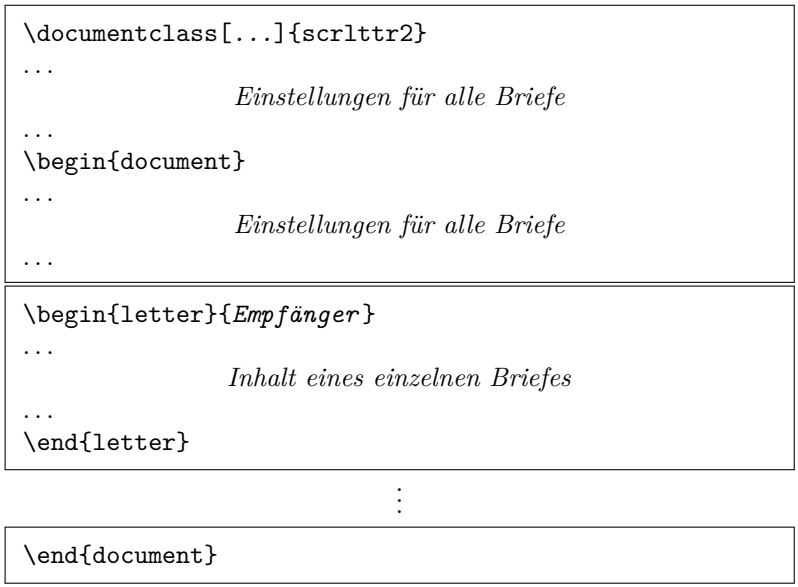


Abbildung 4.1.: Genereller Aufbau eines Briefdokuments mit beliebig vielen einzelnen Briefen (den Aufbau eines einzelnen Briefs zeigt Abbildung 4.2)

Die Unterscheidung zwischen ein- und doppelseitigem Satz ist bei Briefen jedoch in der Regel nicht sinnvoll. Da Briefe normalerweise nicht gebunden werden, betrachtet man bei Briefen jede Seite für sich. Das gilt auch dann, wenn ausnahmsweise Vorder- oder Rückseite bedruckt werden. Daher spielt bei Briefen normalerweise auch der vertikale Ausgleich keine Rolle. Sollten Sie diesen trotzdem benötigen oder wissen wollen, was das ist, sei auf die in Abschnitt 3.4, Seite 53 erklärten Anweisungen `\raggedbottom` und `\flushbottom` verwiesen.

4.7. Genereller Aufbau eines Briefdokuments

Der generelle Aufbau eines Briefdokuments weicht etwas vom Aufbau eines normalen Dokuments ab. Während ein Buchdokument normalerweise nur ein Buch enthält, kann ein einzelnes Briefdokument mehrere Briefe enthalten. Wie in Abbildung 4.1 veranschaulicht wird, besteht ein Briefdokument aus einem Vorspann, den einzelnen Briefen und dem Abschluss.

Der Vorspann beinhaltet dabei alle Einstellungen, die generell alle Briefe betreffen. Diese können in den Einstellungen der einzelnen Briefe jedoch zumindest teilweise überschrieben werden. Die einzige Einstellung, die derzeit nicht innerhalb eines einzelnen Briefes überschrieben werden kann, ist die Version von scrlttr2, zu der Kompatibilität erreicht werden soll (siehe Option `version` in Abschnitt 4.4, Seite 31).

Ich empfehle, vor `\begin{document}` nur allgemeine Einstellungen wie das Laden von Paketen und das Setzen von Optionen vorzunehmen. Alle Einstellungen, die das Setzen einer Variablen oder sonstige Textangaben beinhalten, sollten nach `\begin{document}` vorgenommen werden. Dies empfiehlt sich umso mehr, wenn das Babel-Paket (siehe [Bra01]) verwendet

```
\begin{letter}[Optionen]{Empfänger}
...
                Einstellungen für diesen Brief
...
\opening{Anrede}
```

```
...
                        Brieftext
...
```

```
\closing{Grußformel}
\ps
...
                        Postscriptum
...
\encl{Anlagen}
\cc{Verteiler}
\end{letter}
```

Abbildung 4.2.: Genereller Aufbau eines einzelnen Briefes innerhalb eines Briefdokuments (siehe Abbildung 4.1)

wird oder sprachabhängige Variablen von scrlttr2 verändert werden sollen.

Der Abschluss besteht in der Regel nur aus `\end{document}`. Natürlich können Sie dort aber auch zusätzliche Kommentare einfügen.

Wie in Abbildung 4.2 verdeutlicht wird, bestehen die einzelnen Briefe wiederum aus einer Einleitung, dem eigentlichen Brieftext und einem Schlussteil. In der Einleitung werden alle Einstellungen vorgenommen, die nur für diesen einen Brief gelten sollen. Entscheidend ist hierbei, dass diese Einleitung immer mit `\opening` endet. Ebenso beginnt der Schlussteil immer mit `\closing`. Gegebenenfalls können die Argumente *Anrede* und *Grußformel* der beiden Anweisungen leer bleiben, die Anweisungen müssen jedoch gesetzt werden und haben immer ein Argument.

Es soll an dieser Stelle nicht verschwiegen werden, dass zwischen den einzelnen Briefen weitere Einstellungen getroffen werden können. Diese gelten dann für alle nachfolgenden Briefe. Um Briefdokumente übersichtlich und wartbar zu halten, sollte man sich jedoch gut überlegen, ob man zwischen die Briefe tatsächlich weitere generelle Einstellungen mit beschränkter Gültigkeit setzen will. Ich kann dies nicht empfehlen.

```
\begin{letter}[Optionen]{Empfänger}
  ...
\end{letter}
```

Die Briefumgebung `letter` ist einer der zentralen Dreh- und Angelpunkte der Briefklasse. Als Besonderheit kann man bei scrlttr2 der Briefumgebung zusätzliche *Optionen* mit auf den Weg geben. Diese werden dann intern per `\KOMAoptions`-Anweisung ausgeführt.

Der *Empfänger* wird als obligatorischer Parameter an die Umgebung übergeben. Dabei dient der doppelte Backslash als Trennzeichen zwischen einzelnen Teilen der Anschrift. Diese einzelnen Teile werden im Anschriftfeld als einzelne Zeilen ausgegeben. Dennoch sollte der doppelte Backslash hier nicht als fester Zeilenumbruch verstanden werden. Absätze, vertikaler Leerraum und Ähnliches sind in der Anschrift nicht erlaubt. Sie können zu unerwarteten Effekten und Fehlermeldungen führen. Dies ist übrigens bei der Standardbriefklasse genauso.

Beispiel: Angenommen, jemand wollte einen Brief an Petra Mustermann schreiben. Ein minimalistisches Briefdokument dafür würde so aussehen:

```
\documentclass[version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen}
\end{letter}
\end{document}
```

Allerdings würde dabei noch keinerlei Ausgabe entstehen. Es würde noch nicht einmal die Anschrift auf dem Briefbogen ausgegeben. Warum das so ist, erfahren Sie bei der Erklärung zur Anweisung `\opening` auf Seite 154.

`\AtBeginLetter{`*Anweisungen*`}`
`\AtEndLetter{`*Anweisungen*`}`

Wie in [Tea06] erwähnt, gibt es bei LaTeX die Möglichkeit, zu bestimmten Gelegenheiten während des LaTeX-Laufs eines Dokuments zusätzliche *Anweisungen* ausführen zu lassen. Zu diesem Zweck stellt der LaTeX-Kern beispielsweise die Anweisungen `\AtEndOfClass` und `\AtBeginDocument` zur Verfügung. Man nennt solche Eingriffspunkte auch *hooks*, also Haken.

v2.95

Die Klasse scrlttr2 fügt zwei weitere Haken hinzu, die mit `\AtBeginLetter` und `\AtEndLetter` mit Inhalt versehen werden können. Wie man schon daran erkennt, dass die LaTeX-Kern-Anweisungen für Haken nicht in [Tea05b] sondern in [Tea06] dokumentiert sind, sind diese Anweisungen eigentlich eher für Paket- und Klassenautoren gedacht. Bei der Briefklasse kann es jedoch sinnvolle Anwendungen für die beiden neuen Haken auch auf Benutzerebene geben. Das folgende Beispiel zeigt dies.

Beispiel: Angenommen, Sie haben mehrere Briefe in einem Dokument. Sie verwenden außerdem eine eigene Anweisung, um in den Briefen einen Fragebogen zu setzen. Dabei werden die Fragen automatisch mit Hilfe eines Zählers nummeriert. Da scrlttr2 dieser Zähler nicht bekannt ist, würde er auch im Gegensatz etwa zur Seitenzahl am Anfang eines neuen Briefes nicht zurückgesetzt. Wenn jeder Brief zehn Fragen beinhaltet, hätte damit die erste Frage im fünften Brief die Nummer 41 statt der Nummer 1. Sie lösen das, indem Sie scrlttr2 mitteilen, dass am Anfang jedes Briefes der Zähler zurückgesetzt werden soll:

```
\newcounter{Frage}
\newcommand{\Frage}[1]{%
  \refstepcounter{Frage}\par
  \noindent\begin{tabularx}{\textwidth}{l@{}X}
    \theFrage:~ & #1\\
  \end{tabularx}%
}%
\AtBeginLetter{\setcounter{Frage}{0}}
```

Damit hat dann auch die erste Frage im 1001. Brief wieder die Nummer Eins. Die hier angegebene Definition benötigt übrigens das tabularx-Paket (siehe [Car99c]).

`\opening{`*Anrede*`}`

Dies ist eine der wichtigsten Anweisungen in scrlttr2. Vordergründig wird damit die *Anrede* des Briefes, beispielsweise »Sehr geehrte Frau . . . «, gesetzt. Tatsächlich setzt diese Anweisung aber auch alle Elemente des Briefbogens wie die Faltmarken, den Briefkopf, die Anschrift, die Absenderergänzung, die Geschäftszeile, den Titel, den Betreff und den Seitenfuß. Kurz gesagt: ohne Anrede kein Brief. Soll tatsächlich einmal ein Brief ohne Anrede gesetzt werden, so muss eben das Argument von `\opening` leer bleiben.

Beispiel: Kommen wir auf das Beispiel von Seite 153 zurück. Wird dieses um eine Anrede ergänzt, dann ergibt sich aus

```
\documentclass[version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
\end{letter}
\end{document}
```

der Briefbogen von Abbildung 4.3.

`\closing{`*Grußfloskel*`}`

Mit der Anweisung `\closing` wird in erster Linie die *Grußfloskel* gesetzt. Diese kann auch mehrzeilig sein. Die einzelnen Zeilen sollten dann mit doppeltem Backslash voneinander getrennt werden. Absätze innerhalb der *Grußfloskel* sind jedoch nicht gestattet.

Darüber hinaus setzt diese Anweisung aber auch noch gleich den Inhalt der Variablen `signature` als Signatur. Näheres zur Signatur und deren Konfiguration ist Abschnitt 4.20 ab Seite 200 zu entnehmen.

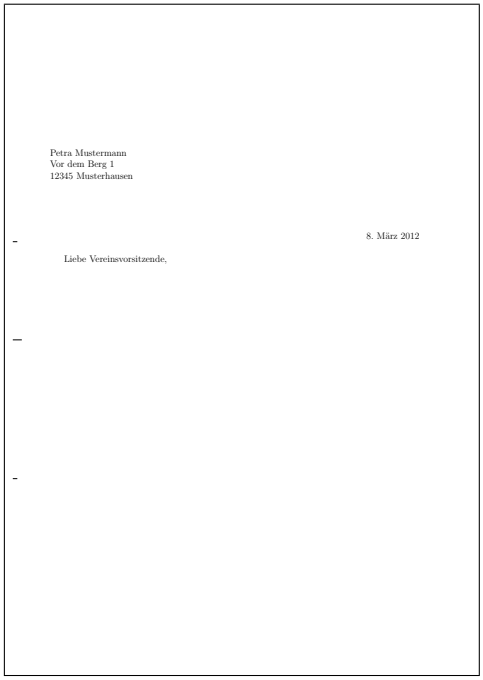


Abbildung 4.3.: Ergebnis eines minimalistischen Briefes nur mit Anschrift und Anrede (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

Beispiel: Erweitern wir unser Beispiel um einige Zeilen Brieftext und eine Grußfloskel zu:

```
\documentclass[version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\end{letter}
\end{document}
```

Damit sieht das Ergebnis wie in Abbildung 4.4 aus.

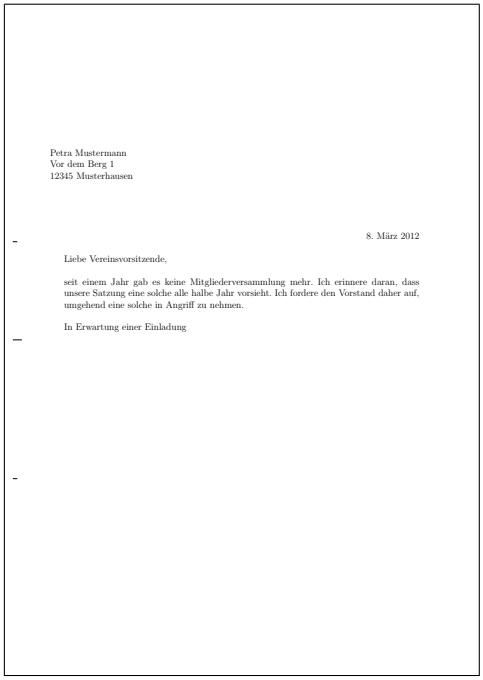


Abbildung 4.4.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text und Grußfloskel (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

`\ps`

Diese Anweisung schaltet lediglich auf das Postskriptum um. Dazu wird ein neuer Absatz begonnen und ein vertikaler Abstand – in der Regel zur Signatur – eingefügt. Auf die Anweisung `\ps` kann beliebiger Text folgen. Dabei muss der Anwender auch selbst entscheiden, ob er den Nachsatz etwa mit der Abkürzung »PS:«, die übrigens ohne Punkt gesetzt wird, beginnen will. Die Klasse scrlttr2 setzt diese Abkürzung weder automatisch noch optional.

Beispiel: Unser Beispielbrief, um ein Postskriptum erweitert,

```
\documentclass[version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
```

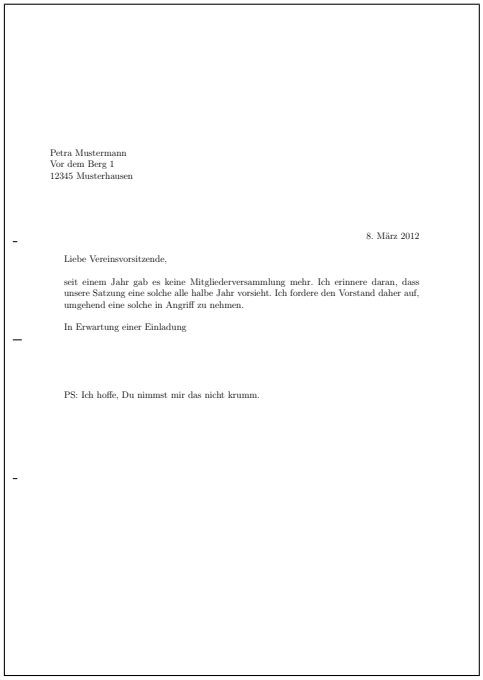


Abbildung 4.5.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel und Postskriptum (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

```
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\end{letter}
\end{document}
```

sieht dann wie in Abbildung 4.5 aus.

Als Briefe noch von Hand geschrieben wurden, war das Postskriptum sehr beliebt. Es handelte sich bei diesen Nachsätzen ursprünglich um Angaben, die im eigentlichen Brief vergessen wurden. Bei Briefen, die mit LaTeX geschrieben werden, ist es natürlich einfach, Vergessenes nachträglich in den Brief einzuarbeiten. Trotzdem ist das Postskriptum noch immer sehr beliebt, kann man damit doch sehr schön noch einmal auf ganz andere äußerst wichtige oder eigentlich ganz unwichtige Dinge hinweisen.

```
\cc{Verteiler}
\setkomavar{ccseparator}[Bezeichnung]{Inhalt}
```

Ein *Verteiler* kann mit der Anweisung \cc gesetzt werden. Der *Verteiler* wird der Anweisung dabei als Argument übergeben. Wenn der *Inhalt* der Variablen ccseparator nicht leer ist, wird dem *Verteiler* die *Bezeichnung* und der *Inhalt* dieser Variablen vorangestellt. Der *Verteiler* selbst wird dann um die entsprechende Breite eingerückt ausgegeben.

Es empfiehlt sich, den *Verteiler* `\raggedright` zu setzen und die einzelnen Angaben durch doppelten Backslash voneinander zu trennen.

Beispiel: Der Beispielbrief soll dieses Mal nicht nur an die Vorsitzende, sondern mit Verteiler auch an alle Mitglieder des Vereins gehen:

```
\documentclass[version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ist in Abbildung 4.6 zu sehen.

Vor dem Verteiler wird automatisch ein Abstand eingefügt.

```
\encl{Anlagen}
\setkomavar{enclseparator}[Bezeichnung]{Inhalt}
```

Die *Anlagen* sind genauso aufgebaut wie der Verteiler. Der einzige Unterschied besteht darin, dass die Einleitung hier von der *Bezeichnung* und dem *Inhalt* der Variablen `enclseparator` bestimmt wird.

Beispiel: Dem Beispielbrief wird nun als Anlage noch ein Auszug aus der Satzung beigefügt. Da es nur eine Anlage gibt, wird auch die voreingestellte Bezeichnung passend geändert:

```
\documentclass[version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
```

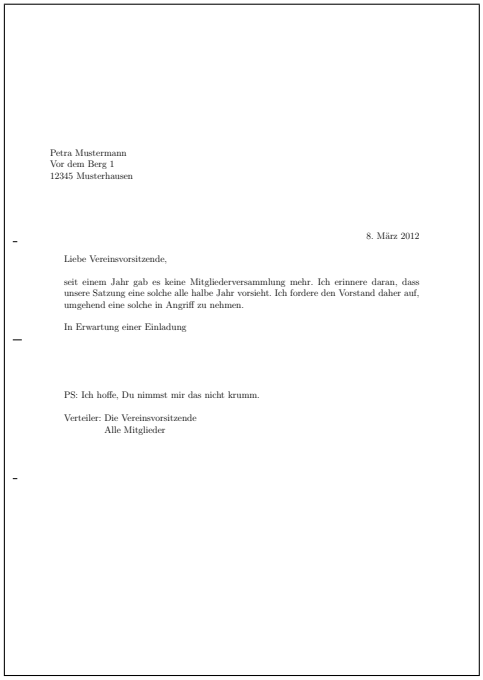
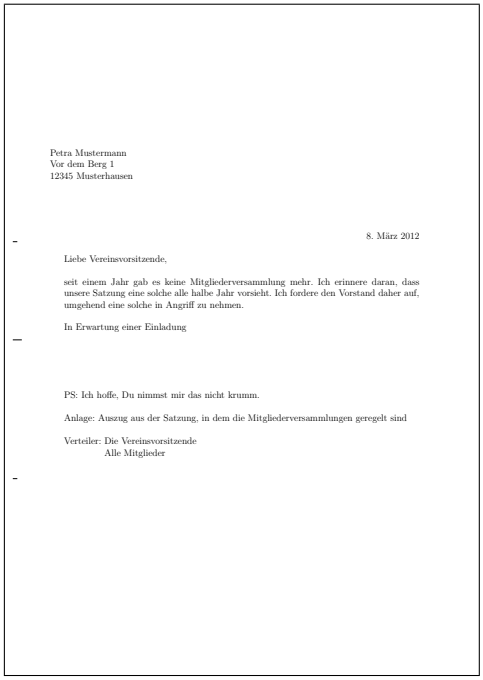


Abbildung 4.6.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum und Verteiler (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

```
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ist in Abbildung 4.7 zu sehen.


Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

8. März 2012

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

In Erwartung einer Einladung

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder

Abbildung 4.7.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen und Verteiler (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

4.8. Wahl der Schriftgröße für das Dokument oder einen Brief

Die Grundschrift und deren Größe sind zentrale Elemente der Gestaltung eines Dokuments. Wie in Kapitel 2 ausgeführt wurde, hängt die Aufteilung zwischen Satzspiegel und Rändern wesentlich davon ab. Die Grundschrift ist dabei die Schrift, die für die Masse des Textes eines Dokuments verwendet wird. Alle davon abweichenden Einstellungen, sei es in der Form, der Dicke, der Neigung oder der Größe, stehen in einer Beziehung zur Grundschrift.

`fontsize=`*`Größe`*

Während von den Standardklassen und den meisten anderen Klassen nur eine sehr beschränkte Anzahl an Schriftgrößen unterstützt wird, bietet KOMA-Script die Möglichkeit, jede beliebige *`Größe`* für die Grundschrift anzugeben. Dabei kann als Einheit für die *`Größe`* auch jede bekannte TEX-Einheit verwendet werden. Wird die *`Größe`* ohne Einheit angegeben, so wird `pt` als Einheit angenommen.

Wird die Option innerhalb des Dokuments gesetzt, so werden ab diesem Punkt die Grundschriftgröße und alle davon abhängigen Größen geändert. Das kann beispielsweise dann nützlich sein, wenn ein weiterer Brief insgesamt in einer kleineren Schriftgröße gesetzt werden soll. Es wird darauf hingewiesen, dass bei Verwendung nach dem Laden der Klasse die Aufteilung zwischen Satzspiegel und Rändern nicht automatisch neu berechnet wird (siehe `\recalctypearea`, Abschnitt 2.4, Seite 38). Wird diese Neuberechnung jedoch vorgenommen,

so erfolgt sie auf Basis der jeweils gültigen Grundschriftgröße. Die Auswirkungen des Wechsels der Grundschriftgröße auf zusätzlich geladene Pakete sind von diesen Paketen abhängig. Dabei sind Fehler möglich, die nicht als Fehler von KOMA-Script betrachtet werden.

Diese Option sollte keinesfalls als Ersatz für `\fontsize` (siehe [Tea05a]) missverstanden werden. Sie sollte auch nicht an Stelle einer der von der Grundschrift abhängigen Schriftgrößenanweisungen, `\tiny` bis `\Huge`, verwendet werden! Bei scrlttr2 ist `fontsize=12pt` voreingestellt.

Beispiel: Angenommen, bei dem Verein aus dem Beispielbrief handelt es sich um die *»Freunde ungesunder Schriftgrößen«*, weshalb er in 14 pt statt in 12 pt gesetzt werden soll. Dies kann durch eine kleine Änderung der ersten Zeile erreicht werden:

```
\documentclass[version=last,fontsize=14pt]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Alternativ könnte die Option auch als optionales Argument von `letter` gesetzt werden:

```
\documentclass[version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}[fontsize=14pt]{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
```

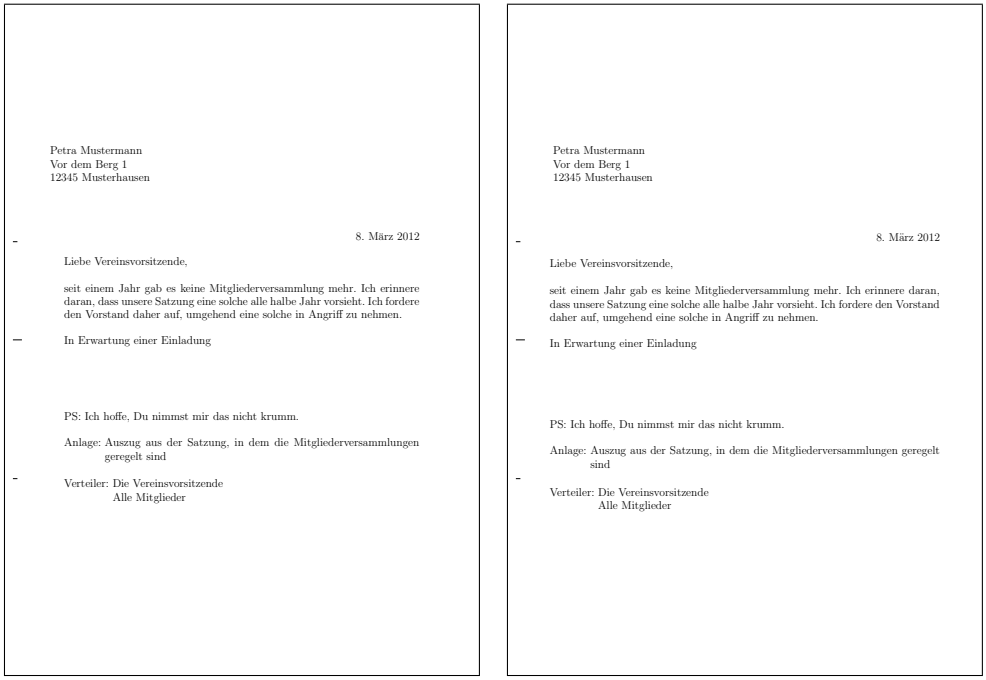


Abbildung 4.8.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen, Verteiler und ungesund großer Schrift (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe); links wurde die Schriftgröße als optionales Argument von `letter` gesetzt, rechts als optionales Argument von `\documentclass`

```
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Da bei dieser späten Änderung der Schriftgröße der Satzspiegel nicht geändert wird, unterscheiden sich die beiden Ergebnisse in Abbildung 4.8.

4.9. Textauszeichnungen

Es gilt sinngemäß, was in Abschnitt 3.6 geschrieben wurde. Dabei sind Namen und Bedeutung der einzelnen Elemente in Tabelle 4.2 aufgelistet. Die Voreinstellungen sind den jeweiligen Abschnitten zu entnehmen.

Mit der Anweisung `\usekomafont` kann die aktuelle Schriftart auf diejenige umgeschaltet werden, die für das angegebene *`Element`* definiert ist.

Ein allgemeines Beispiel für die Anwendung von `\setkomafont` und `\usekomafont` finden Sie in Abschnitt 4.9, Seite 56.

Tabelle 4.2.: Elemente, deren Schrift bei der Klasse scrlttr2 mit `\setkomafont` und `\addtokomafont` verändert werden kann

`addressee`
: Name und Anschrift im Anschriftfenster (Abschnitt 4.10, Seite 181)

`backaddress`
: Rücksendeadresse für einen Fensterbriefumschlag (Abschnitt 4.10, Seite 181)

`descriptionlabel`
: Label, also das optionale Argument von `\item`, in einer `description`-Umgebung (Abschnitt 4.16, **??**)

`foldmark`
: Faltmarke auf dem Briefpapier; ermöglicht Änderung der Linienfarbe (Abschnitt 4.10, Seite 165)

`footnote`
: Marke und Text einer Fußnote (Abschnitt 4.15, Seite 83)

`footnotelabel`
: Marke einer Fußnote; Anwendung erfolgt nach dem Element `footnote` (Abschnitt 4.15, Seite 83)

`footnotereference`
: Referenzierung der Fußnotenmarke im Text (Abschnitt 4.15, Seite 83)

`footnoterule`
v3.07
: Linie über dem Fußnotenapparat (Abschnitt 4.15, Seite 86)

`fromaddress`
: Absenderadresse im Briefkopf (Abschnitt 4.10, Seite 169)

...

Tabelle 4.2.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`fromname`
: Name des Absenders im Briefkopf abweichend von `fromaddress` (Abschnitt 4.10, Seite 169)

`fromrule`
: Linie im Absender im Briefkopf; gedacht für Farbänderungen (Abschnitt 4.10, Seite 169)

`pagefoot`
: wird nach dem Element `pageheadfoot` für den mit Variable `nextfoot` definieren Seitenfuß verwendet oder wenn das Paket scrpage2 geladen ist (Kapitel 5, Seite 220)

`pagehead`
: alternative Bezeichnung für `pageheadfoot`

`pageheadfoot`
: Seitenkopf und Seitenfuß bei allen von KOMA-Script definierten Seitenstilen (Abschnitt 4.13, Seite 196)

`pagenumber`
: Seitenzahl im Kopf oder Fuß der Seite (Abschnitt 4.13, Seite 196)

`pagination`
: alternative Bezeichnung für `pagenumber`

`refname`
: Bezeichnung der Felder in der Geschäftszeile (Abschnitt 4.10, Seite 187)

`refvalue`
: Werte der Felder in der Geschäftszeile (Abschnitt 4.10, Seite 187)

`specialmail`
: Versandart im Anschriftfenster (Abschnitt 4.10, Seite 181)

`subject`
: Betreff in der Brieferöffnung (Abschnitt 4.10, Seite 190)

`title`
: Titel in der Brieferöffnung (Abschnitt 4.10, Seite 190)

...

Tabelle 4.2.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`toaddress`	Abweichung vom Element `addressee` für die Anschrift (ohne Name) des Empfängers im Anschriftfeld (Abschnitt 4.10, Seite 181)
`toname`	Abweichung vom Element `addressee` für den Namen des Empfängers im Anschriftfeld (Abschnitt 4.10, Seite 181)

4.10. Briefbogen

Der Briefbogen ist die erste Seite und damit das Aushängeschild jedes Briefes. Im geschäftlichen Bereich handelt es sich dabei oft um einen Vordruck, auf dem viele Elemente wie ein Briefkopf mit Absenderinformationen und Logo bereits enthalten sind. Bei KOMA-Script sind diese Elemente frei positionierbar. Damit ist es nicht nur möglich, einen Briefbogen direkt nachzubilden, sondern auch vorgesehene Felder wie die Anschrift unmittelbar auszufüllen. Die freie Positionierbarkeit wird über Pseudolängen (siehe Abschnitt 4.2 ab Seite 149) erreicht. Eine schematische Darstellung des Briefbogens und der dafür verwendeten Variablen ist in Abbildung 4.9 zu finden. Dabei sind die Namen der Variablen zur besseren Unterscheidung von Anweisungen und deren Argumenten fett gedruckt.

Folgeseiten sind vom Briefbogen zu unterscheiden. Folgeseiten im Sprachgebrauch dieser Anleitung sind alle Briefseiten abgesehen von der ersten.

`foldmarks=`*`Einstellung`*

Falt- oder Falzmarken sind kleine horizontale Striche am linken und kleine vertikale Striche am oberen Rand. KOMA-Script unterstützt für den Briefbogen derzeit drei konfigurierbare horizontale und eine konfigurierbare vertikale Faltmarke. Dazu wird noch eine horizontale Loch- oder Seitenmittenmarke unterstützt, die nicht in der Vertikalen verschoben werden kann.

Mit der Option `foldmarks` können Faltmarken für eine vertikale Zwei-, Drei- oder Vierteilung und eine horizontale Zweiteilung aktiviert oder deaktiviert werden. Die einzelnen Teile müssen dabei nicht äquidistant sein. Die Positionen von drei der vier horizontalen und der vertikalen Marke sind über Pseudolängen konfigurierbar (siehe Abschnitt 17.1.1 ab Seite 323).

v2.97e

Über die Option `foldmarks` können entweder mit den Standardwerten für einfache Schalter, die in Tabelle 2.5, Seite 40 angegeben sind, alle konfigurierten Faltmarken am linken und oberen Rand ein- und ausgeschaltet werden, oder es kann durch die Angabe eines oder mehrerer Buchstaben aus Tabelle 4.3 die Verwendung der einzelnen Faltmarken gezielt konfiguriert werden. Auch in diesem Fall werden die Faltmarken nur dann angezeigt, wenn die Faltmarken nicht mit `false`, `off` oder `no` generell abgeschaltet wurden. Die genaue Position

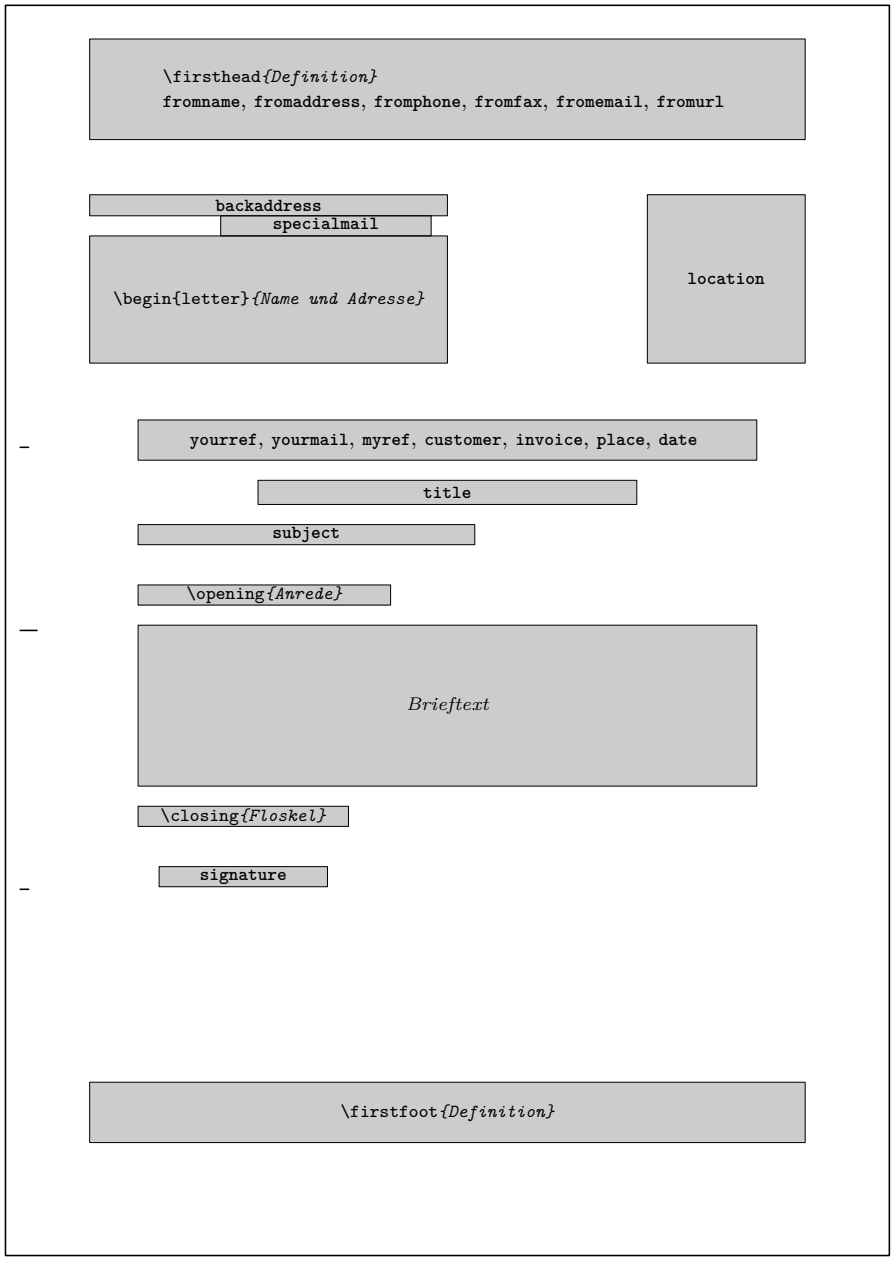


Abbildung 4.9.: Schematische Darstellung des Briefbogens mit den wichtigsten Anweisungen und Variablen für die skizzierten Elemente

Tabelle 4.3.: Kombinierbare Werte für die Konfiguration der Faltmarken mit der Option `foldmarks`

`B`	untere, horizontale Faltmarke am linken Rand aktivieren
`b`	untere, horizontale Faltmarke am linken Rand deaktivieren
`H`	alle horizontalen Faltmarken am linken Rand aktivieren
`h`	alle horizontalen Faltmarken am linken Rand deaktivieren
`L`	linke, vertikale Faltmarke am oberen Rand aktivieren
`l`	linke, vertikale Faltmarke am oberen Rand deaktivieren
`M`	mittlere, horizontale Faltmarke am linken Rand aktivieren
`m`	mittlere, horizontale Faltmarke am linken Rand deaktivieren
`P`	Locher- bzw. Seitenmittenmarke am linken Rand aktivieren
`p`	Locher- bzw. Seitenmittenmarke am linken Rand deaktivieren
`T`	obere, horizontale Faltmarke am linken Rand aktivieren
`t`	obere, horizontale Faltmarke am linken Rand deaktivieren
`V`	alle vertikalen Faltmarken am oberen Rand aktivieren
`v`	alle vertikalen Faltmarken am oberen Rand deaktivieren

der Faltmarken ist von den Einstellungen des Anwenders beziehungsweise der `lco`-Dateien (siehe Abschnitt 4.21 ab Seite 201) abhängig. Voreingestellt sind `true` und `TBMPL`.

Beispiel: Angenommen, Sie wollen alle Faltmarken außer der Lochermarke abschalten. Wenn die Voreinstellung zuvor noch nicht geändert wurde, können Sie das Abschalten wie folgt erreichen:

```
\KOMAoption{foldmarks=blmt}
```

Besteht die Möglichkeit, dass die Voreinstellung bereits geändert wurde, so sollten Sie lieber auf Nummer Sicher gehen. Unser Beispiel ist dann entsprechend abzuändern.

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
```

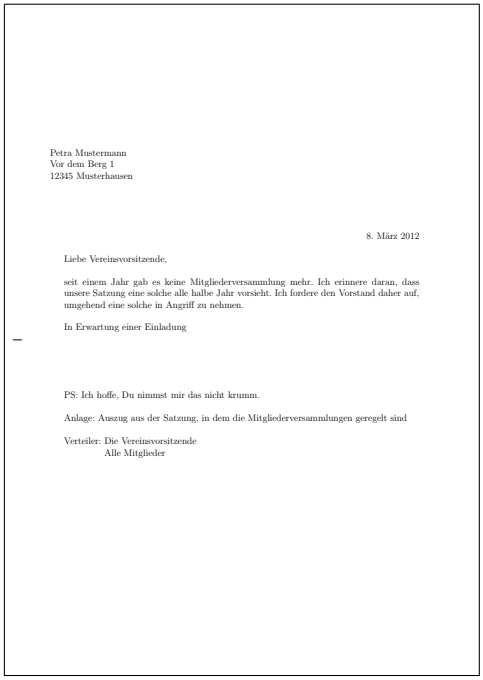


Abbildung 4.10.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe)

```
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ist in Abbildung 4.10 zu sehen.

v2.97c Über das Element `foldmark` kann die Farbe der Faltmarken geändert werden. Dazu werden die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 56) verwendet. Voreingestellt ist keine Änderung.

`enlargefirstpage=`*`Ein-Aus-Wert`*

Die erste Seite eines Briefes fällt immer aus dem normalen Satzspiegel. Von scrlttr2 werden Mechanismen bereitgestellt, um die Höhe und vertikale Ausrichtung von Kopf und Fuß der ersten Seite unabhängig von den Folgeseiten zu bestimmen. Würde dadurch der Fuß der ersten Seite in den Textbereich ragen, so wird der Textbereich der ersten Seite automatisch mit Hilfe von `\enlargethispage` verkleinert. Soll der Textbereich umgekehrt auch vergrößert werden, falls der Fuß der ersten Seite dies erlaubt, so kann das mit dieser Option erreicht werden. Es

passt dann bestenfalls etwas mehr Text auf die erste Seite. Siehe hierzu auch die Erklärung zur Pseudolänge `firstfootvpos` auf Seite 330. Als *Ein-Aus-Wert* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Voreingestellt ist `false`.

`firsthead=`*Ein-Aus-Wert*

v2.97e Das oberste Element eines Briefbogens ist normalerweise der Briefkopf. Bei KOMA-Script kann mit der Option `firsthead` gewählt werden, ob der Briefkopf auf dem Briefbogen überhaupt gesetzt werden soll. Als *Ein-Aus-Wert* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. In der Voreinstellung ist der Briefkopf aktiviert.

`fromalign=`*Methode*

v2.97e Die Option `fromalign` bestimmt, wo der Absender auf der ersten Seite platziert werden soll. Neben verschiedenen Platzierungen im Briefkopf gibt es auch die Möglichkeit, den Absender in der Absenderergänzung unter zu bringen. Gleichzeitig dient diese Option als zentraler Schalter, um die Erweiterungen der Briefkopfgestaltung überhaupt zu aktivieren oder zu deaktivieren. Sind die Erweiterungen deaktiviert, so bleiben die übrigen nachfolgend angegebenen Optionen ohne Wirkung. Mögliche Werte für `fromalign` sind Tabelle 4.4 zu entnehmen. Voreingestellt ist der Wert `left`.

`fromrule=`*Position*
`\setkomavar{fromname}[`*Bezeichnung*`]{`*Inhalt*`}`
`\setkomavar{fromaddress}[`*Bezeichnung*`]{`*Inhalt*`}`

Der Name des Absenders wird über die Variable `fromname` bestimmt. Im Briefkopf wird dabei die *Bezeichnung* (siehe auch Tabelle 4.6, Seite 175) nicht gesetzt.

Optional kann mit Einstellung `fromrule=aftername` im erweiterten Briefkopf auf den Namen eine horizontale Linie folgen. Alternativ kann die Linie mit `fromrule=afteraddress` auch unterhalb des kompletten Absenders gesetzt werden. Eine Übersicht über alle möglichen Einstellungen für die Linie bietet Tabelle 4.5. Die Länge der Linie wird über die Pseudolänge `fromrulewidth` bestimmt.

In der Voreinstellung ist die Linie im erweiterten Briefkopf nicht aktiviert. Im Standardbriefkopf wird die Linie immer nach dem Namen gesetzt.

Unter dem Namen folgt die Anschrift des Absenders. Diese wird über die Variable `fromaddress` bestimmt. Im Briefkopf wird dabei die *Bezeichnung* (siehe auch Tabelle 4.6) nicht gesetzt.

Die Schrift, die für den kompletten Absender verwendet wird, kann über das Element `fromaddress` eingestellt werden. Abweichungen davon können für den Absendernamen über das Element `fromname` und für die mit `fromrule` gesetzte Linie über das Element `fromrule` eingestellt werden. In der Voreinstellung erfolgt keinerlei Schriftumschaltung. Bei der Linie

Tabelle 4.4.: Mögliche Werte für Option `fromalign` zur Platzierung des Absenders auf dem Briefbogen von scrlttr2

`center`, `centered`, `middle`	Absender wird innerhalb des Briefkopfs zentriert; ein Logo wird gegebenenfalls am Anfang der erweiterten Absenderangabe platziert; die Erweiterungen der Briefkopfgestaltung werden aktiviert.
`false`, `no`, `off`	Standardgestalt für den Absender wird verwendet; die Erweiterungen der Briefkopfgestaltung werden deaktiviert.
`left`	Absender steht linksbündig im Briefkopf; ein Logo wird gegebenenfalls rechtsbündig platziert; die Erweiterungen der Briefkopfgestaltung werden aktiviert.
`locationleft`, `leftlocation`	Absender steht linksbündig in der Absenderergänzung; ein Logo wird gegebenenfalls darüber platziert; der Briefkopf wird automatisch deaktiviert, kann aber über Option `firsthead` wieder aktiviert werden.
`locationright`, `rightlocation`, `location`	Absender steht rechtsbündig in der Absenderergänzung; ein Logo wird gegebenenfalls darüber platziert; der Briefkopf wird automatisch deaktiviert, kann aber über Option `firsthead` wieder aktiviert werden.
`right`	Absender steht rechtsbündig im Briefkopf; ein Logo wird gegebenenfalls linksbündig platziert; die Erweiterungen der Briefkopfgestaltung werden aktiviert.

Tabelle 4.5.: Mögliche Werte für Option `fromrule` zur Platzierung einer horizontalen Linie im Absender des erweiterten Briefkopfs von scrlttr2

`afteraddress`, `below`, `on`, `true`, `yes`	Linie unterhalb des kompletten Absenders
`aftername`	Linie direkt unter dem Namen des Absenders
`false`, `no`, `off`	keine Linie

ist die Möglichkeit der Schriftumschaltung hauptsächlich dazu gedacht, die Farbe der Linie ändern zu können, um etwa Grau an Stelle von Schwarz zu verwenden. Siehe hierzu [Ker07].

Beispiel: Geben wir nun dem Absender aus den bisherigen Beispielen einen Namen.

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromalign=false,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{\includegraphics{musterlogo}}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Dabei wird zunächst einmal nicht der erweiterte Briefkopf, sondern nur der Standardbriefkopf verwendet. Das Ergebnis ist in Abbildung 4.11, links zu sehen. Im Vergleich dazu ist rechts daneben das gleiche Beispiel, jedoch mit Option `fromalign=center`, also mit den aktivierten Erweiterungen für den Briefkopf, abgebildet. Wie zu sehen ist, hat diese Variante zunächst einmal keine Linie.

In Abbildung 4.11 taucht nun auch erstmals eine Signatur unter dem Gruß auf. Diese wird automatisch aus dem Absendername gewonnen. Wie sie konfiguriert werden kann, ist in Abschnitt 4.20 ab Seite 199 zu finden.

Nun soll der Brief mit aktivierter Erweiterung für den Briefkopf mit Hilfe der Option

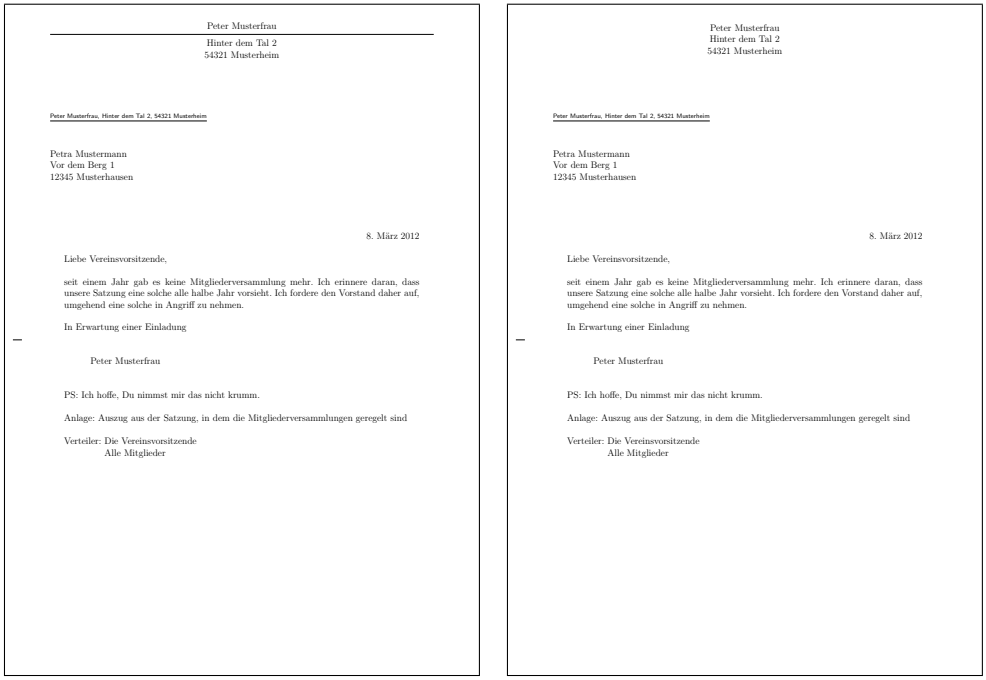


Abbildung 4.11.: Ergebnis eines kleinen Briefes mit Absender, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen und Verteiler (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe); links der Standardbriefkopf mit `fromalign=false`, rechts der erweiterte Briefkopf mit `fromalign=center`

`fromrule` auch noch eine Linie unter dem Namen erhalten:

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromalign=center,fromrule=aftername,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
```

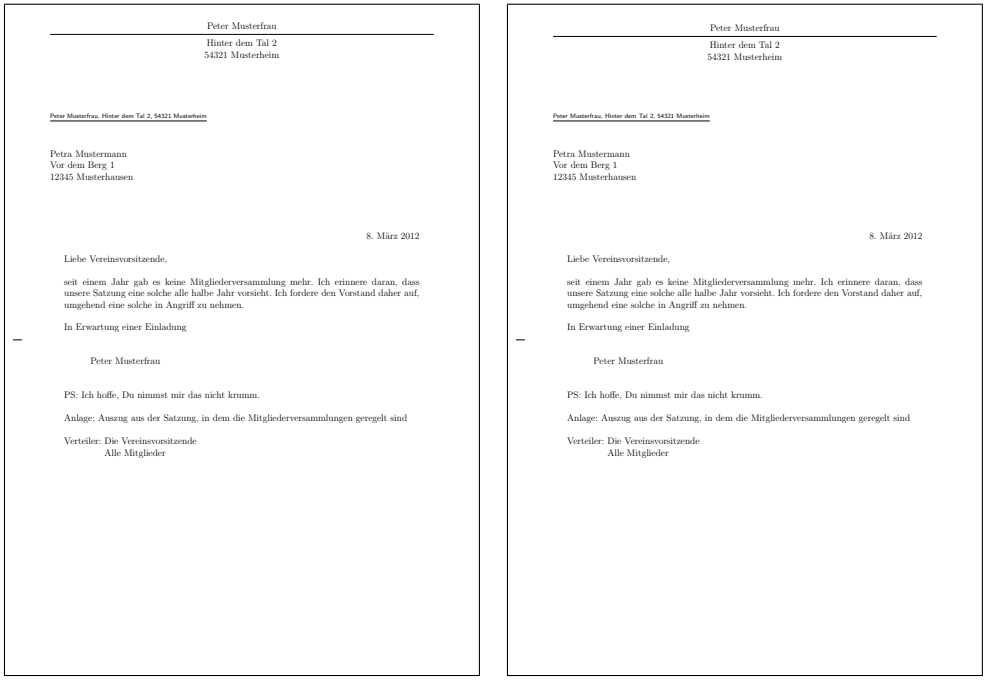


Abbildung 4.12.: Ergebnis eines kleinen Briefes mit Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links der Standardbriefkopf mit `fromalign=false`, rechts der erweiterte Briefkopf mit `fromalign=center`

```
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ist in Abbildung 4.12, rechts zu sehen. Im Vergleich dazu ist links daneben das gleiche Beispiel noch einmal mit dem Standardbriefkopf und ohne Reaktion auf die zusätzliche Option.

Ein wichtiger Hinweis betrifft noch die Absenderadresse: Innerhalb der Absenderadresse werden einzelne Teilangaben durch doppelten Backslash voneinander getrennt. Solche Teilan-

gaben sind beispielsweise Straße und Hausnummer, Postleitzahl und Ort oder eine Länderangabe. Dieser doppelte Backslash wird je nach Verwendung der Absenderadresse unterschiedlich interpretiert und ist nicht zwangsläufig als Zeilenumbruch zu verstehen. Absätze, vertikale Abstände und Ähnliches sind innerhalb der Absenderangaben normalerweise nicht gestattet. Man muss scrlttr2 schon sehr genau kennen, um solche Mittel gegebenenfalls sinnvoll im Absender einsetzen zu können. Außerdem sollte man in dem Fall unbedingt die Variablen für Rücksendeadresse (siehe Variable `backaddress`, Seite 181) und Signatur (siehe Variable `signature`, Seite 200) selbst setzen.

```
fromphone=Ein-Aus-Wert
fromfax=Ein-Aus-Wert
fromemail=Ein-Aus-Wert
fromurl=Ein-Aus-Wert
\setkomavar{fromphone}[Bezeichnung]{Inhalt}
\setkomavar{fromfax}[Bezeichnung]{Inhalt}
\setkomavar{fromemail}[Bezeichnung]{Inhalt}
\setkomavar{fromurl}[Bezeichnung]{Inhalt}
\setkomavar{phoneseparator}[Bezeichnung]{Inhalt}
\setkomavar{faxseparator}[Bezeichnung]{Inhalt}
\setkomavar{emailseparator}[Bezeichnung]{Inhalt}
\setkomavar{urlseparator}[Bezeichnung]{Inhalt}
```

Mit den Optionen `fromphone`, `fromfax`, `fromemail` und `fromurl` kann bestimmt werden, ob die Telefonnummer, die Faxnummer, die E-Mail-Adresse und die URL im Absender gesetzt werden soll. Als *Ein-Aus-Wert* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Voreingestellt ist jeweils `false`. Die Inhalte selbst werden über die gleichnamigen Variablen bestimmt. Die Voreinstellungen für die dabei verwendeten Bezeichnungen sind Tabelle 4.6 zu entnehmen, die verwendeten Trennzeichen, die zwischen der *Bezeichnung* und dem *Inhalt* einer Variablen eingefügt werden, Tabelle 4.7.

Beispiel: Herr Musterfrau aus unserem Beispiel hat auch Telefon und eine E-Mail-Adresse. Diese möchte er ebenfalls im Briefkopf haben. Gleichzeitig soll die Trennlinie nun nach dem Briefkopf stehen. Also gibt er die entsprechenden Optionen an und setzt auch die zugehörigen Variablen:

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromalign=false,fromrule=afteraddress,
  fromphone,fromemail,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
```

Tabelle 4.6.: Vordefinierte Bezeichnungen der Variablen für die Absenderangaben im Briefkopf (die Bezeichnungen und Inhalte der verwendeten Variablen für Trennzeichen ist Tabelle 4.7 zu entnehmen)

`fromemail` `\usekomavar*{emailseparator}\usekomavar{emailseparator}`
`fromfax` `\usekomavar*{faxseparator}\usekomavar{faxseparator}`
`fromname` `\headfromname`
`fromphone` `\usekomavar*{phoneseparator}\usekomavar{phoneseparator}`
`fromurl` `\usekomavar*{urlseparator}\usekomavar{urlseparator}`

```
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis aus Abbildung 4.13, links ist jedoch ernüchternd. Die Optionen werden ignoriert. Das liegt daran, dass diese zusätzlichen Variablen und Optionen nur im erweiterten Briefkopf verwendet werden. Es muss also wie in Abbildung 4.13, rechts die Option `fromalign` verwendet werden:

Tabelle 4.7.: Vordefinierte Bezeichnungen und Inhalte der Trennzeichen für die Absenderangaben im Briefkopf

Name	Bezeichnung	Inhalt
emailseparator	\emailname	:~
faxseparator	\faxname	:~
phoneseparator	\phonename	:~
urlseparator	\wwwname	:~

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromalign=center,fromrule=afteraddress,
  fromphone,fromemail,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Den Vergleich zweier Alternativen mit linksbündigem Absender durch Einstellung `fromalign=left` und rechtsbündigem Absender durch Einstellung `fromalign=right` zeigt Abbildung 4.14.

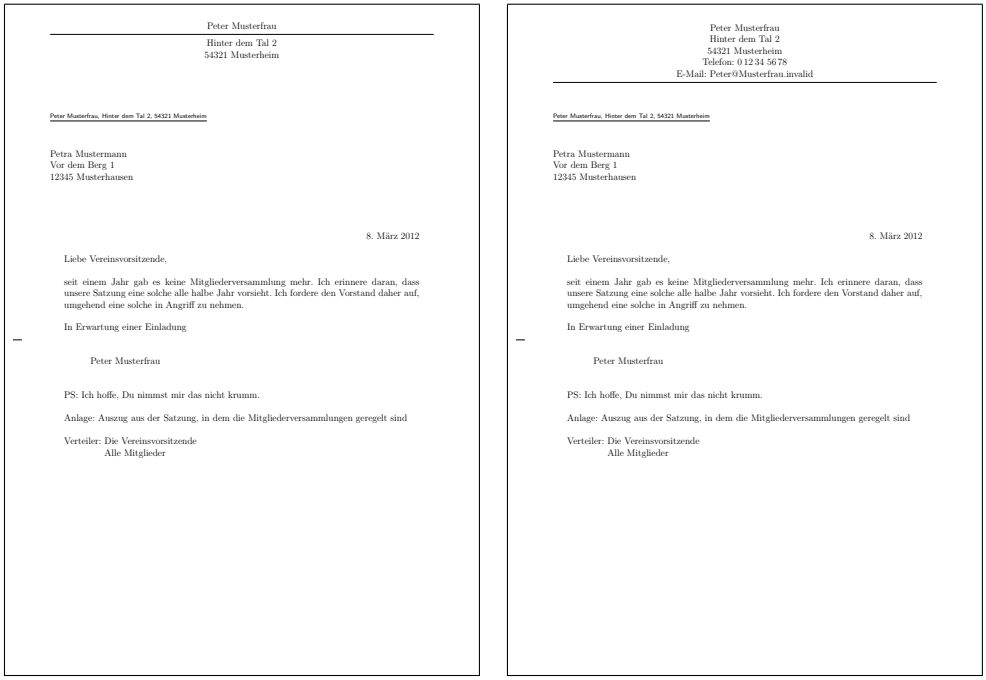


Abbildung 4.13.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links der Standardbriefkopf mit `fromalign=false`, rechts der erweiterte Briefkopf mit `fromalign=center`

`fromlogo=`*`Ein-Aus-Wert`*
`\setkomavar{fromlogo}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Mit der Option `fromlogo` kann bestimmt werden, ob ein Logo im Briefkopf gesetzt werden soll. Als *`Ein-Aus-Wert`* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Voreingestellt ist `false`, also kein Logo. Das Logo selbst wird über die Variable `fromlogo` definiert. Die *`Bezeichnung`* für das Logo ist in der Voreinstellung leer und wird von KOMA-Script auch nicht verwendet (siehe auch Tabelle 4.6).

Beispiel: Herr Musterfrau findet es besonders schick, wenn er seine Briefe mit einem Logo versieht. Sein Logo hat er als Grafikdatei gespeichert, die er gerne mit Hilfe der Anweisung `\includegraphics` laden würde. Dazu bindet er zusätzlich das Paket graphics (siehe [Car99d]) ein.

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromrule=afteraddress,
  fromphone,fromemail,fromlogo,
  version=last]{scrlttr2}
```

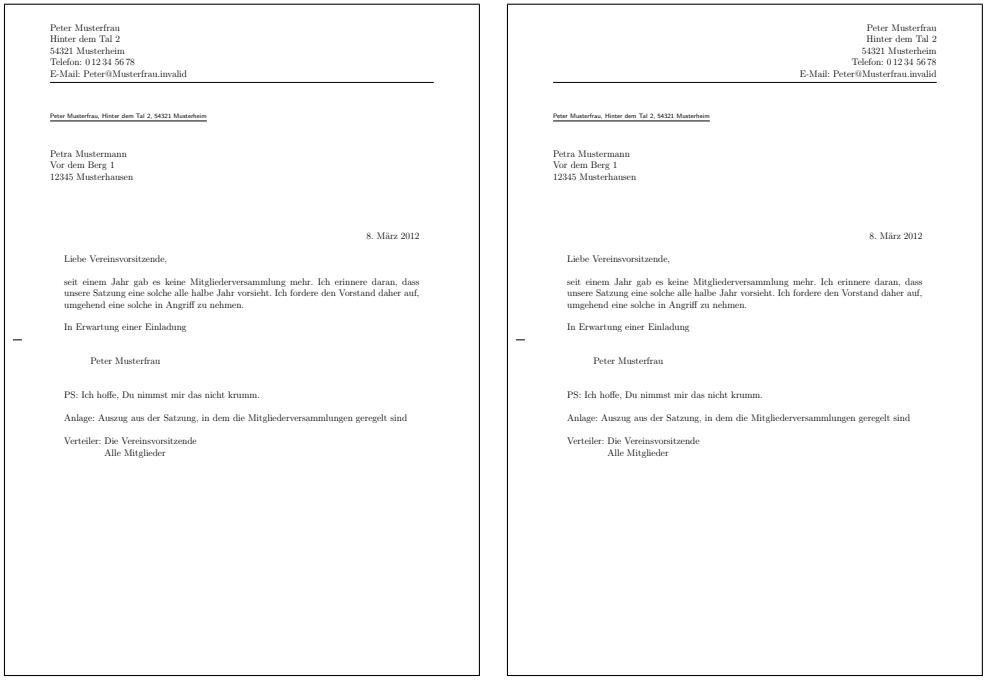


Abbildung 4.14.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links mit linksbündigem Kopf durch `fromalign=left`, rechts mit Option `fromalign=right` und damit rechtsbündigem Kopf

```
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{\includegraphics{musterlogo}}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
```

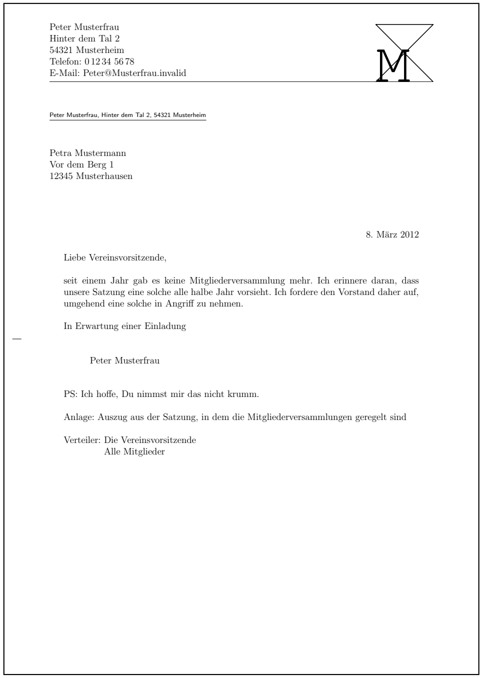
```
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ist in Abbildung 4.15 links oben zu sehen. Die beiden anderen Bilder in dieser Abbildung zeigen das Ergebnis bei rechtsbündigem und bei zentriertem Absender.

`\setkomavar{firsthead}[`*Bezeichnung*`]{`*Inhalt*`}`

In vielen Fällen reichen die Möglichkeiten aus, die scrlttr2 über Optionen und obige Variablen für die Gestaltung des Briefkopfes bietet. In einigen Fällen will man jedoch den Briefkopf freier gestalten können. In diesen Fällen muss man auf die Möglichkeiten der vordefinierten Briefköpfe, die über die oben erwähnten Option ausgewählt werden können, verzichten. Stattdessen gestaltet man sich seinen Briefkopf frei. Dazu definiert man den gewünschten Aufbau über den *Inhalt* der Variablen `firsthead`. Dabei können beispielsweise mit Hilfe der `\parbox`-Anweisung (siehe [Tea05b]) mehrere Boxen neben- und untereinander gesetzt werden. Einem versierten Anwender sollte es so möglich sein, seinen eigenen Briefkopf zu gestalten. Natürlich kann und sollte man dabei auch Zugriffe auf andere Variablen mit Hilfe von `\usekomavar` nehmen. Die *Bezeichnung* der Variablen `firsthead` wird von KOMA-Script nicht verwendet.

Lediglich aus Gründen der Kompatibilität zu frühere Version von scrlttr2 existiert auch noch Anweisung `\firsthead`. Diese sollte jedoch nicht mehr verwendet werden.



Peter Musterfrau
Hinter dem Tal 2
54321 Musterheim
Telefon: 0 12 34 56 78
E-Mail: Peter@Musterfrau.invalid

Peter Musterfrau, Hinter dem Tal 2, 54321 Musterheim

Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

8. März 2012

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

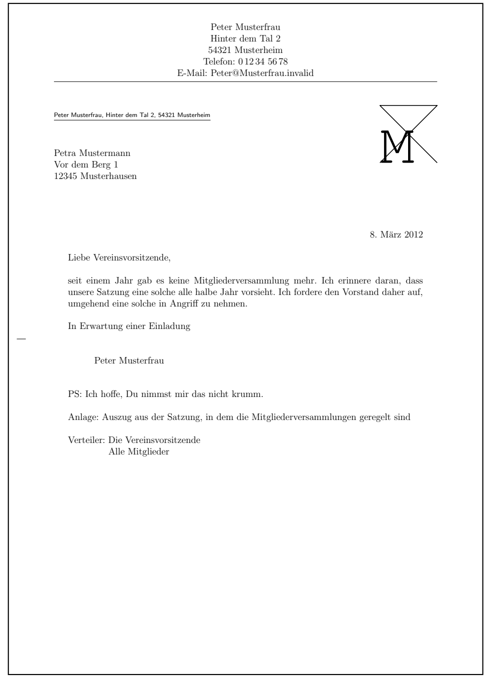
In Erwartung einer Einladung

Peter Musterfrau

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder



Peter Musterfrau
Hinter dem Tal 2
54321 Musterheim
Telefon: 0 12 34 56 78
E-Mail: Peter@Musterfrau.invalid

Peter Musterfrau, Hinter dem Tal 2, 54321 Musterheim

Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

8. März 2012

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

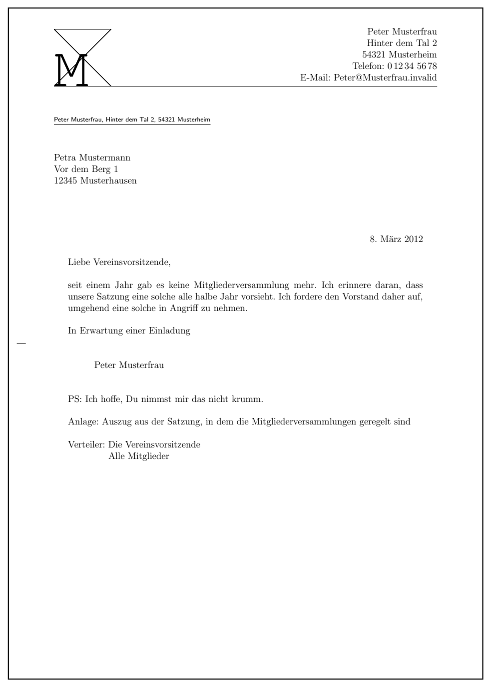
In Erwartung einer Einladung

Peter Musterfrau

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder



Peter Musterfrau
Hinter dem Tal 2
54321 Musterheim
Telefon: 0 12 34 56 78
E-Mail: Peter@Musterfrau.invalid

Peter Musterfrau, Hinter dem Tal 2, 54321 Musterheim

Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

8. März 2012

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

In Erwartung einer Einladung

Peter Musterfrau

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder

Abbildung 4.15.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links, oben mit linksbündigem Absender, rechts daneben mit zentriertem Absender und rechts mit rechtsbündigem Absender

```
addrfield=Modus
backaddress=Wert
priority=Priorität
\setkomavar{toname}[Bezeichnung]{Inhalt}
\setkomavar{toaddress}[Bezeichnung]{Inhalt}
\setkomavar{backaddress}[Bezeichnung]{Inhalt}
\setkomavar{backaddressseparator}[Bezeichnung]{Inhalt}
\setkomavar{specialmail}[Bezeichnung]{Inhalt}
\setkomavar{fromzipcode}[Bezeichnung]{Inhalt}
\setkomavar{zipcodeseparator}[Bezeichnung]{Inhalt}
\setkomavar{place}[Bezeichnung]{Inhalt}
\setkomavar{PPcode}[Bezeichnung]{Inhalt}
\setkomavar{PPdatamatrix}[Bezeichnung]{Inhalt}
\setkomavar{addresseeimage}[Bezeichnung]{Inhalt}
```

Mit der Option `addrfield` kann gewählt werden, ob ein Anschriftfeld gesetzt werden soll oder nicht. Voreingestellt ist die Verwendung eines Anschriftfeldes. Die Option versteht die in Tabelle 4.8 angegebenen Werte für den *Modus*. Voreingestellt ist `true`. Bei den Werten `true`, `PP` und `backgroundimage` werden Name und Adresse des Empfängers, die im Anschriftfeld gesetzt werden, über das Argument der Umgebung `letter` (siehe Abschnitt 4.7, Seite 152) bestimmt. Diese Angaben werden außerdem in die Variablen `toname` und `toaddress` kopiert. Die voreingestellten Schriftarten können über die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, ab Seite 56) verändert werden. Dabei existieren drei Elemente. Zunächst gibt es das Element `addressee`, das generell für die Anschrift zuständig ist. Dazu gibt es die Elemente `toname` und `toaddress`, die sich nur auf den Namen bzw. die Adresse des Empfängers beziehen. Für `toname` und `toaddress` können also Abweichungen von der Einstellung für `addressee` definiert werden.

v3.03

v2.97c

Im Anschriftfeld wird in der Voreinstellung `addrfield=true` zusätzlich noch die unterstrichene Rücksendeadresse gesetzt. Mit Option `backaddress` kann gewählt werden, ob und in welcher Form die Rücksendeadresse für Fensterbriefumschläge im Anschriftfeld gesetzt werden soll. Die Option versteht dazu einerseits die Standardwerte für einfache Schalter, die in Tabelle 2.5, Seite 40 angegeben sind. Dabei bleibt der Stil der Rücksendeadresse unverändert. Beim Einschalten der Rücksendeadresse kann andererseits gleichzeitig auch der Stil der Rücksendeadresse gewählt werden. So aktiviert der Wert `underlined` die unterstrichene Rücksendeadresse, während `plain` den Stil ohne Unterstreichung auswählt. Voreingestellt ist `underlined`, also das Setzen der unterstrichenen Rücksendeadresse.

v2.96

Die Rücksendeadresse selbst wird über den *Inhalt* der Variable `backaddress` bestimmt. Voreingestellt ist hier der über `fromname` angegebene Name und die über `fromaddress` angegebene Adresse, wobei der Doppelbackslash in diesem Fall durch den Inhalt der Variablen `backaddressseparator` ersetzte wird. Für diese ist ein Komma gefolgt von einem nicht umbrechbaren Leerzeichen vordefiniert. Die *Bezeichnung* der Variablen `backaddress` wird von KOMA-Script nicht genutzt. Die Schriftart der Rücksendeadresse ist über das Element `backaddress` konfigurierbar. Voreingestellt ist hierbei `\sffamily` (siehe Tabelle 4.9), wobei

Tabelle 4.8.: Mögliche Werte für Option `addrfield` zur Auswahl der Art der Anschrift bei scrlttr2

`true`, `on`, `yes`	Es wird eine Anschrift mit Rücksendeadresse und Versandart oder Priorität gesetzt.
`false`, `off`, `no`	Es wird keine Anschrift gesetzt.
`PP`, `pp`, `PPexplicite`, `PPExplicite`, `ppexplicite`, `ppExplicite`	Es wird eine Anschrift mit explizit über die Variablen `fromzipcode`, `place` und `PPcode` ausgefüllten Port-Payé-Kopf (P. P.-Kopf), ggf. mit Priorität und über Variable `PPdatamatrix` gesetzte Data-Matrix aber ohne Rücksendeadresse und Versandart gesetzt.
`backgroundimage`, `PPbackgroundimage`, `PPBackgroundImage`, `PPBackGroundImage`, `ppbackgroundimage`, `ppBackgroundImage`, `ppBackGroundImage`	Es wird eine Anschrift mit einer in Variable `addresseeimage` abgelegten Hintergrundgrafik als Port-Payé-Kopf (P. P.-Kopf) aber ohne Rücksendeadresse und Versandart gesetzt.
`image`, `Image`, `PPimage`, `PPImage`, `ppimage`, `ppImage`	Eine in Variable `addresseeimage` abgelegte Abbildung wird als Anschrift mit Port-Payé gesetzt. Adressinformationen und Angaben für Rücksendeadresse, Versandart oder Priorität werden ignoriert.

vor der Anwendung der konfigurierten Schriftumschaltung noch auf `\scriptsize` umgeschaltet wird.

Zwischen Rücksendeadresse und Empfängeradresse kann bei der Standardeinstellung `addrfield=true` noch eine optionale Versandart gesetzt werden. Diese wird genau dann v3.03 gesetzt, wenn die Variable `specialmail` einen *`Inhalt`* hat und `priority=manual` gewählt wird, was der Voreinstellung entspricht. Die *`Bezeichnung`* von `specialmail` wird durch scrlttr2 nicht genutzt. Die Ausrichtung wird mit Hilfe der Pseudolängen `specialmailindent` und `specialmailrightindent` (siehe Seite 327) festgelegt. Die voreingestellte Schriftart des v2.97c Elements `specialmail`, die Sie Tabelle 4.9 entnehmen können, kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 56) verändert werden.

v3.03 Wird hingegen mit `priority=A` oder `priority=B` (siehe Tabelle 4.10) eine internationale Priorität ausgewählt, so wird diese bei `addrfield=true` als Versandart und bei `addrfield=PP` an entsprechender Stelle im Port-Payé-Kopf gesetzt. Dabei wird die Grundschrift über das Element `priority` und die davon abweichende Schrift für den Prioritätsschlüssel, »A« oder »B«, über das Element `prioritykey` bestimmt. Die voreingestellten Schriftarten der beiden Elemente, die Sie Tabelle 4.9 entnehmen können, kann mit Hilfe der Anweisungen `\setkomafont`

Tabelle 4.9.: Voreinstellungen für die Schrift der Elemente des Anschriftfensters

Element	Voreinstellung
addressee	
backaddress	\sffamily
PPdata	\sffamily
PPlogo	\sffamily\bfseries
priority	\fontsize{10pt}{10pt}\sffamily\bfseries
prioritykey	\fontsize{24.88pt}{24.88pt}\selectfont
specialmail	
toaddress	
toname	

und \addtokomafont (siehe Abschnitt 4.9, Seite 56) verändert werden.

v3.03 Bei addrfield=PP wird im Port-Payé-Kopf außerdem die Postleitzahl aus der Variablen fromzipcode und der Ort aus der Variablen place gesetzt. Dabei wird der Postleitzahl aus dem *Inhalt* der Variablen fromzipcode die *Bezeichnung* der Variablen fromzipcode gefolgt vom *Inhalt* der Variablen zipcodeseparator vorangestellt. Die Voreinstellung dieser *Bezeichnung* hängt von der verwendeten lco-Datei (siehe Abschnitt 4.21 ab Seite 201) ab. Für den *Inhalt* von zipcodeseparator ist hingegen ein kleiner Abstand, gefolgt von einem Streckenstrich, gefolgt von einem kleinen Abstand (\,--\,) voreingestellt.

v3.03 Darüber hinaus wird bei addrfield=PP im Port-Payé-Kopf auch noch ein Code gesetzt, der den Absender eindeutig identifiziert. Dieser ist in Variable PPcode abgelegt. Rechts von der Anschrift kann zusätzlich eine Data-Matrix gesetzt werden, die in Variable PPdatamatrix abgelegt ist.

v3.03 Postleitzahl, Ort und Code werden in der Voreinstellung mit einer Schrift der Größe 8 pt gesetzt. Dabei wird die Schrift des Elements PPdata verwendet. Dessen Voreinstellung ist Tabel-

Tabelle 4.10.: Mögliche Werte für Option priority zur Auswahl einer internationalen Priorität im Adressfeld von scrlttr2

false, off, no, manual	Es wird keine Priorität gesetzt.
B, b, economy, Economy, ECONOMY, B-ECONOMY, B-Economy, b-economy	Es wird die internationale Priorität B-Economy gesetzt. Bei addrfield=true erfolgt dies an Stelle der Versandart.
A, a, priority, Priority, PRIORITY, A-PRIORITY, A-Priority, a-priority	Es wird die internationale Priorität A-Priority gesetzt. Bei addrfield=true erfolgt dies an Stelle der Versandart.

Tabelle 4.11.: Mögliche Werte für Option `locfield` zur Wahl der Breite des Feldes für die Absenderergänzung bei scrlttr2

`narrow`	schmales Feld für Absenderergänzungen
`wide`	breites Feld für Absenderergänzungen

le 4.9 zu entnehmen und kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 56) verändert werden.

v3.03 Bei den beiden Einstellungen `addrfield=backgroundimage` und `addrfield=image` wird eine Abbildung in das Adressfenster gesetzt. Diese ist im *Inhalt* der Variablen `addresseeimage` abgelegt. Die *Bezeichnung* dieser Variablen wird von KOMA-Script nicht genutzt. Während bei Einstellung `addrfield=image` außer der Abbildung nichts gesetzt wird, wird bei `addrfield=backgroundimage` zusätzlich noch die Anschrift aus dem obligatorischen Argument der `letter`-Umgebung ausgegeben.

Die Anordnung des Port-Payé-Kopfes wird ebenso wie die Anordnung der Port-Payé-Anschrift über die Pseudolängen `toaddrindent` (siehe Seite 326) sowie `PPheadwidth` und `PPheadheight` (siehe Seite 327) bestimmt. Für die Anordnung der Data-Matrix ist die Pseudolänge `PPdatamatrixvskip` (siehe Seite 327) zuständig.

Es wird an dieser Stelle ausdrücklich darauf hingewiesen, dass KOMA-Script selbst keine externen Abbildungen setzen kann, sollen also über die Variablen `addresseeimage` oder `PPdatamatrix` externe Abbildungen gesetzt werden, so ist beispielsweise das Grafikpaket graphics oder graphicx zu laden und in den Variablen dessen Anweisung `\includegraphics` zu verwenden.

`locfield=`*`Einstellung`*

Neben dem Anschriftfeld setzt scrlttr2 noch ein Feld mit erweiterter Absenderangabe. Dieses Feld kann beispielsweise für Bankverbindungen und Ähnliches verwendet werden. Je nach Einstellung der oben erklärten Option `fromalign` wird es außerdem für das Logo des Absenders oder den Absender selbst mitverwendet. Die Breite dieses Feldes kann beispielsweise in einer `lco`-Datei (siehe Abschnitt 4.21) gesetzt werden. Wird dort die Breite 0 gesetzt, so kann über die Option `locfield` zwischen zwei unterschiedlichen Voreinstellungen für die Breite dieses Feldes gewählt werden. Siehe hierzu auch die Erklärungen zur Pseudolänge `locwidth` in Abschnitt 17.1.4, Seite 327. Mögliche Werte für die Option sind Tabelle 4.11 zu entnehmen. Voreingestellt ist `narrow`.

`\setkomavar{location}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Der Inhalt der Absenderergänzung, soweit er nicht durch Logo oder den Absender selbst belegt ist, wird mit der Variablen `location` festgelegt. Für den *Inhalt* dieser Variablen dürfen auch Formatierungsanweisungen wie `\raggedright` verwendet werden. Die *Bezeichnung* dieser Variablen wird von KOMA-Script nicht genutzt.

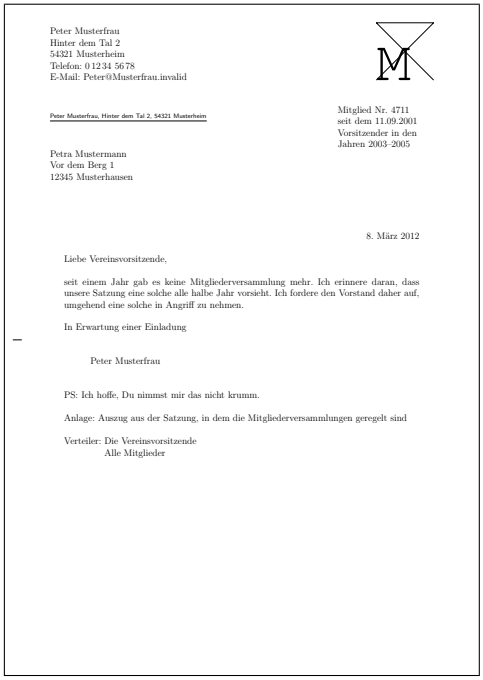
Beispiel: Herr Musterfrau möchte ein paar zusätzliche Informationen zu seiner Mitgliedschaft angeben. Er wählt dazu die Absenderergänzung:

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromphone,fromemail,fromlogo,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{\includegraphics{musterlogo}}
\setkomavar{location}{\raggedright
  Mitglied Nr.~4711\\
  seit dem 11.09.2001\\
  Vorsitzender in den Jahren 2003--2005}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das entsprechende Feld neben der Anschrift wird dann wie in Abbildung 4.16 gesetzt.

`numericaldate=`*Ein-Aus-Wert*

Mit dieser Option kann zwischen der sprachabhängigen Standarddarstellung des Datums und einem ebenfalls sprachabhängigen, kurzen, rein nummerischen Datum umgeschaltet werden.


Peter Musterfrau
Hinter dem Tal 2
54321 Musterheim
Telefon: 0 12 34 56 78
E-Mail: Peter@Musterfrau.invalid

Peter Musterfrau, Hinter dem Tal 2, 54321 Musterheim

Mitglied Nr. 4711
seit dem 11.09.2001
Vorsitzender in den
Jahren 2003–2005

Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

8. März 2012

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

In Erwartung einer Einladung

Peter Musterfrau

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder

Abbildung 4.16.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe)

Die Standarddarstellung wird nicht von KOMA-Script bereitgestellt. Sie kann wahlweise von einem Paket wie german, babel oder auch isodate stammen. Das kurze nummerische Datum wird hingegen von scrlttr2 selbst erzeugt. Die Option versteht die Standardwerte für einfache Schalter, die in Tabelle 2.5, Seite 40 angegeben sind. Voreingestellt ist mit `false` die Verwendung der Standarddarstellung.

`\setkomavar{date}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Das Datum in der gewählten Darstellung wird in im *`Inhalt`* der Variablen `date` abgelegt. Die Einstellung von `numericaldate` hat keine Auswirkung, wenn diese Variable überschrieben wird. Gesetzt wird das Datum normalerweise als Teil der Geschäftszeile. Dies ist auch dann der Fall, wenn die Geschäftszeile ansonsten leer bleibt. Für weitere Angaben zum automatischen Setzen des Datums siehe die nachfolgend beschriebene Option `refline`.

`refline=`*`Einstellung`*

Insbesondere bei Geschäftsbriefen findet sich häufig eine Zeile mit Angaben wie Namenskürzeln, Durchwahl, Kunden- und Rechnungsnummer oder zur Bezugnahme auf ein früheres Schreiben. Diese Zeile wird in dieser Anleitung *Geschäftszeile* genannt.

Bei scrlttr2 können Kopf, Fuß, Anschrift und das Feld mit der Absenderergänzung links und rechts aus dem normalen Satzspiegel herausragen. Über `refline=wide` kann gewählt werden, dass dies auch für die Geschäftszeile gelten soll. Die Geschäftszeile enthält normalerweise

Tabelle 4.12.: Mögliche Werte für Option `refline` zur Konfiguration der Geschäftszeile bei scrlttr2

`narrow`	Die Breite Geschäftszeile richtet sich nach dem Satzspiegel.
`wide`	Die Breite der Geschäftszeile richtet sich nach Anschrift und Absenderergänzung.
`dateleft` (v3.09)	Das Datum steht automatisch links in der Geschäftszeile.
`dateright` (v3.09)	Das Datum steht automatisch rechts in der Geschäftszeile.
`nodate` (v3.09)	Das Datum wird nicht automatisch in die Geschäftszeile gesetzt.

zumindest das Datum, kann aber auch weitere Angaben aufnehmen. Mögliche Werte für diese Option sind Tabelle 4.12 zu entnehmen. Voreingestellt sind `narrow` und `dateright`. (v3.09)

```
\setkomavar{yourref}[Bezeichnung]{Inhalt}
\setkomavar{yourmail}[Bezeichnung]{Inhalt}
\setkomavar{myref}[Bezeichnung]{Inhalt}
\setkomavar{customer}[Bezeichnung]{Inhalt}
\setkomavar{invoice}[Bezeichnung]{Inhalt}
```

Typische Felder der Geschäftszeile werden über die fünf Variablen `yourref`, `yourmail`, `myref`, `customer` und `invoice` verwaltet. Die Bedeutung dieser Felder sind Tabelle 4.1 auf Seite 144 zu entnehmen. Jede dieser Variablen hat außerdem eine vordefinierte *`Bezeichnung`*, die in Tabelle 4.13 zu finden ist. Wie weitere Variablen zur Geschäftszeile hinzugefügt werden können, ist in Abschnitt 17.2 ab Seite 331 erklärt.

Schriftart und Farbe der Feldbezeichnung und des Feldinhaltes können über die beiden Elemente `refname` und `refvalue` geändert werden. Dazu werden die Anweisungen `\setkomafont` (v2.97c)

Tabelle 4.13.: Vordefinierte Bezeichnungen der typischen Variablen der Geschäftszeile unter Verwendung sprachabhängiger Anweisungen

Name	Bezeichnung	bei deutscher Sprache
`yourref`	`\yourrefname`	Ihr Zeichen
`yourmail`	`\yourmailname`	Ihr Schreiben vom
`myref`	`\myrefname`	Unser Zeichen
`customer`	`\customername`	Kundennummer
`invoice`	`\invoicename`	Rechnungsnummer
`date`	`\datename`	Datum

Tabelle 4.14.: Voreinstellungen für die Schrift der Elemente der Geschäftszeile

Element	Voreinstellung
`refname`	`\sffamily\scriptsize`
`refvalue`	

und `\addtokomafont` (siehe Abschnitt 4.9, Seite 56) verwendet. Die Voreinstellungen der beiden Elemente sind Tabelle 4.14 zu entnehmen.

`\setkomavar{place}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`
`\setkomavar{placeseparator}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Sind bis auf `date` alle Variablen der Geschäftszeile leer, so wird keine eigentliche Geschäftszeile gesetzt. Stattdessen werden dann nur rechtsbündig Ort und Datum ausgegeben. Dabei bestimmt der *`Inhalt`* der Variablen `place` den Ort. Für das Trennzeichen, das in diesem Fall nach dem Ort gesetzt wird, ist der *`Inhalt`* der Variablen `placeseparator` zuständig. Der vordefinierte *`Inhalt`* des Trennzeichens ist dabei ein Komma gefolgt von einem nicht umbrechbaren Leerzeichen. Ist der Ort leer, so wird auch das Trennzeichen nicht gesetzt. Der vordefinierte *`Inhalt`* der Variablen `date` ist `\today` und hängt von der Einstellung der Option `numericaldate` ab (siehe Seite 185).

Beispiel: Herr Musterfrau setzt nun auch die Variable für den Ort:

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromphone,fromemail,fromlogo,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{\includegraphics{musterlogo}}
\setkomavar{location}{\raggedright
  Mitglied Nr.~4711\\
  seit dem 11.09.2001\\
  Vorsitzender in den Jahren 2003--2005}
\setkomavar{date}{29. Februar 2011}
\setkomavar{place}{Musterheim}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
```

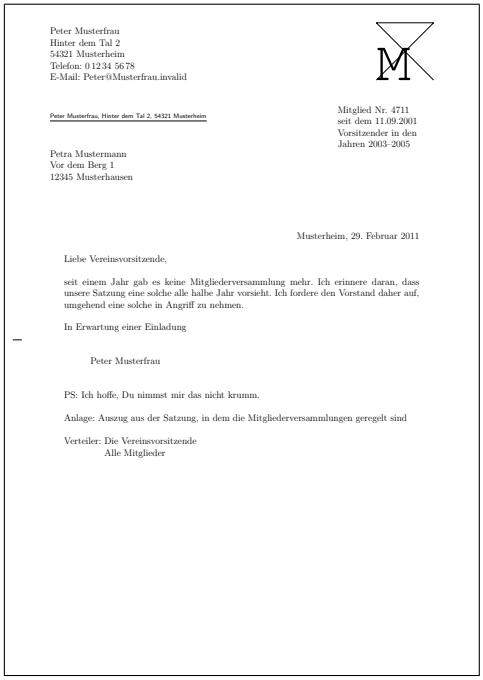


Abbildung 4.17.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke

```
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Damit erscheint vor dem Datum, wie in Abbildung 4.17 zu sehen, der Ort gefolgt von den automatischen Trennzeichen vor dem Datum. Dieses wurde nun ebenfalls explizit gesetzt, damit bei einem späteren LaTeX-Durchlauf des Briefes das ursprüngliche Datum erhalten bleibt.

Tabelle 4.15.: Vordefinierte Bezeichnungen der Variablen für den Betreff

Name	Bezeichnung
`subject`	`\usekomavar*{subjectseparator}%` `\usekomavar{subjectseparator}`
`subjectseparator`	`\subjectname`

`\setkomavar{title}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Bei scrlttr2 kann ein Brief zusätzlich mit einem Titel versehen werden. Der Titel wird zentriert in der Schriftgröße `\LARGE` unterhalb der Geschäftszeile ausgegeben. Die Schriftart für das Element `title` kann mit den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9 ab Seite 56) geändert werden. Dabei sind auch Größenangaben erlaubt. Die Größe `\Large` wird der Schriftauswahl dabei immer voran gestellt und ist daher auch nicht Teil der Voreinstellung `\normalcolor\sffamily\bfseries`.

Beispiel: Angenommen, Sie schreiben eine Mahnung. Sie setzen einen entsprechenden Titel:

```
\setkomavar{title}{Mahnung}
```

Damit sollte der Empfänger die Mahnung als solche erkennen.

Während der *`Inhalt`* der Variablen, wie im Beispiel gezeigt, den Titel definiert, wird die *`Bezeichnung`* der Variablen `title` von KOMA-Script nicht genutzt.

`subject=`*`Einstellung`*
`\setkomavar{subject}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`
`\setkomavar{subjectseparator}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Um einen Betreff zu setzten, legt man den *`Inhalt`* der Variablen `subject` entsprechend fest. Mit Option `subject=true` kann dann zum einen gewählt werden, dass der Betreff mit einem Titel versehen werden soll oder nicht. Der Titel wird über die Bezeichnung der Variablen `subject` bestimmt (siehe Tabelle 4.15). Der vordefinierte Inhalt des Trennzeichens `subjectseparator` besteht aus einem Doppelpunkt, gefolgt von einem Leerzeichen.

Zum anderen kann über `subject=afteropening` gewählt werden, dass der Betreff abweichend von der Voreinstellung erst nach der Anrede gesetzt werden soll. Eine andere Formatierung kann mit `subject=underlined` und `subject=centered` oder `subject=right` eingestellt werden. Mögliche Werte für Option `subject` sind Tabelle 4.16 zu entnehmen. Es wird ausdrücklich darauf hingewiesen, dass bei der Einstellung `subject=underlined` der Betreff komplett in eine Zeile passen muss! Voreingestellt sind `subject=left`, `subject=beforeopening` und `subject=untitled`.

v2.97c

Der Betreff wird in einer eigenen Schriftart gesetzt. Um diese zu ändern, verwenden Sie die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9 ab Seite 56). Für das Element `subject` ist in der Klasse scrlttr2 als Schrift `\normalcolor\bfseries` voreingestellt.

Tabelle 4.16.: Mögliche Werte für Option `subject` zur Platzierung und Formatierung eines Betreffs bei scrlttr2

`afteropening`	Betreff nach der Anrede setzen
`beforeopening`	Betreff vor der Anrede setzen
`centered`	Betreff zentrieren
`left`	Betreff linksbündig setzen
`right`	Betreff rechtsbündig setzen
`titled`	Betreff mit Titel versehen
`underlined`	Betreff unterstreichen (Hinweis im Text beachten!)
`untitled`	Betreff nicht mit Titel versehen

Beispiel: Herr Musterfrau setzt nun auch den Betreff. Als eher traditioneller veranlagter Mensch möchte er außerdem, dass der Betreff mit einer entsprechenden Spitzmarke versehen wird und setzt deshalb auch die entsprechende Option:

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromphone,fromemail,fromlogo,
  subject=titled,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{\includegraphics{musterlogo}}
\setkomavar{location}{\raggedright
  Mitglied Nr.~4711\\
  seit dem 11.09.2001\\
  Vorsitzender in den Jahren 2003--2005}
```

```
\setkomavar{date}{29. Februar 2011}
\setkomavar{place}{Musterheim}
\setkomavar{subject}{Mitgliederversammlung vermisst}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ist in Abbildung 4.18 zu sehen.

`firstfoot=`*`Ein-Aus-Wert`*

v2.97e

Diese Option bestimmt, ob der Briefbogenfuß überhaupt gesetzt wird. Das Abschalten mit `firstfoot=false` hat Auswirkungen wenn gleichzeitig Option `enlargefirstpage` (siehe Seite 168) verwendet wird, da sich dadurch die Seite logisch nach unten verlängert. Zwischen dem Ende des Satzspiegels und dem Seitenende bleibt dann nur der normale Abstand zwischen Satzspiegel und Seitenfuß.

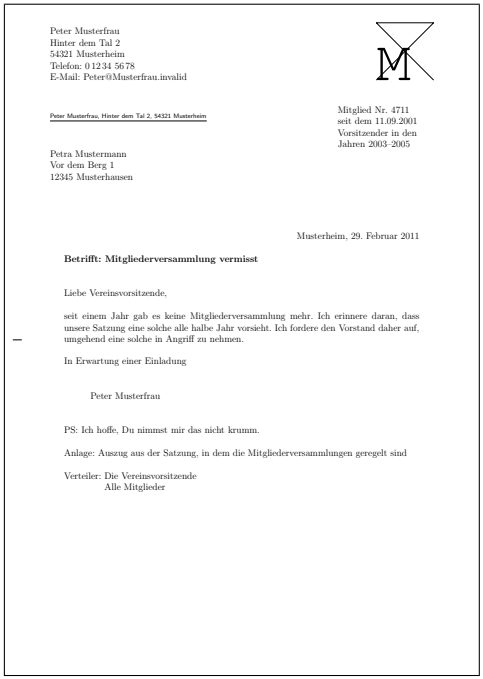
Die Option versteht die Standardwerte für einfache Schalter, die in Tabelle 2.5, Seite 40 angegeben sind. Voreingestellt ist das Setzen des Briefbogenfußes.

`\setkomavar{firstfoot}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

v3.08

Der Fuß der ersten Seite ist in der Voreinstellung leer. Es besteht jedoch die Möglichkeit, über den *`Inhalt`* der Variablen `firstfoot` einen neue Festlegung zu treffen. Die *`Bezeichnung`* der Variablen wird von KOMA-Script nicht genutzt.

Beispiel: Sie wollen den Inhalt der Variablen `frombank`, also die Bankverbindung im Fuß der ersten Seite ausgeben. Der doppelte Backslash soll dabei durch ein Komma ersetzt werden:


Peter Musterfrau
Hinter dem Tal 2
54321 Musterheim
Telefon: 0 1234 5678
E-Mail: Peter@Musterfrau.invalid

Peter Musterfrau, Hinter dem Tal 2, 54321 Musterheim

Mitglied Nr. 4711
seit dem 11.09.2001
Vorsitzender in den
Jahren 2003–2005

Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

Musterheim, 29. Februar 2011

Betrifft: Mitgliederversammlung vermisst

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

In Erwartung einer Einladung

Peter Musterfrau

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder

Abbildung 4.18.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Betreff, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke

```
\setkomavar{firstfoot}{%
  \parbox[b]{\linewidth}{%
    \centering\def\\{, }\usekomavar{frombank}%
  }%
}
```

Natürlich können Sie für das Trennzeichen auch eine eigene Variable definieren. Ich überlasse dem Leser dies als Übung.

Will man eine Art Brieffuß als Gegengewicht zum Briefkopf verwenden, so kann dieser beispielsweise wie folgt definiert werden:

```
\setkomavar{firstfoot}{%
  \parbox[t]{\textwidth}{\footnotesize
    \begin{tabular}[t]{l@{}}%
      \multicolumn{1}{@{}l@{}}{Gesellschafter:}\\
      Hugo Mayer\\
      Bernd Müller
    \end{tabular}%
    \hfill
    \begin{tabular}[t]{l@{}}%
      \multicolumn{1}{@{}l@{}}{Geschäftsführerin:}\\
      Liselotte Mayer\\[1ex]
      \multicolumn{1}{@{}l@{}}{Gerichtsstand:}\\
```

```
    Hinterdupfeldingen
  \end{tabular}%
  \ifkomavarempty{frombank}{}{%
    \hfill
    \begin{tabular}[t]{l@{}}%
      \multicolumn{1}{@{}l@{}}{%
        \usekomavar*{frombank}:}\\
      \usekomavar{frombank}
    \end{tabular}%
    }%
  }%
}
```

Das Beispiel stammt ursprünglich von Torsten Krüger, und mit

```
\setkomavar{frombank}{Konto 12\,345\,678\\
                      bei der HansWurstBank\\
                      BLZ 876\,543\,21}
```

kann die Bankverbindung passend dazu gesetzt werden.

Im Beispiel wurde ein mehrzeiliger Fuß gesetzt. Bei einer Kompatibilitätseinstellung ab Version 2.9u (siehe `version` in Abschnitt 4.4, Seite 31) reicht der Platz dafür in der Regel nicht aus. Sie sollten dann `firstfootvpos` (siehe Seite 330) entsprechend verringern.

Lediglich aus Gründen der Kompatibilität zu frühere Version von scrlttr2 existiert auch noch Anweisung `\firstfoot`. Diese sollte jedoch nicht mehr verwendet werden.

`\setkomavar{frombank}[`*Bezeichnung*`]{`*Inhalt*`}`

Die im vorherigen Beispiel verwendete Variable `frombank` nimmt derzeit eine Sonderstellung ein. Sie wird intern bisher nicht verwendet. Sie kann jedoch vom Anwender verwendet werden, um die Bankverbindung in das Absenderergänzungsfeld (siehe Variable `location`, Seite 184) oder wie im Beispiel in den Fuß zu setzen.

4.11. Absatzauszeichnung

In der Einleitung zu Abschnitt 3.10 ab Seite 71 wird dargelegt, warum der Absatzeinzug gegenüber dem Absatzabstand vorzuziehen ist. Die Elemente, auf die sich diese Argumente beziehen, beispielsweise Abbildungen, Tabellen, Listen, abgesetzte Formeln und auch neue Seiten, sind in Standardbriefen eher selten. Auch sind Briefe normalerweise nicht so umfänglich, dass ein nicht erkannter Absatz sich schwerwiegend auf die Lesbarkeit auswirkt. Die Argumente sind daher bei Standardbriefen eher schwach. Dies dürfte ein Grund dafür sein, dass der Absatzabstand bei Briefen eher gebräuchlich ist. Es bleiben damit für Standardbriefe im Wesentlichen zwei Vorteile des Absatzeinzugs. Zum einen hebt sich ein solcher Brief aus der

Masse hervor und zum anderen durchbricht man damit nicht nur für Briefe das einheitliche Erscheinungsbild aller Dokumente aus einer Quelle, die so genannte *Corporate Identity*. Von dieser Unterscheidung gegenüber den anderen KOMA-Script-Klassen abgesehen gilt, was in Abschnitt 3.10 geschrieben wurde.

4.12. Erkennung von rechten und linken Seiten

Es gilt sinngemäß, was in Abschnitt 3.11 geschrieben wurde.

4.13. Kopf und Fuß bei vordefinierten Seitenstilen

Eine der allgemeinen Eigenschaften eines Dokuments ist der Seitenstil. Bei LaTeX versteht man unter dem Seitenstil in erster Linie den Inhalt der Kopf- und Fußzeilen. Wie bereits in Abschnitt 4.10 erwähnt, werden Kopf und Fuß des Briefbogens als Elemente des Briefbogens betrachtet und unterliegen damit nicht den Einstellungen für den Seitenstil. Es geht also hier im Wesentlichen um den Seitenstil der weiteren Briefseiten nach dem Briefbogen. Bei einseitigen Briefen ist das der Seitenstil des Zweitbogens. Bei doppelseitigen Briefen ist auch der Seitenstil aller Rückseiten betroffen.

`headsepline=`*`Ein-Aus-Wert`*
`footsepline=`*`Ein-Aus-Wert`*

Mit diesen Optionen kann eingestellt werden, ob eine Trennlinie unter dem Kopf oder über dem Fuß von Folgeseiten gewünscht wird. Als *`Ein-Aus-Wert`* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Ein Aktivieren der Option `headsepline` schaltet die Linie unter dem Kopf ein. Ein Aktivieren der Option `footsepline` schaltet die Linie über dem Fuß ein. Die Deaktivierung der Optionen schaltet die jeweilige Linie aus.

Bei den Seitenstilen `empty` und `plain` (siehe später in diesem Abschnitt) hat die Option `headsepline` selbstverständlich keine Auswirkung. Bei diesen Seitenstilen soll ja auf einen Seitenkopf ausdrücklich verzichtet werden.

Typografisch betrachtet hat eine solche Linie immer die Auswirkung, dass der Kopf oder Fuß optisch näher an den Text heranrückt. Dies bedeutet nun nicht, dass Kopf oder Fuß räumlich weiter vom Textkörper weggerückt werden müssten. Stattdessen sollte sie dann bei der Berechnung des Satzspiegels als zum Textkörper gehörend betrachtet werden. Dies wird bei KOMA-Script dadurch erreicht, dass bei Verwendung der Klassenoption `headsepline` automatisch die Paketoption `headinclude` mit gleichem Wert an das typearea-Paket weitergereicht wird. Entsprechendes gilt bei `footsepline` für `footinclude`. Im Gegensatz zu `headsepline` wirkt sich die Option `footsepline` auch beim Seitenstil `plain` aus, da `plain` eine Seitenzahl im Fuß ausgibt. Das Paket scrpage2 (siehe Kapitel 5) bietet weitere Einflussmöglichkeiten für Linien im Kopf und Fuß und kann auch mit scrlttr2 kombiniert werden.

`pagenumber=`*Position*

Mit Hilfe dieser Option kann bestimmt werden, ob und wo eine Seitenzahl auf Folgeseiten gesetzt werden soll. Die Option wirkt sich auf die Seitenstile `headings`, `myheadings` und `plain` aus. Sie beeinflusst außerdem die Voreinstellung der Seitenstile des scrpage2-Paketes, soweit sie vor dem Laden dieses Paketes gesetzt wird (siehe Kapitel 5). Es gibt Werte, die sich nur auf die horizontale Position auswirken, Werte, die nur die vertikale Position beeinflussen, und Werte, die zugleich die vertikale und die horizontale Position festlegen. Mögliche Werte sind Tabelle 4.17 zu entnehmen. Voreingestellt ist `botcenter`.

`\pagestyle{`*Seitenstil*`}`
`\thispagestyle{`*lokaler Seitenstil*`}`

Bei Briefen mit scrlttr2 wird zwischen vier verschiedenen Seitenstilen unterschieden.

`empty` ist der Seitenstil, bei dem Kopf- und Fußzeile von Folgeseiten vollständig leer bleiben. Dieser Seitenstil wird auch automatisch für die erste Briefseite verwendet, da auf dieser Seite Kopf und Fuß über `\opening` (siehe Abschnitt 4.10, Seite 154) mit anderen Mitteln gesetzt werden.

`headings` ist der Seitenstil für automatische Kolumnentitel auf Folgeseiten. Dabei werden als automatisch gesetzte Marken der Absendername aus der Variablen `fromname` und der Betreff aus der Variablen `subject` verwendet (siehe Abschnitt 4.10, Seite 169 und Seite 190). Wo genau diese Marken und die Seitenangabe ausgegeben werden, hängt von der oben erklärten Option `pagenumber` und dem Inhalt der Variablen `nexthead` und `nextfoot` ab. Der Autor kann die Marken aber auch noch nach `\opening` manuell beeinflussen. Hierzu stehen wie üblich die Anweisungen `\markboth` und `\markright`, bei Verwendung von scrpage2 auch `\markleft` (siehe Abschnitt 5.1.2, Seite 218), zur Verfügung.

`myheadings` ist der Seitenstil für manuelle Kolumnentitel auf Folgeseiten. Im Unterschied zu `headings` müssen die Marken vom Anwender gesetzt werden. Er verwendet dazu die Anweisungen `\markboth` und `\markright`. Bei Verwendung von scrpage2 steht außerdem `\markleft` zur Verfügung.

`plain` ist der voreingestellte Seitenstil, bei dem auf Folgeseiten keinerlei Kolumnentitel verwendet, sondern nur eine Seitenangabe ausgegeben wird. Wo diese gesetzt wird, hängt von der oben erklärten Option `pagenumber` ab.

Die Form der Seitenstile wird außerdem durch die oben erklärten Optionen `headsepline` und `footsepline` beeinflusst. Der Seitenstil ab der aktuellen Seite wird mit `\pagestyle` umgeschaltet. Demgegenüber verändert `\thispagestyle` nur den Seitenstil der aktuellen Seite. Die Briefklasse verwendet `\thispagestyle{empty}` selbst innerhalb von `\opening` für die erste Briefseite.

Tabelle 4.17.: Mögliche Werte für Option `pagenumber` zur Positionierung der Paginierung bei den Seitenstilen `headings` und `plain` von scrlttr2

`bot`, `foot` Seitenzahl im Fuß ohne Änderung der horizontalen Position
`botcenter`, `botcentered`, `botmittle`, `footcenter`, `footcentered`, `footmiddle` Seitenzahl zentriert innerhalb des Fußes
`botleft`, `footleft` Seitenzahl links im Fuß
`botright`, `footright` Seitenzahl rechts im Fuß
`center`, `centered`, `middle` Seitenzahl zentriert ohne Änderung der vertikalen Position
`false`, `no`, `off` keine Seitenzahl
`head`, `top` Seitenzahl im Kopf ohne Änderung der horizontalen Position
`headcenter`, `headcentered`, `headmiddle`, `topcenter`, `topcentered`, `topmiddle` Seitenzahl zentriert innerhalb des Kopfes
`headleft`, `topleft` Seitenzahl links im Kopf
`headright`, `topright` Seitenzahl rechts im Kopf
`left` Seitenzahl links ohne Änderung der vertikalen Position
`right` Seitenzahl rechts ohne Änderung der vertikalen Position

Um die Schriftart von Kopf und Fuß der Seite oder der Seitenangabe zu ändern, verwenden Sie die Benutzerschnittstelle, die in Abschnitt 4.9 beschrieben ist. Für den Kopf und den Fuß ist dabei das gleiche Element zuständig, das Sie wahlweise mit `pageheadfoot` oder `pagehead` benennen können. Für den Fuß ist zusätzlich auch noch das Element `pagefoot` zuständig, das nach `pageheadfoot` in mit Variable `nextfoot` oder per Paket scrpage2 (siehe Kapitel 5, Seite 220) definierten Seitenstilen zur Anwendung kommt. Das Element für die Seitenzahl

v3.00

innerhalb des Kopfes oder Fußes heißt `pagenumber`. Die Voreinstellungen sind in Tabelle 3.8, Seite 77 zu finden. Beachten Sie dazu auch das Beispiel aus Abschnitt 3.12, Seite 77.

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

In den meisten Fällen werden die Möglichkeiten, die scrlttr2 über Optionen und Variablen für die Gestaltung des Seitenkopfes und -fußes auf Folgeseiten zur Verfügung stellt, vollkommen ausreichen. Dies umso mehr, als man zusätzlich mit `\markboth` und `\markright` die Möglichkeit hat, die Angaben zu ändern, die scrlttr2 in den Kopf setzt. Die Anweisungen `\markboth` und `\markright` können insbesondere mit dem Seitenstil `myheadings` verwendet werden. Bei Verwendung des Paketes scrpage2 gilt dies natürlich auch für den Seitenstil `scrheadings`. Dort steht außerdem die Anweisung `\markleft` zur Verfügung.

```
\setkomavar{nexthead}[Bezeichnung]{Inhalt}
\setkomavar{nextfoot}[Bezeichnung]{Inhalt}
```

In einigen wenigen Fällen will man jedoch den Kopf oder Fuß der Folgeseiten ähnlich dem Briefbogen freier gestalten. In diesen Fällen muss auf die vordefinierten Möglichkeiten, die per oben erklärter Option `pagenumber` auswählbar sind, verzichtet werden. Stattdessen gestaltet man sich den Kopf und Fuß der Folgeseiten im Seitenstil `headings` oder `myheadings` frei.

v3.08

Dazu setzt man den gewünschten Aufbau als *`Inhalt`* der Variablen `nexthead` beziehungsweise `nextfoot`. Innerhalb des Inhalts von `nexthead` und `nextfoot` können beispielsweise mit Hilfe der `\parbox`-Anweisung (siehe [Tea05b]) mehrere Boxen neben- und untereinander gesetzt werden. Einem versierten Anwender sollte es so möglich sein, eigene Seitenköpfe und -füße zu gestalten. Natürlich kann und sollte im Wert mit Hilfe von `\usekomavar` auch auf weitere Variablen zugegriffen werden. Die *`Bezeichnung`* wird von KOMA-Script bei beiden Variablen nicht genutzt.

Lediglich aus Gründen der Kompatibilität zu frühere Version von scrlttr2 existieren auch noch die Anweisungen `\nexthead` und `\nextfoot`. Diese sollten jedoch nicht mehr verwendet werden.

4.14. Vakatseiten

Es gilt sinngemäß, was in Abschnitt 3.13 geschrieben wurde. Aber

Vakatseiten sind bei Briefen eher unüblich. Das liegt nicht zuletzt daran, dass wahrhaft doppelseitige Briefe recht selten sind, da Briefe normalerweise nicht gebunden werden. Trotzdem unterstützt auch scrlttr2 für den Fall von doppelseitigen Briefen Einstellungen für Vakatseiten. Da die hier vorgestellten Anweisungen aber in Briefen kaum Verwendung finden, finden Sie hier keine Beispiele. Bei Bedarf orientieren Sie sich bitte an den Beispielen in Abschnitt 3.13 ab Seite 80.

4.15. Fußnoten

Es gilt sinngemäß, was in Abschnitt 3.14 geschrieben wurde.

4.16. Listen

Es gilt sinngemäß, was in Abschnitt 3.14 geschrieben wurde.

4.17. Mathematik

Die KOMA-Script-Klassen stellen keine eigene Umgebungen für mathematische Formeln, Gleichungssysteme oder ähnliche Elemente der Mathematik bereit. Stattdessen stützt sich KOMA-Script im Bereich der Mathematik voll und ganz auf den LaTeX-Kern. Da jedoch Mathematik in Form von nummerierten Gleichungen und Formeln in Briefen eher ungewöhnlich ist, unterstützt scrlttr2 dies nicht aktiv. Daher gibt es bei scrlttr2 auch nicht die Optionen `leqno` und `fleqn`, die in Abschnitt 3.19 für scrbook, scrreprt und scrartcl dokumentiert sind.

Auf eine Beschreibung der Mathematikumgebungen des LaTeX-Kerns, also `displaymath`, `equation` und `eqnarray`, wird an dieser Stelle verzichtet. Wer diese verwenden möchte, sei auf [SKPH99] verwiesen. Für mehr als nur einfachste mathematische Formeln und Gleichungen sei jedoch die Verwendung von amsmath empfohlen (siehe [Ame02]).

4.18. Gleitumgebungen für Tabellen und Abbildungen

Gleitumgebungen für Tabellen und Abbildungen sind in Briefen normalerweise fehl am Platz. Daher werden sie von scrlttr2 auch nicht unterstützt. Wenn solche dennoch benötigt werden, deutet dies häufig auf einen Missbrauch der Briefklasse hin. In solchen Fällen müssen Sie versuchen, die Gleitumgebungen mit Hilfe von Zusatzpaketen wie tocbasic (siehe Kapitel 13) nachzurüsten. Tabellen und Abbildungen ohne Gleitumgebungen und ohne Titel sind hingegen natürlich auch mit der Briefklasse scrlttr2 möglich.

4.19. Randnotizen

Es gilt sinngemäß, was in Abschnitt 3.21 geschrieben wurde. Bei Briefen sind Randnotizen allerdings eher unüblich und sollten äußerst sparsam eingesetzt werden.

4.20. Schlussgruß

Dass der Schlussgruß mit `\closing` gesetzt wird, wurde bereits in Abschnitt 4.7, Seite 154 erklärt. Unter dem Schlussgruß wird häufig noch eine Signatur, eine Art Erläuterung zur

Unterschrift, gesetzt. Die Unterschrift wiederum findet Platz zwischen dem Schlussgruß und der Signatur.

`\setkomavar{signature}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Die Variable `signature` nimmt eine Art Erläuterung zur Unterschrift auf. Ihr *`Inhalt`* ist mit `\usekomavar{fromname}` vordefiniert. Eine solche Erläuterung kann auch mehrzeilig sein. Die einzelnen Zeilen sollten dann mit doppeltem Backslash voneinander getrennt werden. Absätze innerhalb der Erläuterung sind jedoch nicht gestattet.

`\raggedsignature`

Grußfloskel und Erläuterung der Unterschrift werden innerhalb einer Box gesetzt. Die Breite dieser Box wird durch die längste Zeile innerhalb von Grußfloskel und Erläuterung bestimmt.

Wo genau diese Box platziert wird, ist durch die Pseudolängen `sigindent` und `sigbeforevskip` (siehe Abschnitt 17.1.7, Seite 330) bestimmt. Durch den Befehl `\raggedsignature` wird die Ausrichtung innerhalb der Box bestimmt. In den vordefinierten `lco`-Dateien ist die Anweisung entweder auf `\centering` (alle außer `KOMAold`) oder auf `\raggedright` (`KOMAold`) gesetzt. Um innerhalb der Box beispielsweise eine linksbündige oder rechtsbündige Ausrichtung zu erhalten, kann der Befehl in gleicher Weise umdefiniert werden wie `\raggedsection` (siehe das entsprechende Beispiel in Abschnitt 3.16, Seite 101).

Beispiel: Herr Musterfrau will sich nun wirklich wichtig machen und deshalb in der Signatur nochmals darauf hinweisen, dass er selbst auch mal Vereinsvorsitzender war. Deshalb ändert er die Variable `signature`. Außerdem will er, dass die Signatur linksbündig unter dem Schlussgruß steht und definiert daher auch noch `\raggedsignature` um:

```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromphone,fromemail,fromlogo,
  subject=titled,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{signature}{Peter Musterfrau\\
  (ehemaliger Vorsitzender)}
\renewcommand*{\raggedsignature}{\raggedright}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{\includegraphics{musterlogo}}
\setkomavar{location}{\raggedright
```

```
  Mitglied Nr.~4711\\
  seit dem 11.09.2001\\
  Vorsitzender in den Jahren 2003--2005}
\setkomavar{date}{29. Februar 2011}
\setkomavar{place}{Musterheim}
\setkomavar{subject}{Mitgliederversammlung vermisst}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ist in Abbildung 4.19 zu sehen.

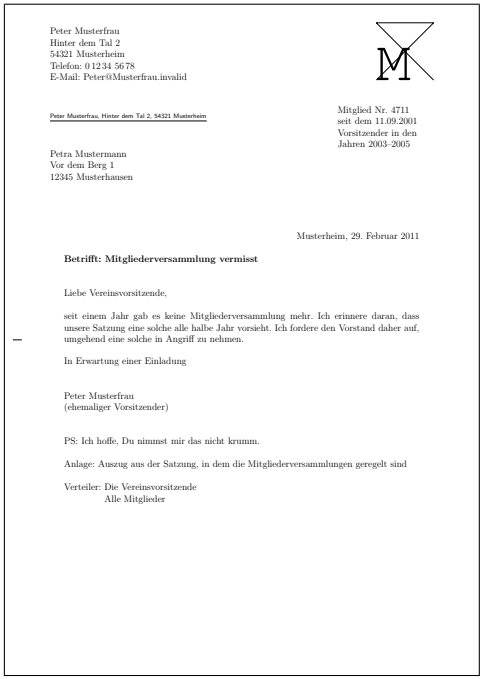
4.21. *Letter-Class-Option*-Dateien

Normalerweise wird man Einstellungen wie den Absender nicht in jedem Brief neu wählen, sondern diverse Parameter für bestimmte Gelegenheiten immer wieder verwenden. Ganz Ähnliches gilt für die verwendeten Briefköpfe und den Fußbereich der ersten Seite. Es ist deshalb sinnvoll, diese Einstellungen in einer eigenen Datei zu speichern. Die Klasse scrlttr2 bietet hierfür die `lco`-Dateien an. Die Endung `lco` steht für l*etter class o*p*tion*, also Briefklassenoption.

In `lco`-Dateien können alle Anweisungen verwendet werden, die auch an der Stelle im Dokument verwendet werden könnten, an der die `lco`-Datei geladen wird. Außerdem könnten interne Anweisungen verwendet werden, die für Paketautoren freigegeben sind. Bei scrlttr2 sind dies inbesondere die Anweisungen `\@newplength`, `\@setplength` und `\@addtoplength` (siehe Abschnitt 17.1).

v3.04

KOMA-Script liegen bereits einige `lco`-Dateien bei. Die Dateien `DIN.lco`, `DINmtext.lco`, `SNleft.lco`, `SN.lco`, `UScommercial9`, `UScommercial9DW` und `NF.lco` dienen dazu, scrlttr2 an verschiedene Normen anzupassen. Sie können von angehenden Experten sehr gut als Vorlage für eigene Parametersätze verwendet werden. Die Datei `KOMAold.lco` dient hingegen dazu, die


Peter Musterfrau
Hinter dem Tal 2
54321 Musterheim
Telefon: 0 1234 5678
E-Mail: Peter@Musterfrau.invalid

Peter Musterfrau, Hinter dem Tal 2, 54321 Musterheim

Mitglied Nr. 4711
seit dem 11.09.2001
Vorsitzender in den
Jahren 2003–2005

Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

Musterheim, 29. Februar 2011

Betrifft: Mitgliederversammlung vermisst

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

In Erwartung einer Einladung

Peter Musterfrau
(ehemaliger Vorsitzender)

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder

Abbildung 4.19.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Betreff, Anrede, Text, Grußfloskel, geänderter Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke

Kompatibilität zu scrlettr, der alten Briefklasse, zu verbessern. Da hierbei auch auf Anweisungen zurückgegriffen wird, die nicht für Paketautoren freigegeben sind, sollte man sie nicht als Vorlage für eigene `lco`-Dateien verwenden. Eine Liste aller vordefinierten `lco`-Dateien ist in Tabelle 4.18, Seite 205 zu finden.

Wenn Sie einen Parametersatz für eine Briefnorm, die bisher nicht von KOMA-Script unterstützt wird, erstellt haben, so sind Sie ausdrücklich gebeten, diesen Parametersatz an die Supportadresse von KOMA-Script zu schicken. Bitte geben Sie dabei auch die Erlaubnis zur Weiterverbreitung unter den Lizenzbedingungen von KOMA-Script (siehe dazu die Datei `LEGALDE.TXT` im KOMA-Script-Paket). Wenn Sie zwar über die notwendigen Maße aus einer bisher nicht unterstützen Briefnorm verfügen, sich jedoch nicht in der Lage sehen, selbst eine passende `lco`-Datei zu erstellen, so können Sie sich ebenfalls mit dem KOMA-Script-Autor, Markus Kohm, in Verbindung setzen. Beispiele für teilweise sehr komplexe `lco`-Dateien finden sich unter anderem unter [KDP] und in [Koh03].

`\LoadLetterOption{`*Name*`}`

Normalerweise werden `lco`-Dateien direkt über `\documentclass` geladen. Dazu gibt man den Namen der `lco`-Datei ohne die Endung als Option an. Das Laden der `lco`-Datei erfolgt dann direkt nach der Klasse.

Es ist jedoch auch möglich, eine `lco`-Datei zu einem späteren Zeitpunkt und sogar innerhalb einer anderen `lco`-Datei zu laden. Dazu dient die Anweisung `\LoadLetterOption`. Der *Name*

der lco-Datei wird dieser ebenfalls ohne Endung als Parameter übergeben.

Beispiel: Peter Musterfrau erstellt auch ein Dokument, in dem mehrere Briefe enthalten sind. Die Mehrzahl der Briefe soll nach DIN erstellt werden. Also beginnt er:

```
\documentclass{scrlttr2}
```

Allerdings soll bei einem Brief stattdessen die Variante DINmtext verwendet werden. Bei dieser steht das Adressfeld weiter oben, damit mehr Text auf die erste Seite passt. Dafür ist die Faltung so angepasst, dass das Adressfeld bei DIN C6/5-Umschlägen trotzdem in das Adressfenster passt. Sie erreichen das so:

```
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen}
  \LoadLetterOption{DINmtext}
  \opening{Hallo,}
```

Da der Aufbau der ersten Seite erst mit \opening wirklich beginnt, genügt es, wenn die lco-Datei vor \opening geladen wird. Dies muss also nicht vor \begin{letter} erfolgen. Die Änderungen durch das Laden der lco-Datei sind dann auch lokal zu dem entsprechenden Brief.

v2.97 Wird eine lco-Datei über \documentclass geladen, so darf sie nicht den Namen einer Option haben.

Beispiel: Da Herr Musterfrau regelmäßig Briefe mit immer gleichen Einstellungen schreibt, findet er es ziemlich lästig, diese Angaben immer wieder in jeden neuen Brief kopieren zu müssen. Zu seiner Erleichterung schreibt er deshalb eine lco-Datei, die ihm die Arbeit erleichtert:

```
\ProvidesFile{ich.lco}[2008/06/11 lco
  (Peter Musterfrau)]
\KOMAoptions{foldmarks=true,foldmarks=blmtP,
  fromphone,fromemail,fromlogo,subject=titled}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{signature}{Peter Musterfrau\\
  (ehemaliger Vorsitzender)}
\renewcommand*{\raggedsignature}{\raggedright}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{%
  \includegraphics{musterlogo}}
\setkomavar{location}{\raggedright
```

```
    Mitglied Nr.~4711\\
    seit dem 11.09.2001\\
    Vorsitzender in den Jahren 2003--2005}
  \setkomavar{place}{Musterheim}
  \setkomavar{frombank}{Bank freundlichen Gru\ss es}
```

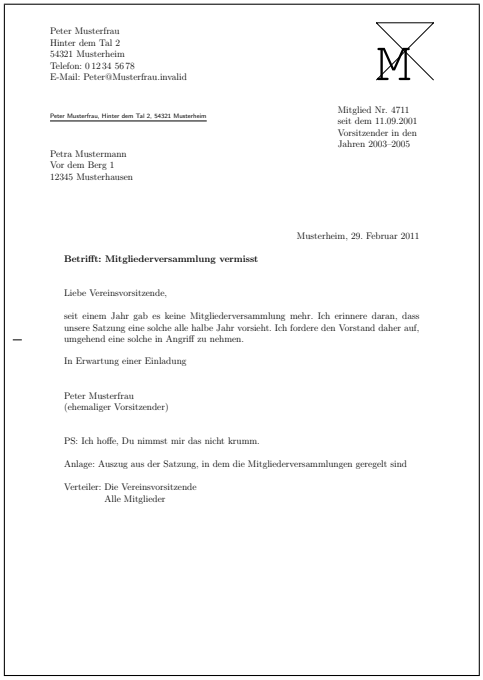
Damit schrumpft sein Brief aus dem letzten Beispiel erheblich zusammen:

```
\documentclass[version=last,ich]{scrlttr2}
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\setkomavar{date}{29. Februar 2011}
\setkomavar{subject}{Mitgliederversammlung vermisst}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche alle halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\setkomavar*{enclseparator}{Anlage}
\encl{Auszug aus der Satzung, in dem die
  Mitgliederversammlungen geregelt sind}
\cc{Die Vereinsvorsitzende\\Alle Mitglieder}
\end{letter}
\end{document}
```

Das Ergebnis ändert sich jedoch nicht, wie Abbildung 4.20 zeigt.

Bitte beachten Sie, dass in der Bankverbindung in `ich.lco` für »ß« die TEX-Schreibweise »`\ss`« verwendet wurde. Dies hat seinen Grund darin, dass direkt nach `\documentclass` weder ein Paket für die Eingabecodierung, beispielsweise mit `\usepackage[ansinew]{inputenc}` für typische Windows-Editoren oder `\usepackage[utf8]{inputenc}` für moderne Editoren, noch ein Paket zur Sprachumschaltung, beispielsweise für die neue, deutsche Rechtschreibung mit `\usepackage[ngerman]{babel}`, geladen ist.

In Tabelle 4.18 finden Sie eine Liste aller vordefinierten `lco`-Dateien. Falls Sie einen Drucker verwenden, der einen sehr großen unbedruckbaren Rand links oder rechts besitzt, werden Sie mit der Option `SN` möglicherweise Probleme bekommen. Da die Schweizer Norm SN 101 130


Peter Musterfrau
Hinter dem Tal 2
54321 Musterheim
Telefon: 0 12 34 56 78
E-Mail: Peter@Musterfrau.invalid

Peter Musterfrau, Hinter dem Tal 2, 54321 Musterheim

Mitglied Nr. 4711
seit dem 11.09.2001
Vorsitzender in den
Jahren 2003–2005

Petra Mustermann
Vor dem Berg 1
12345 Musterhausen

Musterheim, 29. Februar 2011

Betrifft: Mitgliederversammlung vermisst

Liebe Vereinsvorsitzende,

seit einem Jahr gab es keine Mitgliederversammlung mehr. Ich erinnere daran, dass unsere Satzung eine solche alle halbe Jahr vorsieht. Ich fordere den Vorstand daher auf, umgehend eine solche in Angriff zu nehmen.

In Erwartung einer Einladung

Peter Musterfrau
(ehemaliger Vorsitzender)

PS: Ich hoffe, Du nimmst mir das nicht krumm.

Anlage: Auszug aus der Satzung, in dem die Mitgliederversammlungen geregelt sind

Verteiler: Die Vereinsvorsitzende
Alle Mitglieder

Abbildung 4.20.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Betreff, Anrede, Text, Grußfloskel, geänderter Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke bei Verwendung einer `lco`-Datei

vorsieht, dass das Adressfeld 8 mm vom rechten Papierrand gesetzt wird, werden bei Schweizer Briefen auch die Kopfzeile und die Absenderergänzung mit einem entsprechend geringen Abstand zum Papierrand gesetzt. Dies betrifft ebenfalls die Geschäftszeile bei der Einstellung `refline=wide` (siehe Abschnitt 4.10, Seite 186). Sollten Sie damit ein Problem haben, erstellen Sie sich eine eigene `lco`-Datei, die zunächst `SN` lädt und in der `toaddrhpos` (siehe Abschnitt 17.1.3, Seite 325) dann auf einen kleineren Wert gesetzt wird. Verringern Sie dann außerdem `toaddrwidth` entsprechend.

Die `lco`-Datei `DIN` wird übrigens immer als erste `lco`-Datei automatisch geladen, damit alle Pseudolängen mehr oder weniger sinnvoll vordefiniert sind. Es ist daher nicht notwendig diese voreingestellte Datei selbst zu laden.

Beachten Sie bitte noch, dass es nicht möglich ist, innerhalb einer `lco`-Datei mittels `\PassOptionsToPackage` Optionen an Pakete zu übergeben, die von der Klasse bereits geladen sind. Normalerweise betrifft dies nur die Pakete typearea, scrlfile, scrbase und keyval.

Tabelle 4.18.: Vordefinierte `lco`-Dateien

`DIN`	Parametersatz für Briefe im Format A4 nach DIN 676; geeignet für Fensterbriefumschläge in den Formaten C4, C5, C6 und C6/5 (C6 lang)

. . .

Tabelle 4.18.: Vordefinierte `lco`-Dateien (*Fortsetzung*)

`DINmtext`
: Parametersatz für Briefe im Format A4 nach DIN 676, wobei die Alternative für mehr Text auf der ersten Briefseite verwendet wird; nur geeignet für Fensterbriefumschläge in den Formaten C6 und C6/5 (C6 lang)

`KakuLL`
: Parametersatz für japanische Briefe im Format A4; geeignet für japanische Fensterbriefumschläge des Typs Kaku A4, bei denen das Fenster in etwa 90 mm breit, 45 mm hoch, 25 mm vom linken und 24 mm vom oberen Rand entfernt ist (siehe dazu auch den Anhang der englischen KOMA-Script-Anleitung)

`KOMAold`
: Parametersatz für Briefe im Format A4 mit Annäherung an das Aussehen von Briefen der obsoleten Briefklasse scrlettr; geeignet für Fensterbriefumschläge in den Formaten C4, C5, C6 und C6/5 (C6 lang); es werden einige zusätzliche Anweisungen zur Verbesserung der Kompatibilität mit der obsoleten Briefklasse scrlettr definiert; scrlttr2 verhält sich mit dieser `lco`-Datei möglicherweise nicht genau wie bei Verwendung der übrigen `lco`-Dateien

`NF`
: Parametersatz für französische Briefe nach NF Z 11-001; geeignet für Fensterbriefumschläge im Format DL (110 mm auf 220 mm) mit einem Fenster von 45 mm Breite und 100 mm Höhe ca. jeweils 20 mm entfernt vom rechten unteren Rand; diese Datei wurde ursprünglich von Jean-Marie Pacquet entwickelt, der auf [Pac] neben einer Erweiterung auch eine LyX-Einbindung bereit stellt.

`NipponEH`
: Parametersatz für japanische Briefe im Format A4; geeignet für japanische Fensterbriefumschläge der Typen Chou oder You 3 oder 4, bei denen das Fenster in etwa 90 mm breit, 55 mm hoch, 22 mm vom linken und 12 mm vom oberen Rand entfernt ist (siehe dazu auch den Anhang der englischen KOMA-Script-Anleitung)

`NipponEL`
: Parametersatz für japanische Briefe im Format A4; geeignet für japanische Fensterbriefumschläge der Typen Chou oder You 3 oder 4, bei denen das Fenster in etwa 90 mm breit, 45 mm hoch, 22 mm vom linken und 12 mm vom oberen Rand entfernt ist (siehe dazu auch den Anhang der englischen KOMA-Script-Anleitung)

...

Tabelle 4.18.: Vordefinierte `lco`-Dateien (*Fortsetzung*)

`NipponLH`
: Parametersatz für japanische Briefe im Format A4; geeignet für japanische Fensterbriefumschläge der Typen Chou oder You 3 oder 4, bei denen das Fenster in etwa 90 mm breit, 55 mm hoch, 25 mm vom linken und 12 mm vom oberen Rand entfernt ist (siehe dazu auch den Anhang der englischen KOMA-Script-Anleitung)

`NipponLL`
: Parametersatz für japanische Briefe im Format A4; geeignet für japanische Fensterbriefumschläge der Typen Chou oder You 3 oder 4, bei denen das Fenster in etwa 90 mm breit, 45 mm hoch, 25 mm vom linken und 12 mm vom oberen Rand entfernt ist (siehe dazu auch den Anhang der englischen KOMA-Script-Anleitung)

`NipponRL`
: Parametersatz für japanische Briefe im Format A4; geeignet für japanische Fensterbriefumschläge der Typen Chou oder You 3 oder 4, bei denen das Fenster in etwa 90 mm breit, 45 mm hoch, 22 mm vom rechten und 28 mm vom oberen Rand entfernt ist (siehe dazu auch den Anhang der englischen KOMA-Script-Anleitung)

`SN`
: Parametersatz für Schweizer Briefe nach SN 010 130 mit Anschrift rechts; geeignet für Schweizer Fensterbriefumschläge in den Formaten C4, C5, C6 und C6/5 (C6 lang)

`SNleft`
: Parametersatz für Schweizer Briefe mit Anschrift links; geeignet für Schweizer Fensterbriefumschläge mit dem Fenster links in den Formaten C4, C5, C6 und C6/5 (C6 lang)

`UScommercial9`
: Parametersatz für US-amerikanische Briefe im Format letter; geeignet für US-amerikanische Fensterbriefumschläge der Größe *commercial No. 9* mit einem Anschriftfenster der Breite 4 1/2Unitin und Höhe 1 1/8 in an einer Position 7/8 in von links und 1/2 in von unten ohne Rücksendeadresse im Fenster; bei Faltung zunächst an der Mittelmarke und dann an der oberen Faltmarke kann auch Papier im Format legal verwendet werden, führt dann jedoch zu einer Papiergrößen-Warnung

...

Tabelle 4.18.: Vordefinierte `lco`-Dateien (*Fortsetzung*)

`UScommercial9DW`	Parametersatz für US-amerikanische Briefe im Format letter; geeignet für US-amerikanische Fensterbriefumschläge der Größe *commercial No. 9* mit einem Anschriftfenster der Breite 3 5/8Unitin und Höhe 1 1/8 in an einer Position 3/4 in von links und 1/2 in von unten mit einem Absenderfenster der Breite 3 1/2 in und Höhe 7/8 in an einer Position 5/16 in von links und 2 1/2 in von unten, jedoch ohne Rücksendeadresse im Fenster; bei Faltung zunächst an der Mittelmarke und dann an der oberen Faltmarke kann auch Papier im Format legal verwendet werden, führt dann jedoch zu einer Papiergrößen-Warnung

4.22. Adressdateien und Serienbriefe

Als besonders lästig wird bei Briefen immer das Eintippen der Adressen und das Erstellen von Serienbriefen betrachtet. Die Klasse scrlttr2 bietet hierfür eine gewisse Unterstützung.

```
\adrentry{Name}{Vorname}{Adresse}{Tel.}{F1}{F2}{Kommentar}
         {Kürzel}
```

Mit der scrlttr2-Klasse können Adressdateien ausgewertet werden. Dies ist beispielsweise für Serienbriefe sehr nützlich. Eine Adressdatei muss die Endung `.adr` haben und besteht aus einer Reihe von `\adrentry`-Einträgen. Ein solcher Eintrag besteht aus acht Elementen und kann beispielsweise wie folgt aussehen:

```
\adrentry{Maier}
         {Herbert}
         {Wiesenweg 37\\ 09091 Blumental}
         {0\,23\,34 / 91\,12\,74}
         {Bauunternehmer}
         {}
         {kauft alles}
         {MAIER}
```

Die Elemente fünf und sechs, `F1` und `F2`, können frei bestimmt werden. Denkbar wären neben Hinweisen auf das Geschlecht oder akademische Grade auch der Geburtstag oder das Eintrittsdatum in einen Verein. Um das Überschreiben von TEX- oder LATEX-Anweisungen zu vermeiden, ist es empfehlenswert, für *Kürzel* ausschließlich Großbuchstaben zu verwenden.

Beispiel: Herr Maier gehört zu Ihren engeren Geschäftspartnern. Da Sie eine rege Korrespondenz mit ihm pflegen, ist es Ihnen auf Dauer zu mühsam, jedesmal alle Empfängerdaten aufs Neue einzugeben. scrlttr2 nimmt Ihnen diese Arbeit ab. Angenommen, Sie haben Ihre Kundenkontakte in der Datei `partner.adr` gespeichert und Sie

möchten Herrn Maier einen Brief schreiben, dann sparen Sie sich viel Tipparbeit, wenn Sie folgendes eingeben:

```
\input{partner.adr}
\begin{letter}{\MAIER}
  Der Brief ...
\end{letter}
```

Achten Sie bitte darauf, dass Ihr TEX-System auch auf die `.adr`-Dateien zugreifen kann, da sonst eine Fehlermeldung von `\input` verursacht wird. Entweder Sie legen die Brief- und Adressdateien im selben Verzeichnis an, oder Sie binden ein Adressverzeichnis fest in Ihr TEX-System ein.

`\addrentry{`*Name*`}{`*Vorname*`}{`*Adresse*`}{`*Telefon*`}{`*F1*`}{`*F2*`}{`*F3*`}{`*F4*`}{`*Kürzel*`}`

Da über die Jahre hinweg immer wieder Klagen aufkamen, dass insgesamt nur zwei freie Felder zu wenig seien, verfügt scrlttr2 nun alternativ über die Anweisung `\addrentry`. Mit dem zusätzlichen »`d`« im Namen sind hier auch zwei weitere freie Felder hinzugekommen, dafür ist jedoch der Kommentar entfallen. Ansonsten kann die Anweisung genau wie `\adrentry` verwendet werden.

In einer `adr`-Datei können die beiden Anweisungen `\adrentry` und `\addrentry` nebeneinander verwendet werden. Ich weise jedoch darauf hin, dass Zusatzpakete wie das adrconv-Paket von Axel Kielhorn eventuell nicht auf die Verwendung von `\addrentry` ausgelegt sind. Hier muss der Anwender gegebenenfalls selbst entsprechende Erweiterungen vornehmen.

Neben dem vereinfachten Zugriff auf Kundendaten können die `.adr`-Dateien auch für Serienbriefe genutzt werden. So ist es ohne die komplizierte Anbindung an Datenbanksysteme möglich, solche Massenpostsendungen zu erstellen.

Beispiel: Sie wollen einen Serienbrief an alle Mitglieder Ihres Anglervereins schicken, um zur nächsten Mitgliederversammlung einzuladen.

```
\documentclass{scrlttr2}
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}

\begin{document}
\renewcommand*{\adrentry}[8]{%
  \begin{letter}{#2 #1\\#3}
    \opening{Liebe Vereinsmitglieder,}
    unsere nächste Mitgliederversammlung findet am
    Montag, dem 12.~August 2002, statt.

    Folgende Punkte müssen besprochen werden...
    \closing{Petri Heil,}
  \end{letter}
```

```
}

\input{mitglieder.adr}
\end{document}
```

Sind in Ihrer `adr`-Datei auch `\addrentry`-Anweisungen enthalten, müssen Sie dafür eine entsprechende Definition vor dem Einladen der Adressdatei ergänzen:

```
\renewcommand*{\addrentry}[9]{%
  \adrentry{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#9}%
}
```

Bei diesem Beispiel wird kein Gebrauch von dem zusätzlichen freien Feld gemacht und deshalb `\addrentry` mit Hilfe von `\adrentry` definiert.

Natürlich kann der Briefinhalt auch von den Adressatenmerkmalen abhängig gemacht werden. Als Bedingungsfelder können die frei bestimmbaren Elemente fünf oder sechs eines `\adrentry`-Eintrages oder die frei bestimmbaren Elemente fünf bis acht eines `\addrentry`-Eintrags genutzt werden.

Beispiel: Angenommen, Sie verwenden das Element fünf, um das Geschlecht eines Vereinsmitgliedes zu hinterlegen (`m/w`) und das sechste Element weist auf einen Rückstand der Mitgliedsbeiträge hin. Wollen Sie nun alle säumigen Mitglieder anschreiben und persönlich anreden, so hilft Ihnen folgendes Beispiel weiter:

```
\renewcommand*{\adrentry}[8]{
  \ifdim #6pt>0pt\relax
    % #6 ist ein Betrag (Gleitkommazahl) größer 0.
    % Es werden also die Säumigen erfasst.
    \begin{letter}{#2 #1\\#3}
      \if #5m \opening{Lieber #2,} \fi
      \if #5w \opening{Liebe #2,} \fi

      Leider mussten wir feststellen, dass du mit
      der Zahlung deiner Mitgliedsbeiträge im
      Rückstand bist.

      Wir möchten Dich bitten, den offenen Betrag
      von #6~EUR auf das Vereinskonto einzuzahlen.
      \closing{Petri Heil,}
    \end{letter}
  \fi
}
```

Es ist also möglich, den Brieftext auf bestimmte Empfängermerkmale gezielt abzustimmen und so den Eindruck eines persönlichen Schreibens zu erwecken. Die Anwendungsbreite ist

lediglich durch die maximale Anzahl von zwei freien \adrentry-Elementen beziehungsweise vier freien \addrentry-Elementen begrenzt.

\adrchar{*Anfangsbuchstaben*}
\addrchar{*Anfangsbuchstaben*}

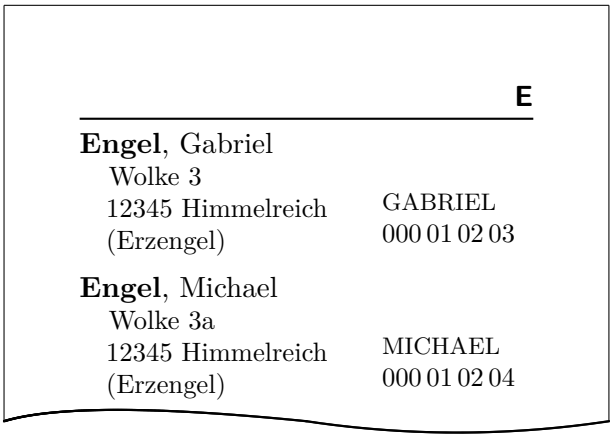
Es ist auch möglich, die Informationen einer .adr-Datei in Adressverzeichnisse oder Telefonlisten umzuwandeln. Sie benötigen dazu zusätzlich das adrconv-Paket von Axel Kielhorn (siehe [Kie99]). In diesem Paket sind interaktive LaTeX-Dokumente enthalten, mit deren Hilfe sehr einfach entsprechende Listen erstellt werden können.

Damit die Listen alphabetisch sortiert ausgegeben werden, muss bereits die Adressdatei sortiert gewesen sein. Es empfiehlt sich dabei, vor jedem neuen Anfangsbuchstaben eine Anweisung \adrchar mit diesem Buchstaben als Argument einzufügen. scrlttr2 selbst ignoriert diese Anweisung ebenso wie \addrchar.

Beispiel: Angenommen Sie haben folgende, eher winzige Adressdatei für die ein Adressbuch erstellt werden soll:

```
\adrchar{E}
\adrentry{Engel}{Gabriel}
         {Wolke 3\\12345 Himmelreich}
         {000\,01\,02\,03}{}{}{Erzengel}
         {GABRIEL}
\adrentry{Engel}{Michael}
         {Wolke 3a\\12345 Himmelreich}
         {000\,01\,02\,04}{}{}{Erzengel}
         {MICHAEL}
\adrentry{Engel}{Raphael}
         {Wolke 3b\\12345 Himmelreich}
         {000\,01\,02\,05}{}{}{Erzengel}
         {RAPHAEL}
\adrchar{K}
\adrentry{Kohm}{Markus}
         {Freiherr-von-Drais-Stra\ss e 66\\
           68535 Edingen-Neckarhausen}
         {+49~62\,03~1\,??\,??}{}{}
         {\"Uberhaupt kein Engel}
         {KOMA}
\adrchar{T}
\adrentry{Teufel}{Luzifer}
         {Hinter der Flamme 1\\
           66666 H\"ollenschlund}
         {}{}{}{Gefallener Engel ohne Telefon}
         {LUZIFER}
```

Diese bearbeiten Sie nun unter Verwendung von `adrdir.tex` aus [Kie99]. Eine Seite des Ergebnisses sieht dann etwa so aus:



E

Engel, Gabriel
Wolke 3
12345 Himmelreich
(Erzengel)

GABRIEL
000 01 02 03

Engel, Michael
Wolke 3a
12345 Himmelreich
(Erzengel)

MICHAEL
000 01 02 04

Dabei wird der Buchstabe in der Kopfzeile von `\adrchar` erzeugt. Siehe dazu die Definition in `adrdir.tex`.

Näheres zum adrconv-Paket ist der zugehörigen Anleitung zu entnehmen. Dort finden Sie auch Angaben darüber, ob die aktuelle Version von adrconv bereits mit `\addrentry` und `\addrchar` umgehen kann. Frühere Versionen kannten nur `\adrentry` und `\adrchar`.

Kopf- und Fußzeilen mit scrpage2

Wie bereits in den beiden vorherigen Kapiteln erwähnt wurde, enthält KOMA-Script auch ein Paket, das weitreichende Kontrolle über Format und Inhalt der Kopf- und Fußzeilen gibt. Bereits seit 2001 wird hierfür nicht mehr scrpage, sondern das stark erweiterte Paket scrpage2 verwendet. Diese Anleitung beschreibt daher auch nur noch scrpage2. Das Paket scrpage ist obsolet.

An Stelle von scrpage2 kann natürlich auch fancyhdr (siehe [vO00]) verwendet werden. scrpage2 harmoniert jedoch mit den KOMA-Script-Klassen deutlich besser. Genau deshalb und weil der Vorläufer von fancyhdr damals viele Möglichkeiten vermissen lies, ist scrpage2 entstanden. Natürlich ist das Paket scrpage2 nicht an eine KOMA-Script-Klasse gebunden, sondern kann auch sehr gut mit anderen Klassen verwendet werden.

5.1. Grundlegende Funktionen

Um die nachfolgende Beschreibung zu verstehen, muss noch einiges zu LaTeX gesagt werden. Im LaTeX-Kern sind die Standardseitenstile `empty`, welcher eine völlig undekorierte Seite erzeugt, und `plain`, welcher meist nur die Seitenzahl enthält, definiert. In vielen Klassen ist der Stil `headings` zu finden, welcher eine komplexe Seitendekoration mit automatischen Kolumnentitel erzeugt. Die Variante `myheadings` gleicht `headings`. Die Kolumnentitel müssen dabei aber manuell gesetzt werden. Ausführlicher wird das in Abschnitt 3.12 beschrieben. Dort wird auch erläutert, dass auf einigen Seiten der Seitenstil automatisch – in der Regel zu `plain` – wechselt.

Das Paket scrpage2 unterscheidet nicht mehr zwischen Seitenstilen mit automatischem und mit manuellem Kolumnentitel. Die Wahl des Seitenstils erfolgt unabhängig davon, ob mit automatischem oder manuellem Kolumnentitel gearbeitet wird. Näheres dazu finden Sie in Abschnitt 5.1.2.

5.1.1. Vordefinierte Seitenstile

Zu den grundlegenden Funktionen von scrpage2 gehören unter anderem vordefinierte, konfigurierbare Seitenstile.

`scrheadings`
`scrplain`

Das Paket scrpage2 liefert für Seiten mit Kolumnentitel einen eigenen Seitenstil namens `scrheadings`. Dieser Seitenstil kann mittels `\pagestyle{scrheadings}` aktiviert werden. Wird dieser Seitenstil benutzt, dann wird gleichzeitig der plain-Stil durch den dazu passenden Stil `scrplain` ersetzt. Passend bedeutet, dass auch der plain-Stil auf in Abschnitt 5.1.3 vorgestellte Befehle, die beispielsweise die Kopfbreite ändern, reagiert und im Grundlayout

übereinstimmt. Die Aktivierung des Seitenstils `scrheadings` oder des zugehörigen plain-Stils, `scrplain`, hat keine Auswirkung darauf, ob mit manuellen oder automatischen Kolumnentiteln gearbeitet wird (siehe Abschnitt 5.1.2). Der Seitenstil `scrplain` kann auch direkt per `\pagestyle` aktiviert werden.

```
\lehead[scrplain-links-gerade]{scrheadings-links-gerade}
\cehead[scrplain-mittig-gerade]{scrheadings-mittig-gerade}
\rehead[scrplain-rechts-gerade]{scrheadings-rechts-gerade}
\lefoot[scrplain-links-gerade]{scrheadings-links-gerade}
\cefoot[scrplain-mittig-gerade]{scrheadings-mittig-gerade}
\refoot[scrplain-rechts-gerade]{scrheadings-rechts-gerade}
\lohead[scrplain-links-ungerade]{scrheadings-links-ungerade}
\cohead[scrplain-mittig-ungerade]{scrheadings-mittig-ungerade}
\rohead[scrplain-rechts-ungerade]{scrheadings-rechts-ungerade}
\lofoot[scrplain-links-ungerade]{scrheadings-links-ungerade}
\cofoot[scrplain-mittig-ungerade]{scrheadings-mittig-ungerade}
\rofoot[scrplain-rechts-ungerade]{scrheadings-rechts-ungerade}
\ihead[scrplain-innen]{scrheadings-innen}
\chead[scrplain-zentriert]{scrheadings-zentriert}
\ohead[scrplain-außen]{scrheadings-außen}
\ifoot[scrplain-innen]{scrheadings-innen}
\cfoot[scrplain-zentriert]{scrheadings-zentriert}
\ofoot[scrplain-außen]{scrheadings-außen}
```

Die Seitenstile von scrpage2 sind so definiert, dass ihr Kopf und Fuß flexibel angepasst werden kann. Hierzu sind sowohl im Kopf als auch im Fuß drei Felder vorhanden sind, deren Inhalt modifiziert werden kann. Die Befehle zur Modifikation sind in Abbildung 5.1 verdeutlicht. Die in der Mitte dargestellten Befehle modifizieren sowohl die Felder der linken als auch der rechten Seite. Alle Befehle haben sowohl ein optionales als auch ein obligatorisches Argument. Das optionale Argument bestimmt jeweils das durch den Befehl festgelegte Feld im plain-Seitenstil, `scrplain`. Das obligatorische Argument definiert das entsprechende Feld im Seitenstil `scrheadings`.

Beispiel: Angenommen, man möchte bei `scrheadings` zentriert im Seitenfuß die Seitenzahl dargestellt haben, dann benutzt man einfach:

```
\cfoot{\pagemark}
```

Sollen die Seitenzahlen im Kopf außen und die Kolumnentitel innen stehen, dann erfolgt dies mit:

```
\ohead{\pagemark}
\ihead{\headmark}
\cfoot{}
```

Das `\cfoot{}` ist nur notwendig, um eine möglicherweise in der Mitte des Fußes vorhandene Seitenzahl zu entfernen.

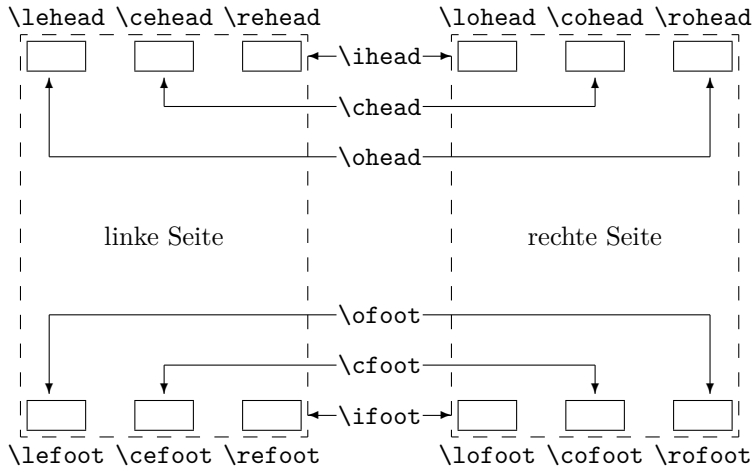


Abbildung 5.1.: Zuordnung der Befehle zur Manipulation der beiden Seitenstile `scrheadings` und `scrplain` zu den Feldern im Kopf und Fuß der Seiten

Die Befehle, die direkt nur einem Feld zugeordnet sind, können für anspruchsvollere Vorhaben genutzt werden.

Beispiel: Angenommen, man hat den Auftrag, einen Jahresbericht einer Firma zu erstellen, dann könnte das so angegangen werden:

```
\ohead{\pagemark}
\rehead{Jahresbericht 2001}
\lohead{\headmark}
\cefoot{Firma WasWeißIch}
\cofoot{Abteilung Entwicklung}
```

Natürlich muss man hier dafür sorgen, dass mittels `\cofoot` der Fuß der rechten Seite aktualisiert wird, wenn eine neue Abteilung im Bericht besprochen wird.

Wie oben dargestellt, gibt es einen zu `scrheadings` korrespondierenden plain-Seitenstil. Da es auch möglich sein soll, diesen Stil anzupassen, unterstützen die Befehle ein optionales Argument. Damit kann der Inhalt des entsprechenden Feldes im plain-Seitenstil modifiziert werden.

Beispiel: Um für die Nutzung von `scrheadings` die Position der Seitenzahlen festzulegen, kann man folgendes benutzen:

```
\cfoot[\pagemark]{}
\ohead[]{\pagemark}
```

Wird anschließend der Stil `plain` genutzt, beispielsweise weil `\chapter` eine neue Seite beginnt und darauf umschaltet, dann steht die Seitenzahl zentriert im Seitenfuß.

`\clearscrheadings`
`\clearscrplain`
`\clearscrheadfoot`

Will man sowohl den Seitenstil `scrheadings` als auch den dazu gehörenden plain-Seitenstil von Grund auf neu definieren, muss man häufig zusätzlich einige der bereits belegten Seitenelemente löschen. Da man selten alle Elemente mit neuem Inhalt füllt, sind dazu in den meisten Fällen mehrere Befehle mit leeren Parametern notwendig. Mit Hilfe dieser drei Befehle ist das Löschen schnell und gründlich möglich. Während `\clearscrheadings` lediglich alle Felder des Seitenstils `scrheadings` und `\clearscrplain` alle Felder des zugehörigen plain-Seitenstils löscht, werden von `\clearscrheadfoot` alle Felder beider Seitenstile auf leere Inhalte gesetzt.

Beispiel: Sie wollen unabhängig davon, wie die Seitenstile derzeit aussehen, die Standardform der KOMA-Script-Klassen bei zweiseitigem Satz erreichen. Dies ist mit nur drei Befehlen möglich:

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

Ohne die Befehle `\clearscrheadings`, `\clearscrplain` und `\clearscrheadfoot` wären doppelt so viele Anweisungen und neun weitere leere Argumente notwendig:

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

Einige davon könnten natürlich entfallen, wenn man von einer konkreten Vorbelegung ausginge.

In den vorausgehenden Beispielen wurden schon zwei Befehle benutzt, die noch gar nicht besprochen wurden. Das soll jetzt nachgeholt werden.

`\leftmark`
`\rightmark`

Diese beiden Befehle erlauben es, auf die Kolumnentitel zuzugreifen, die normalerweise für die linke bzw. die rechte Seite gedacht sind. Diese beiden Befehle werden nicht von scrpage2, sondern direkt vom LaTeX-Kern zur Verfügung gestellt. Wenn in diesem Kapitel vom Kolumnentitel der linken Seite oder vom Kolumnentitel der rechten Seite die Rede ist, dann ist damit eigentlich der Inhalt von `\leftmark` und `\rightmark` gemeint.

`\headmark`

Dieser Befehl ermöglicht es, auf die Inhalte der Kolumnentitel zuzugreifen. Im Gegensatz zu den originalen LaTeX-Befehlen `\leftmark` und `\rightmark` braucht man nicht auf die richtige

Zuordnung zur linken oder rechten Seite zu achten.

`\pagemark`

Dieser Befehl ermöglicht den Zugriff auf die Seitenzahl. Im Abschnitt 5.1.3, Seite 219 wird der Befehl `\pnumfont` zur Formatierung der Seitenzahl vorgestellt, den `\pagemark` automatisch berücksichtigt.

`useheadings`

Das Paket scrpage2 ist in erster Linie dafür gedacht, dass die bereitgestellten Stile benutzt oder eigene Stile definiert werden. Jedoch kann es notwendig sein, auch auf einen von der Dokumentklasse zur Verfügung gestellten Stil zurückzuschalten. Es wäre nahe liegend, dieses mit `\pagestyle{headings}` vorzunehmen. Das hätte aber den Nachteil, dass die nachfolgend besprochenen Befehle `\automark` und `\manualmark` nicht wie erwartet funktionierten. Daher sollte mit `\pagestyle{useheadings}` auf die originalen Stile umgeschaltet werden. Eine solche Umschaltung hat dann keine Auswirkung darauf, ob mit manuellen oder automatischen Kolumnentiteln gearbeitet wird.

5.1.2. Manuelle und automatische Kolumnentitel

Gewöhnlich gibt es zu einem `headings`-Stil eine *my*-Variante. Ist ein solcher Stil aktiv, dann werden die Kolumnentitel nicht mehr automatisch aktualisiert. Bei scrpage2 wird ein anderer Weg beschritten. Ob die Kolumnentitel lebend sind oder nicht, bestimmen die Befehle `\automark` und `\manualmark`. Die Voreinstellung kann auch bereits beim Laden des Paketes über die Optionen `automark` und `manualmark` beeinflusst werden (siehe Abschnitt 5.1.4, Seite 225).

`\manualmark`

Wie der Name bereits verdeutlicht, schaltet `\manualmark` die Aktualisierung der Kolumnentitel aus. Es bleibt somit dem Nutzer überlassen, für eine Aktualisierung bzw. für den Inhalt der Kolumnentitel zu sorgen. Dazu stehen die Befehle `\markboth` und `\markright` aus dem LaTeX-Kern bereit. Diese Anweisungen sind in Abschnitt 3.12, Seite 77 erklärt.

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

v2.2

Die Anweisung `\automark` aktiviert die automatische Aktualisierung des Kolumnentitels. Für die beiden Parameter sind die Bezeichnungen der Gliederungsebenen einzusetzen, deren Titel an entsprechender Stelle erscheinen soll. Gültige Werte für die Parameter sind: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph` und `subparagraph`. Der Wert `part` führt bei Verwendung der meisten Klassen nicht zu dem gewünschten Ergebnis. Bisher ist nur von den KOMA-Script-Klassen ab Version 2.9s bekannt, dass dieser Wert unterstützt wird. Das optionale Argument *rechte Seite* ist verständlicherweise nur für zweiseitigen Satz gedacht. Im

einseitigen Satz sollten Sie normalerweise darauf verzichten. Mit Hilfe der Option `autooneside` können Sie auch einstellen, dass das optionale Argument im einseitigen Satz automatisch ignoriert wird (siehe Abschnitt 5.1.4, Seite 226).

Beispiel: Wird beispielsweise mit einer *book*-Klasse gearbeitet, deren höchste Gliederungsebene *chapter* ist, dann stellt nach einem vorhergehenden `\manualmark` der Befehl

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

den Originalzustand wieder her. Bevorzugt man stattdessen, die tieferen Gliederungsebenen angezeigt zu bekommen, dann erfolgt dies mit:

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

Die Markierung der jeweils höheren Gliederungsebene wird mit Hilfe von `\markboth` (siehe Abschnitt 3.12, Seite 77) gesetzt. Die Markierung der tieferen Gliederungsebene wird mit `\markright` bzw. `\markleft` gesetzt. Der entsprechende Aufruf erfolgt indirekt über die Gliederungsbefehle. Die Anweisung `\markleft` wird von scrpage2 bereitgestellt und ist vergleichbar zu `\markright` (siehe Abschnitt 3.12, Seite 77) aus dem LaTeX-Kern definiert. Obwohl sie nicht als internes Makro definiert ist, wird von einem direkten Gebrauch abgeraten.

5.1.3. Formatierung der Kopf- und Fußzeilen

Im vorherigen Abschnitt ging es hauptsächlich um inhaltliche Dinge. Das genügt natürlich nicht, um die gestalterischen Ambitionen zu befriedigen. Deshalb soll es sich in diesem Abschnitt ausschließlich darum drehen.

`\headfont`
`\footfont`
`\pnumfont`

Die Schriftformatierung für den Seitenkopf und -fuß übernimmt der Befehl `\headfont`, `\footfont` die Abweichung davon für den Fuß und `\pnumfont` wiederum die Abweichung davon für die Seitenzahl.

Beispiel: Um beispielsweise den Kopf in fetter, serifenloser Schrift und den Fuß in nicht fetter, serifenloser Schrift zu setzen und die Seitenzahl geneigt mit Serifen erscheinen zu lassen, nutzt man folgende Definitionen:

```
\renewcommand*{\headfont}{%
  \normalfont\sffamily\bfseries}
\renewcommand*{\footfont}{%
  \normalfont\sffamily}
\renewcommand*{\pnumfont}{%
  \normalfont\rmfamily\slshape}
```

Ab Version 2.8p der KOMA-Script-Klassen wurde die Schnittstelle für Schriftattribute vereinheitlicht. Wird scrpage2 in Verbindung mit einer dieser Klassen verwendet, dann sollte die Zuweisung in der Art erfolgen, wie sie im Abschnitt 3.6 ab Seite 55 beschrieben wird.

Beispiel: Statt `\renewcommand` wird bei Verwendung einer KOMA-Script-Klasse vorzugsweise der Befehl `\setkomafont` verwendet. Die vorhergehenden Definitionen lauten damit:

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

`\setheadwidth[`*Verschiebung*`]{`*Breite*`}`
`\setfootwidth[`*Verschiebung*`]{`*Breite*`}`

Normalerweise entsprechen die Breiten von Kopf- und Fußzeile der Breite des Textbereichs. Die beiden Befehle `\setheadwidth` und `\setfootwidth` ermöglichen dem Anwender, auf einfache Weise die Breiten seinen Bedürfnissen anzupassen. Das obligatorische Argument *Breite* nimmt den Wert der Breite des Kopfes bzw. des Fußes auf, *Verschiebung* ist ein Längenmaß für die Verschiebung des entsprechenden Elements in Richtung des äußeren Seitenrandes.

Für die möglichen Standardfälle akzeptiert das obligatorische Argument *Breite* auch folgende symbolische Werte:

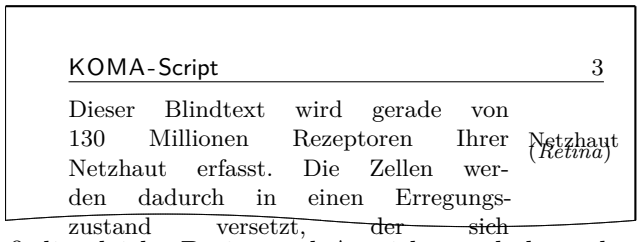
`paper`	– die Breite des Papiers
`page`	– die Breite der Seite
`text`	– die Breite des Textbereichs
`textwithmarginpar`	– die Breite des Textbereichs inklusive Seitenrand
`head`	– die aktuelle Breite des Seitenkopfes
`foot`	– die aktuelle Breite des Seitenfußes

Der Unterschied zwischen `paper` und `page` besteht darin, dass `page` die Breite des Papiers abzüglich der Bindekorrektur ist, falls das typearea-Paket verwendet wird (siehe Kapitel 2). Ohne Verwendung von typearea sind `paper` und `page` identisch.

Beispiel: Angenommen, man möchte ein Seitenlayout wie im *LaTeX-Begleiter*, bei dem die Kopfzeile in den Rand ragt, dann geschieht das ganz einfach mit:

```
\setheadwidth[0pt]{textwithmarginpar}
```

und sieht dann auf einer rechten Seite folgendermaßen aus:



Soll der Seitenfuß die gleiche Breite und Ausrichtung haben, dann hat man jetzt zwei Wege. Der erste ist, man wiederholt das Gleiche für den Seitenfuß mit:

```
\setfootwidth[Opt]{textwithmarginpar}
```

oder man greift auf den anderen symbolischen Wert `head` zurück, da der Kopf bereits die gewünschte Breite hat.

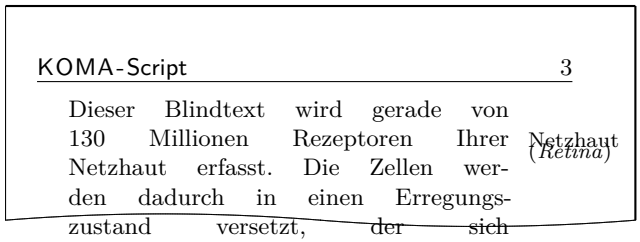
```
\setfootwidth[Opt]{head}
```

Wird keine Verschiebung angegeben, das heißt auf das optionale Argument verzichtet, dann erscheint der Kopf bzw. der Fuß symmetrisch auf der Seite angeordnet. Es wird somit ein Wert für die Verschiebung automatisch ermittelt, der der aktuellen Seitengestalt entspricht.

Beispiel: Entsprechend dem vorherigen Beispiel wird hier auf das optionale Argument verzichtet:

```
\setheadwidth{textwithmarginpar}
```

und sieht dann auf einer rechten Seite folgendermaßen aus:



Wie zu sehen, ist der Kopf jetzt nach innen verschoben, wobei die Kopfbreite sich nicht geändert hat. Die Verschiebung ist so berechnet, dass die Seitenproportionen auch hier sichtbar werden.

```
\setheadtopline[Länge]{Dicke}[Anweisungen]
\setheadsepline[Länge]{Dicke}[Anweisungen]
\setfootsepline[Länge]{Dicke}[Anweisungen]
\setfootbotline[Länge]{Dicke}[Anweisungen]
```

Entsprechend den Größenparametern für die Kopf- und Fußzeile gibt es auch Befehle, die die Dimensionen der Linien im Kopf und Fuß modifizieren können. Dazu sollten diese Lini-

en aber zunächst erst einmal eingeschaltet werden. Siehe hierzu die Optionen `headtopline`, `headsepline`, `footsepline`, `footbotline` in Abschnitt 5.1.4, Seite 224.

`\setheadtopline` – modifiziert die Parameter für die Linie über dem Seitenkopf

`\setheadsepline` – modifiziert die Parameter für die Linie zwischen Kopf und Textkörper

`\setfootsepline` – modifiziert die Parameter für die Linie zwischen Text und Fuß

`\setfootbotline` – modifiziert die Parameter für die Linie unter dem Seitenfuß

Das obligatorische Argument *Dicke* bestimmt, wie stark die Linie gezeichnet wird. Das optionale Argument *Länge* akzeptiert die gleichen symbolischen Werte wie *Breite* bei `\setheadwidth`, also auch einen normalen Längenausdruck. Solange im Dokument dem optionalen Argument *Länge* kein Wert zugewiesen wurde, passt sich die entsprechende Linienlänge automatisch der Breite des Kopfes bzw. des Fußes an.

Möchte man diesen Automatismus für die Länge einer Linie wieder restaurieren, dann nutzt man im Längenargument den Wert `auto`.

v2.2 Mit dem optionalen Argument *Anweisungen* können zusätzliche Anweisungen definiert werden, die vor dem Zeichnen der jeweiligen Linie auszuführen sind. Das können beispielsweise Anweisungen sein, um die Farbe der Linie zu ändern. Bei Verwendung einer KOMA-Script-Klasse können diese Anweisungen auch über `\setkomafont` für eines der Elemente `headtopline`, `headsepline`, `footsepline`, `footbottomline` oder auch `footbotline` gesetzt und mit `\addtokomafont` erweitert werden. Die beiden Anweisungen `\setkomafont` und `\addkomafont` sind in Abschnitt 3.6, Seite 56 näher beschrieben.

```
\setheadtopline[auto]{current}
\setheadtopline[auto]{}
\setheadtopline[auto]{}[]
```

Die hier am Befehl `\setheadtopline` illustrierten Argumente sind natürlich auch für die anderen drei Längenbefehle gültig.

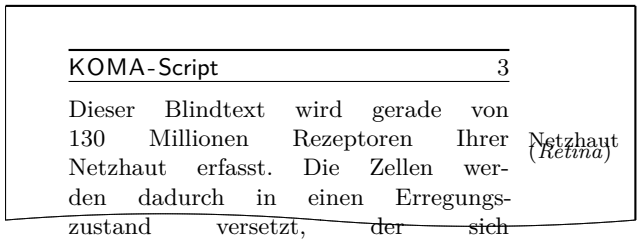
Enthält das obligatorische Argument den Wert `current` oder wird leer gelassen, dann wird die Dicke der Linie nicht verändert. Das kann genutzt werden, wenn die Länge der Linie, aber nicht die Dicke modifiziert werden soll.

Wird das optionale Argument *Anweisungen* weggelassen, so bleiben eventuell zuvor gesetzte Anweisungen erhalten. Wird hingegen ein leeres Argument *Anweisungen* gesetzt, so werden eventuell zuvor gesetzte Anweisungen wieder gelöscht.

Beispiel: Soll beispielsweise der Kopf mit einer kräftigen Linie von 2 pt darüber und einer normalen von 0,4 pt zwischen Kopf und Text abgesetzt werden, dann erfolgt das mit:

```
\setheadtopline{2pt}
\setheadsepline{.4pt}
```

Zusätzlich sind unbedingt die Optionen `headtopline` und `headsepline` vorzugsweise global im optionalen Argument von `\documentclass` zu setzen. Das Ergebnis könnte dann wie folgt aussehen.



Sollen diese Linien zusätzlich in roter Farbe gesetzt werden, dann sind die Anweisungen beispielsweise wie folgt zu ändern:

```
\setheadtopline{2pt}[\color{red}]
\setheadsepline{.4pt}[\color{red}]
```

In diesem und auch dem folgenden Beispiel wurde für die Aktivierung der Farbe die Syntax des color-Pakets verwendet, das dann natürlich auch geladen werden muss. scrpage2 selbst bietet keine direkte Farbunterstützung. Damit ist jedes beliebige Farbunterstützungspaket verwendbar.

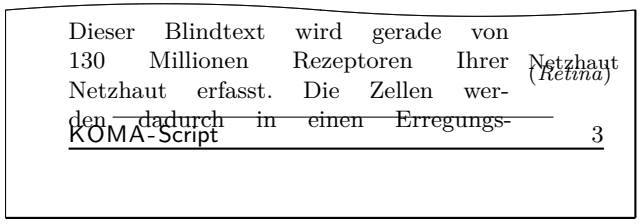
Mit einer KOMA-Script-Klasse kann alternativ

```
\setheadtopline{2pt}
\setheadsepline{.4pt}
\setkomafont{headtopline}{\color{red}}
\setkomafont{headsepline}{\color{red}}
```

verwendet werden.

Die automatische Anpassung an die Kopf- und Fußbreiten illustriert folgendes Beispiel, für das die Optionen `footbotline` und `footsepline` gesetzt sein sollten:

```
\setfootbotline{2pt}
\setfootsepline[text]{.4pt}
\setfootwidth[0pt]{textwithmarginpar}
```



Nun mag nicht jedem die Ausrichtung der Linie über der Fußzeile gefallen, sondern es wird in einem solchen Fall erwartet, dass sie wie der Kolumnentitel linksbündig zum Text ist. Diese Einstellung kann nur global in Form einer Paketoption erfolgen und wird im folgenden Abschnitt 5.1.4 mit anderen Optionen beschrieben.

5.1.4. Optionen beim Laden des Paketes

Während bei den KOMA-Script-Klassen die Mehrzahl der Optionen auch noch nach dem Laden der Klasse mit `\KOMAoptions` und `\KOMAoption` geändert werden kann, trifft dies für das Paket scrpage2 derzeit noch nicht zu. Alle Optionen für dieses Paket müssen als globale Optionen, also im optionalen Argument von `\documentclass`, oder als Paketoptionen, also im optionalen Argument von `\usepackage`, angegeben werden.

`headinclude`
`headexclude`
`footinclude`
`footexclude`

v2.3 Diese Optionen sollten bei Verwendung von KOMA-Script 3 nicht mehr beispielsweise per optionalem Argument von `\usepackage` oder per `\PassOptionsToPackage` direkt an scrpage2 übergeben werden. Lediglich aus Gründen der Kompatibilität sind sie noch in scrpage2 deklariert und werden von diesem als `headinclude`, `headinclude=false`, `footinclude` und `footinclude=false` an das Paket typearea (siehe Abschnitt 2.6, Seite 41) weitergereicht.

`headtopline`
`plainheadtopline`
`headsepline`
`plainheadsepline`
`footsepline`
`plainfootsepline`
`footbotline`
`plainfootbotline`

Eine Grundeinstellung für die Linien unter und über den Kopf- und Fußzeilen kann mit diesen Optionen vorgenommen werden. Diese Einstellungen gelten dann als Standard für alle mit scrpage2 definierten Seitenstile. Wird eine von diesen Optionen verwendet, dann wird eine Linienstärke von 0,4 pt eingesetzt. Da es zum Seitenstil `scrheadings` einen entsprechenden plain-Stil gibt, kann mit den `plain...`-Optionen auch die entsprechende Linie des plain-Stils konfiguriert werden. Diese `plain`-Optionen wirken aber nur, wenn auch die korrespondierende Option ohne `plain` aktiviert wurde. Somit zeigt die Option `plainheadtopline` ohne `headtopline` keine Wirkung.

Bei diesen Optionen ist zu beachten, dass der entsprechende Seitenteil in den Textbereich des Satzspiegels mit übernommen wird, wenn eine Linie aktiviert wurde. Wird also mittels

`headsepline` die Trennlinie zwischen Kopf und Text aktiviert, dann wird automatisch mittels typearea der Satzspiegel so berechnet, dass der Seitenkopf Teil des Textblocks ist.

Die Bedingungen für die Optionen des vorhergehenden Abschnitts gelten auch für diesen Automatismus. Das bedeutet, dass das Paket typearea nach scrpage2 geladen werden muss, beziehungsweise, dass bei Verwendung einer KOMA-Script-Klasse die Optionen `headinclude` und `footinclude` explizit bei `\documentclass` gesetzt werden müssen, um Kopf- bzw. Fußzeile in den Textblock zu übernehmen.

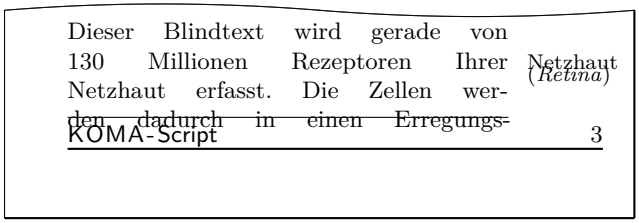
`ilines`
`clines`
`olines`

Bei der Festlegung der Linienlängen kann es vorkommen, dass die Linie zwar die gewünschte Länge, aber nicht die erwünschte Ausrichtung hat, da sie im Kopf- bzw. Fußbereich zentriert wird. Mit den hier vorgestellten Paketoptionen kann global für alle mit scrpage2 definierten Seitenstile diese Vorgabe modifiziert werden. Dabei setzt `ilines` die Ausrichtung so, dass die Linien an den inneren Rand verschoben werden. Die Option `clines` verhält sich wie die Standardeinstellung und `olines` richtet am äußeren Rand aus.

Beispiel: Hier gilt es, das Beispiel zu `\setfootsepline` auf Seite 223 mit dem folgenden zu vergleichen, um die Wirkung der Option `ilines` zu sehen.

```
\usepackage[ilines,footsepline,footbotline]
           {scrpage2}
\setfootbotline{2pt}
\setfootsepline[text]{.4pt}
\setfootwidth[0pt]{textwithmarginpar}
```

Allein die Verwendung der Option `ilines` führt dabei zu der geänderten Ausgabe, die nachfolgend veranschaulicht wird:



Die Trennlinie zwischen Text und Fuß wird bündig innen im Fußteil gesetzt und nicht wie bei der Standardeinstellung zentriert.

`automark`
`manualmark`

Diese Optionen bestimmen gleich zu Beginn des Dokuments, ob eine automatische Aktualisierung der Kolumnentitel erfolgt. Die Option `automark` schaltet die automatische Aktualisierung

ein, `manualmark` deaktiviert sie. Ohne Verwendung einer der beiden Optionen bleibt die Einstellung erhalten, die beim Laden des Paketes gültig war.

Beispiel: Sie laden das Paket scrpage2 unmittelbar nach der Klasse scrreprt und ohne weitere Optionen. Dazu schreiben Sie:

```
\documentclass{scrreprt}
\usepackage{scrpage2}
```

Da bei scrreprt der Seitenstil `plain` voreingestellt ist, ist dies auch jetzt noch der Fall. Außerdem entspricht die Voreinstellung `plain` manuellen Kolumnentiteln. Wenn Sie also anschließend mit

```
\pagestyle{scrheadings}
```

auf den Seitenstil `scrheadings` umschalten, sind noch immer manuelle Kolumnentitel eingestellt.

Verwenden Sie stattdessen die Dokumentklasse scrbook, so ist nach

```
\documentclass{scrbook}
\usepackage{scrpage2}
```

der Seitenstil `headings` mit automatischen Kolumnentiteln aktiviert. Bei anschließender Umschaltung auf den Seitenstil `scrheadings` bleiben automatische Kolumnentitel eingeschaltet. Dabei werden dann weiterhin die Markierungsmakros von scrbook verwendet.

Verwenden Sie hingegen

```
\usepackage[automark]{scrpage2}
```

so wird unabhängig von der verwendeten Klasse auf automatische Kolumnentitel umgeschaltet, wobei die Markierungsmakros von scrpage2 genutzt werden. Natürlich wirkt sich dies auf den Seitenstil `plain` von scrreprt nicht aus. Die Kolumnentitel werden erst sichtbar, wenn auf den Seitenstil `scrheadings` oder `useheadings` oder einen selbst definierten Seitenstil mit Kolumnentiteln umgeschaltet wird.

`autooneside`

Mit dieser Option wird das optionale Argument von `\automark` im einseitigen Satz automatisch ignoriert. Siehe hierzu auch die Erläuterung zum Befehl `\automark` in Abschnitt 5.1.2, Seite 218.

`komastyle`
`standardstyle`

Diese Optionen bestimmen, wie die beiden vordefinierten Seitenstile `scrheadings` und `scrplain` gestaltet sind. Bei `komastyle` wird eine Definition vorgenommen, wie sie den

KOMA-Script-Klassen entspricht. Bei den KOMA-Script-Klassen ist dies die Voreinstellung und kann somit auch für andere Klassen gesetzt werden.

Die Option `standardstyle` definiert die beiden Seitenstile wie es von den Standardklassen erwartet wird. Außerdem wird hier automatisch `markuppercase` aktiviert, es sei denn, `markusedcase` wird ebenfalls als Option übergeben.

`markuppercase`
`markusedcase`

Für die Funktionalität von `\automark` modifiziert scrpage2 interne Befehle, die die Gliederungsbefehle benutzen, um die lebenden Kolumnentitel zu setzen. Da einige Klassen, im Gegensatz zu den KOMA-Script-Klassen, die Kolumnentitel in Großbuchstaben schreiben, muss scrpage2 wissen, wie die genutzte Dokumentklasse die lebenden Kolumnentitel darstellt.

Die Option `markuppercase` zeigt scrpage2, dass die benutzte Klasse die Großschreibweise benutzt. Die Option `markusedcase` sollte angegeben werden, wenn die benutzte Dokumentklasse keine Großschreibweise verwendet. Die Optionen sind nicht geeignet, eine entsprechende Darstellung zu erzwingen. Es kann somit zu unerwünschten Effekten kommen, wenn die Angabe nicht dem Verhalten der Dokumentklasse entspricht.

`nouppercase`

Wie in obiger Erklärung zu `markuppercase` und `markusedcase` bereits ausgeführt wurde, gibt es Klassen und auch Pakete, die beim Setzen der lebenden Kolumnentitel mit Hilfe einer der Anweisungen `\uppercase` oder `\MakeUppercase` den gesamten Eintrag in Großbuchstaben wandeln. Mit der Option `nouppercase` können diese beiden Anweisungen im Kopf und im Fuß außer Kraft gesetzt werden. Das gilt aber nur für Seitenstile, die mit Hilfe von scrpage2 definiert werden. Dazu zählen auch `scrheadings` und der zugehörige plain-Seitenstil.

Die verwendete Methode ist äußerst brutal und kann dazu führen, dass auch erwünschte Änderungen von Klein- in Großbuchstaben unterbleiben. Da diese Fälle nicht sehr häufig sind, stellt `nouppercase` aber meist eine brauchbare Lösung dar.

Beispiel: Sie verwenden die Standardklasse book, wollen aber, dass die lebenden Kolumnentitel nicht in Großbuchstaben, sondern in normaler gemischter Schreibweise gesetzt werden. Die Präambel Ihres Dokuments könnte dann wie folgt beginnen:

```
\documentclass{book}
\usepackage[nouppercase]{scrpage2}
\pagestyle{scrheadings}
```

Die Umschaltung auf den Seitenstil `scrheadings` ist notwendig, weil sonst der Seitenstil `headings` verwendet wird, der von der Option `nouppercase` nicht behandelt wird.

In einigen Fällen setzen nicht nur Klassen, sondern auch Pakete lebende Kolumnentitel in Großbuchstaben. Auch in diesen Fällen hilft `nouppercase` meist, um zu gemischter Schreibweise zurückzuschalten.

5.2. Seitenstile selbst gestalten

5.2.1. Die Anwenderschnittstelle

Nun möchte man ja nicht immer an die vorgegebenen Seitenstile gebunden sein, sondern auch seiner Kreativität freien Lauf lassen. Manchmal ist man auch dazu gezwungen, weil eine bestimmte *Corporate Identity* einer Firma es verlangt. Der einfachste Weg damit umzugehen ist

```
\deftripstyle{Name}[LA][LI]{KI}{KM}{KA}{FI}{FM}{FA}
```

Die einzelnen Felder haben folgende Bedeutung:

Name – die Bezeichnung des Seitenstils, die dann bei der Aktivierung mit `\pagestyle{Name}` oder `\thispagestyle{Name}` verwendet wird

LA – die Dicke der äußeren Linien, d. h. der Linien über der Kopfzeile und unter der Fußzeile (optional)

LI – die Dicke der inneren Linie, d. h. der Linien die Kopf und Fuß vom Textkörper trennen (optional)

KI – Inhalt des Feldes im Kopf innenseitig oder bei einseitigem Layout links

KM – Inhalt des Feldes im Kopf zentriert

KA – Inhalt des Feldes im Kopf außenseitig oder bei einseitigem Layout rechts

FI – Inhalt des Feldes im Fuß innenseitig oder bei einseitigem Layout links

FM – Inhalt des Feldes im Fuß zentriert

FA – Inhalt des Feldes im Fuß außenseitig oder bei einseitigem Layout rechts

Der Befehl `\deftripstyle` stellt sicherlich die einfachste Möglichkeit dar, Seitenstile zu definieren. Leider sind damit auch Einschränkungen verbunden, da in einem Seitenbereich mit einem durch `\deftripstyle` deklarierten Seitenstil keine Änderung der Kopf- und Fußlinien erfolgen kann.

Beispiel: Vorgegeben sei ein doppelseitiges Layout, bei dem die Kolumnentitel innen erscheinen sollen. Weiterhin soll der Dokumenttitel, in diesem Fall kurz „Bericht“, an den Außenrand in den Kopf, die Seitenzahl soll zentriert in den Fuß.

```
\deftripstyle{DerBericht}%
                {\headmark}{}{Bericht}%
                {}{\pagemark}{}
```

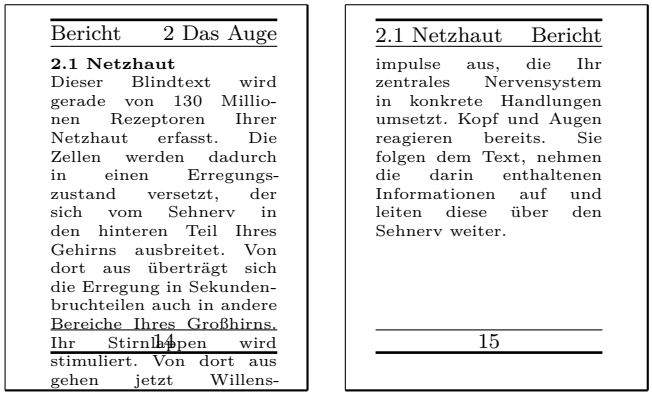


Abbildung 5.2.: Beispiel für einen selbst definierten, von Linien dominierten Seitenstil mit einem statischen und einem lebenden Kolumnentitel im Kopf und der Seitenzahl in der Mitte des Fußes.

Sollen weiterhin die Linien über dem Kopf und unter dem Fuß mit 2 pt erscheinen und der ganze Textkörper mit dünnen Linien von 0,4 pt von Kopf und Fuß abgesetzt werden, dann erweitert man vorherige Definition.

```
\deftripstyle{DerBericht}[2pt][.4pt]%
                {\headmark}{}{Bericht}%
                {}{\pagemark}{}
```

Das Ergebnis ist in Abbildung 5.2 zu sehen.

5.2.2. Die Expertenschnittstelle

Einfache Seitenstile, wie sie mit `\deftripstyle` deklariert werden können, sind erfahrungsgemäß selten. Entweder verlangt ein Professor, dass die Diplomarbeit so aussieht wie seine eigene – und wer will ihm da *ernsthaft* widersprechen – oder eine Firma möchte, dass die halbe Finanzbuchhaltung im Seitenfuß auftaucht. Alles kein Problem, denn es gibt noch:

```
\defpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\newpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\renewpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\providepagestyle{Name}{Kopfdefinition}{Fußdefinition}
```

Dies sind die Befehle, die die volle Kontrolle über die Gestaltung eines Seitenstils ermöglichen. Der Aufbau ist bei allen vier Definitionen gleich, sie unterscheiden sich nur hinsichtlich der Wirkungsweise.

`\defpagestyle` – definiert einen neuen Seitenstil. Existiert bereits einer mit diesem Namen, wird dieser überschrieben.

`\newpagestyle` – definiert einen neuen Seitenstil. Wenn schon einer mit diesem Namen existiert, wird ein Fehler ausgegeben.

`\renewpagestyle` – definiert einen bestehenden Seitenstil um. Wenn noch keiner mit diesem Namen existiert, wird ein Fehler ausgegeben.

`\providepagestyle` – definiert einen neuen Seitenstil nur dann, wenn dieser vorher noch nicht existiert.

Am Beispiel von `\defpagestyle` soll die Syntax der Definitionen im Folgenden erläutert werden.

Name – die Bezeichnung des Seitenstils

Kopfdefinition – die Deklaration des Seitenkopfes bestehend aus fünf Teilen, wobei die in runden Klammern stehenden Angaben optional sind:
(OLL,OLD){GS}{US}{ES}(ULL,ULD)

Fußdefinition – die Deklaration des Seitenfußes bestehend aus fünf Teilen, wobei die in runden Klammern stehenden Angaben optional sind:
(OLL,OLD){GS}{US}{ES}(ULL,ULD)

Wie zu sehen ist, haben Kopf- und Fußdefinition identischen Aufbau. Die einzelnen Parameter haben folgende Bedeutung:

OLL – obere Linienlänge: Kopf = außen, Fuß = Trennlinie

OLD – obere Liniendicke

GS – Definition für die *gerade* Seite

US – Definition für die *ungerade* Seite

ES – Definition für *einseitiges* Layout

ULL – untere Linienlänge Kopf = Trennlinie, Fuß = außen

ULD – untere Liniendicke

Werden die optionalen Linienargumente nicht gesetzt, dann bleibt das Verhalten weiterhin durch die in Abschnitt 5.1.3, Seite 221 vorgestellten Linienbefehle konfigurierbar.

Die drei Felder *GS*, *US* und *ES* entsprechen Boxen, die die Breite des Kopf- bzw. Fußteils haben. Die entsprechenden Definitionen erscheinen in diesen Boxen linksbündig. Um somit etwas links- *und* rechtsseitig in den Boxen zu platzieren, kann der Zwischenraum mit `\hfill` gestreckt werden:

```
{\headmark\hfill\pagemark}
```

Um zusätzlich etwas zentriert erscheinen zu lassen, ist eine erweiterte Definition notwendig. Die Befehle `\rlap` und `\llap` setzen die übergebenen Argumente. Für LaTeX erscheint es aber so, dass diese Texte eine Breite von Null haben. Nur so erscheint der mittlere Text auch wirklich zentriert.

```
{\rlap{\headmark}\hfill zentriert\hfill\llap{\pagemark}}
```

Beispiel: Angenommen es wird die Dokumentklasse scrbook genutzt. Damit liegt ein zweiseitiges Layout vor. Für das Paket scrpage2 wird festgelegt, dass mit automatisch aktualisierten Kolumnentiteln gearbeitet wird und dass im Seitenstil `scrheadings` eine Trennlinie zwischen Kopf und Text gezogen wird.

```
\documentclass{scrbook}
\usepackage[automark,headsepline]{scrpage2}
```

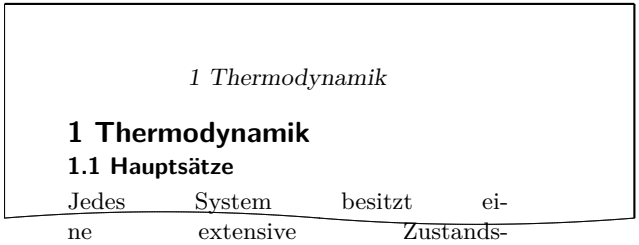
Mit Hilfe der Expertenschnittstelle werden zwei Seitenstile definiert. Der erste legt keine Linienargumente fest, im zweiten wird die Linie über dem Kopf mit einer Dicke von 1 pt und die Linie unter dem Kopf mit 0 pt festgelegt.

```
\defpagestyle{ohneLinien}{%
  {Beispiel\hfill\headmark}
  {\headmark\hfill ohne Linien}
  {\rlap{Beispiel}\hfill\headmark\hfill%
   \llap{ohne Linien}}
}{%
  {\pagemark\hfill}
  {\hfill\pagemark}
  {\hfill\pagemark\hfill}
}
\defpagestyle{mitLinien}{%
  (\textwidth,1pt)
  {mit Linien\hfill\headmark}
  {\headmark\hfill mit Linien}
  {\rlap{\KOMAScript}\hfill \headmark\hfill%
   \llap{mit Linien}}
  (0pt,0pt)
}{%
  (\textwidth,.4pt)
  {\pagemark\hfill}
  {\hfill\pagemark}
  {\hfill\pagemark\hfill}
  (\textwidth,1pt)
}
```

Gleich zu Beginn wird der Seitenstil `scrheadings` aktiviert. Mit `\chapter` wird ein neues Kapitel begonnen. Weiterhin wird automatisch durch `\chapter` der Seitenstil für diese Seite auf `plain` gesetzt. Das folgende `\chead` zeigt, wie durch Modifikation

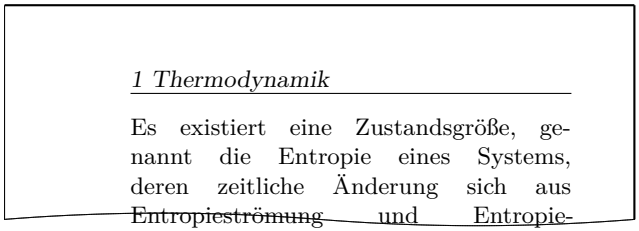
des plain-Stils ein Kolumnentitel erzeugt werden kann. Grundsätzlich sollte jedoch davon Abstand genommen werden, da sonst der Markierungscharakter der plain-Seite verloren geht. Es ist wichtiger anzuzeigen, dass hier ein neues Kapitel beginnt, als dass ein Abschnitt dieser Seite einen bestimmten Titel trägt.

```
\begin{document}
\pagestyle{scrheadings}
\chapter{Thermodynamik}
\chead[\leftmark]{}
\section{Hauptsätze}
Jedes System besitzt eine extensive Zustandsgröße
Energie. Sie ist in einem abgeschlossenen System
konstant.
```



Nach dem Seitenwechsel ist der Seitenstil `scrheadings` aktiv, und somit auch die Trennlinie aus den Paketoptionen sichtbar.

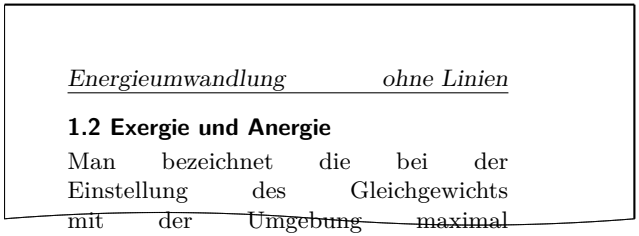
```
Es existiert eine Zustandsgröße, genannt die
Entropie eines Systems, deren zeitliche Änderung
sich aus Entropieströmung und Entropieerzeugung
zusammensetzt.
```



Wiederum nach einem Seitenwechsel wird auf manuelle Kolumnentitel gewechselt und der Seitenstil `ohneLinien` aktiviert. Da keine Linienargumente bei der Definition dieses Stils genutzt wurden, wird die Standard-Linienkonfiguration verwendet. Diese zeichnet hier eine Linie zwischen Kopf und Text, da `headsepline` als Argument für scrpage2 angegeben wurde.

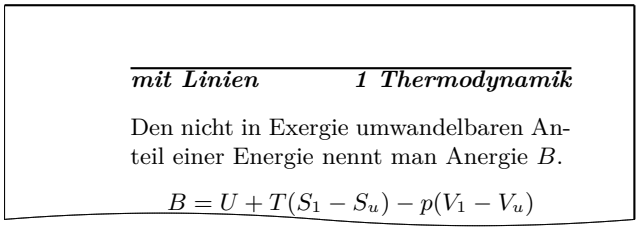
```
\manualmark
\pagestyle{ohneLinien}
```

```
\section{Exergie und Anergie}
\markright{Energieumwandlung}
Man bezeichnet die bei der Einstellung des
Gleichgewichts mit der Umgebung maximal gewinnbare
Arbeit als Exergie.
```



Nach dem Wechsel auf die folgende linke Seite wird der Seitenstil `mitLinien` aktiviert. Die Linieneinstellungen werden hier nun angewendet und entsprechend der Definition dargestellt.

```
\pagestyle{mitLinien}
\renewcommand{\headfont}{\itshape\bfseries}
Den nicht in Exergie umwandelbaren Anteil einer
Energie nennt man Anergie \Var{B}.
\[ B = U + T (S_1 - S_u) - p (V_1 - V_u)\]
\end{document}
```



5.2.3. Seitenstile verwalten

Bei längerer Arbeit mit verschiedenen Seitenstilen wird sich, je nach Geschmack und Aufgabenstellung, ein fester Satz an benutzten Stilen etablieren. Um nicht bei jedem neuen Projekt eine große Kopieraktion von den Daten eines Projekts zum neuen Projekt starten zu müssen, liest scrpage2 am Ende seiner Initialisierungsphase die Datei `scrpage.cfg` ein. In dieser Datei können dann Seitenstile definiert sein, die viele Projekte gemeinsam nutzen können.

Wochentag und Uhrzeit mit scrdate und scrtime

Zu KOMA-Script gehören auch zwei Pakete, um den Umgang mit Datum und Zeit über die beiden Standardbefehle `\today` und `\date` hinaus zu erweitern. Ebenso wie die anderen Pakete aus KOMA-Script können diese Pakete auch mit den Standardklassen verwendet werden.

v3.05a Ab KOMA-Script 3.05a verwenden diese Pakete die gemeinsame Versionsnummerierung von KOMA-Script. Begründet ist dies darin, dass sie auf eine bestimmte Version von scrkbase und scrbase angewiesen sind.

6.1. Der Wochentag mit scrdate

Mit Version 3.05a wurde der Funktionsumfang dieses Pakets erheblich erweitert. Neben dem aktuellen Wochentag kann dieses Paket nun den Wochentag jedes beliebigen Datums nach dem Gregorianischen Kalender bestimmen.

```
\CenturyPart{Jahr}
\DecadePart{Jahr}
```

v3.05a Die Anweisung `\CenturyPart` ergibt den Wert der Jahrhundert-Stellen eines Jahres. Die Anweisung `\DecadePart` ergibt hingegen den Wert der übrigen Stellen, also der Einer und Zehner. Dabei darf die Jahreszahl beliebig viele Stellen aufweisen. Der Wert kann direkt zur Zuweisung an einen Zähler oder für Berechnungen mit Hilfe von `\numexpr` verwendet werden. Für die Ausgabe als arabische Zahl ist `\the` voran zu stellen.

Beispiel: Sie wollen berechnen, in welchem Jahrhundert das aktuelle Jahr liegt und dies ausgeben.

```
Das Jahr \the\year\ ist das Jahr
\the\DecadePart{\year} des
\the\numexpr \CenturyPart{\year}+1\relax.
Jahrhunderts.
```

Als Ergebnis erhalten Sie:

> Das Jahr 2012 ist das Jahr 12 des 21. Jahrhunderts.

Bitte beachten Sie, dass hier die Zählweise verwendet wird, bei der das Jahr 2000 das Jahr 0 – also das erste Jahr – des 21. Jahrhunderts ist.

```
\DayNumber{Jahr}{Monat}{Tag}
\ISODayNumber{ISO-Datum}
```

v3.05a Diese beiden Anweisungen geben den Wert der Nummer des Wochentags zu einem Datum zurück. Sie unterscheiden sich nur in der Art der Angabe des Datums. Während bei

`\DayNumber` Jahr, Monat und Tag des gewünschten Datums eigene Parameter sind, wird bei `\ISODayNumber` das Datum in ISO-Schreibweise, *Jahr-Monat-Tag* angegeben. Dabei spielt es keine Rolle, ob Monat und Tag ein- oder zweistellig angegeben werden. Der Wert kann direkt zur Zuweisung an einen Zähler oder für Berechnungen mit Hilfe von `\numexpr` verwendet werden. Für die Ausgabe als arabische Zahl ist `\the` voran zu stellen.

Beispiel: Sie wollen die Nummer des Wochentags des 1. Mai 2027 wissen.

```
Der 1.~Mai~2027 hat die Wochentagsnummer
\the\ISODayNumber{2027-5-1}.
```

Als Ergebnis erhalten Sie:

> Der 1. Mai 2027 hat die Wochentagsnummer 6.

Als Besonderheit ist es sogar möglich, von einem vorgegebenen Datum eine gewünschte Anzahl an Tagen in die Zukunft oder Vergangenheit zu gehen.

Beispiel: Sie wollen die Nummer des Wochentags wissen, den wir in 12 Tagen haben und den wir 24 Tage vor dem 24. Dezember 2027 gehabt haben werden.

```
In 12~Tagen haben wir die Wochentagsnummer
\the\DayNumber{\year}{\month}{\day+12} und
24~Tage vor dem 24.~Dezember~2027 wird es
die Nummer \the\ISODayNumber{2027-12-24-24}
gewesen sein.
```

Als Ergebnis erhalten Sie beispielsweise:

> In 12 Tagen haben wir die Wochentagsnummer 2 und 24 Tage vor dem 24. Dezember 2027 wird es die Nummer 2 gewesen sein.

Die Wochentage werden dabei wie folgt nummeriert: Sonntag = 0, Montag = 1, Dienstag = 2, Mittwoch = 3, Donnerstag = 4, Freitag = 5 und Samstag = 6.

`\DayNameByNumber{`*Wochentagsnummer*`}`
`\DayName{`*Jahr*`}{`*Monat*`}{`*Tag*`}`
`\ISODayName{`*ISO-Datum*`}`

v3.05a

Üblicherweise ist man weniger an der Nummer eines Wochentags als dem Namen des Wochentags interessiert. Daher liefert die Anweisung `\DayNameByNumber` den Namen des Wochentags zu einer Wochentagsnummer zurück, die man beispielsweise mit einer der beiden zuvor erklärten Anweisungen `\DayNumber` oder `\ISODayNumber` bestimmt hat. Die beiden Anweisungen `\DayName` und `\ISODayName` liefern entsprechend den Wochentag zu einem bestimmten Datum.

Beispiel: Sie wollen den Wochentag des 24. Dezembers 2027 wissen.

```
Bitte zahlen Sie bis zum \ISODayName{2027-12-24},
den 24.\,12.~2027, die Summe von \dots
```

Als Ergebnis erhalten Sie:

> Bitte zahlen Sie bis zum Freitag, den 24. 12. 2027, die Summe von ...

Als Besonderheit ist es auch hier möglich, in gewissem Umfang Berechnungen anzustellen:

Beispiel: Sie wollen den Wochentag wissen, den wir in 12 Tagen haben und den wir 24 Tage vor dem 24. Dezember 2027 hatten.

```
In 12~Tagen haben wir einen
\DayName{\year}{\month}{\day+12} und
24~Tage vor dem 24.~Dezember~2027 ist ein
\ISODayName{2027-12-24-24}, während zwei Wochen
und drei Tage nach einem Mittwoch ein
\DayNameByNumber{3+2*7+3} folgt.
```

Als Ergebnis erhalten Sie beispielsweise:

> In 12 Tagen haben wir einen Dienstag und 24 Tage vor dem 24. Dezember 2027 ist ein Dienstag, während zwei Wochen und drei Tage nach einem Mittwoch ein Samstag folgt.

```
\ISOToday
\IsoToday
\todaysname
\todaysnumber
```

v3.05a

In den bisherigen Beispielen dieses Abschnitts wurde das aktuelle Datum immer recht umständlich über die TEX-Register `\year`, `\month`, `\day` bestimmt. Die Anweisungen `\ISOToday` und `\IsoToday` liefern direkt das aktuelle Datum in ISO-Schreibweise. Sie unterscheiden sich lediglich darin, dass `\ISOToday` Monat und Tag immer zweistellig ausgibt, während `\IsoToday` Monat und Tag bei Werten kleiner 10 einstellig ausgibt. Die Anweisung `\todaysname` bietet direkt den aktuellen Wochentag, während `\todaysnumber` den Wert des aktuellen Wochentags liefert. Näheres zur Verwendung dieses Wertes ist den obigen Erklärungen zu den Anweisungen `\DayNumber` und `\ISODayNumber` zu entnehmen.

Beispiel: Ich will Ihnen zeigen, an was für einem Wochentag dieses Dokument gesetzt wurde. Dazu schreibe ich:

```
Dieses Dokument entstand an einem \todaysname.
```

Das Ergebnis lautet:

> Dieses Dokument entstand an einem Donnerstag.

Wenn Sie den Namen des Tages in Kleinbuchstaben benötigen, weil das in der entsprechenden Sprache innerhalb des Satzes so üblich ist, können Sie das erreichen, obwohl die Namen der Wochentage in scrdate alle groß geschrieben sind. Greifen Sie mit

```
\MakeLowercase{\todaysname}
```

einfach auf die LaTeX-Anweisung `\MakeLowercase` zurück. Diese wandelt ihr Argument komplett in Kleinbuchstaben. Natürlich funktioniert dieser Tipp auch für obige Anweisungen `\DayNameByNumber`, `\DayName` und `\ISODayName`.

`\nameday{`*Name*`}`

So wie mit `\date` die Ausgabe von `\today` direkt geändert werden kann, setzt `\nameday` die Ausgabe von `\todaysname` auf den Wert *Name*.

Beispiel: Sie setzen mit `\date` das aktuelle Datum auf einen festen Wert. Für die Ausgabe des zugehörigen Wochentags interessiert es nur, dass dieser Tag ein Werktag war. Daher schreiben Sie

```
\nameday{Werktag}
```

und erhalten so mit dem Satz aus dem vorherigen Beispiel zu `\todaysname`:

> Dieses Dokument entstand an einem Werktag.

Für `\ISOToday` und `\IsoToday` existieren keine entsprechenden Anweisungen.

Wochentage in unterschiedlichen Sprachen

v3.08b

Das scrdate-Paket beherrscht derzeit die Sprachen Englisch (english und USenglish), Deutsch (german, ngerman, austrian und naustrian), Französisch (french), Italienisch (italian), Spanisch (spanish), Kroatisch (croatian), Finnisch (finnish), Norwegisch (norsk), Schwedisch (swedish) und Dänisch (danish), kann aber auch für andere Sprachen konfiguriert werden. Näheres dazu entnehme man `scrdate.dtx`.

Bei der aktuellen Version ist es egal, ob scrdate vor oder nach german, babel oder ähnlichen Paketen geladen wird, in jedem Falle wird die korrekte Sprache gewählt.

Etwas genauer ausgedrückt: Solange die Sprachauswahl in einer zu babel bzw. german kompatiblen Form erfolgt und die Sprache scrdate bekannt ist, wird die Sprache korrekt gewählt. Ist dies nicht der Fall, werden (US-)englische Ausdrücke verwendet.

6.2. Die aktuelle Zeit mit scrtime

Ein anderes Problem ist die Frage nach der aktuellen Zeit. Dieses kann mit Hilfe des Pakets scrtime gelöst werden.

`\thistime[`*Trennung*`]`
`\thistime*[`*Trennung*`]`

`\thistime` liefert die aktuelle Zeit in Stunden und Minuten. In der Ausgabe wird zwischen den Stunden und Minuten das optionale Argument *Trennung* gesetzt. Voreingestellt ist das Zeichen »:«.

`\thistime*` funktioniert fast genau wie `\thistime`. Der einzige Unterschied besteht darin, dass im Gegensatz zu `\thistime` bei `\thistime*` die Minutenangaben bei Werten kleiner 10 nicht durch eine vorangestellte Null auf zwei Stellen erweitert wird.

Beispiel: Die Zeile

```
Ihr Zug geht um \thistime\ Uhr.
```

liefert als Ergebnis beispielsweise eine Zeile wie

Ihr Zug geht um 15:19 Uhr.

oder

Ihr Zug geht um 23:09 Uhr.

Demgegenüber liefert die Zeile

```
Beim nächsten Ton ist es \thistime*[\ Uhr,\ ]
Minuten und 42 Sekunden.
```

als mögliches Ergebnis etwas wie:

Beim nächsten Ton ist es 8 Uhr, 41 Minuten und 42 Sekunden.

oder

Beim nächsten Ton ist es 23 Uhr, 9 Minuten und 42 Sekunden.

`\settime{`*Wert*`}`

`\settime` setzt die Ausgabe von `\thistime` und `\thistime*` auf einen festen *Wert*. Anschließend wird das optionale Argument von `\thistime` bzw. `\thistime*` ignoriert, da ja die komplette Zeichenkette, die `\thistime` bzw. `\thistime*` nun liefert, hiermit explizit festgelegt wurde.

`12h=`*Ein-Aus-Wert*

v3.05a

Mit der Option `12h` kann gewählt werden, ob die Zeit bei `\thistime` und `\thistime*` im 12-Stunden- oder 24-Stunden-Format ausgegeben werden soll. Als *Ein-Aus-Wert* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 40 verwendet werden. Wird die Option ohne Wert-Angabe verwendet, so wird der Wert `true` angenommen, also auf das 12-Stunden-Format geschaltet. Voreingestellt ist hingegen das 24-Stunden-Format.

Die Option kann wahlweise als Klassenoption bei `\documentclass`, als Paketoption bei `\usepackage` oder auch nach dem Laden von scrtime per `\KOMAoptions` oder `\KOMAoption` (siehe beispielsweise Abschnitt 2.4, Seite 30) gesetzt werden. Sie verliert jedoch bei einem Aufruf von `\settime` ihre Gültigkeit. Die Uhrzeit wird nach Verwendung dieser Anweisung nur noch mit dem dort angegebenen Wert im dort verwendeten Format ausgegeben.

Rein aus Gründen der Kompatibilität zu früheren Versionen von scrtime wird bei `\documentclass` und `\usepackage` auch noch die Option `24h` zur Umschaltung auf das 24-Stunden-Format unterstützt. Deren Verwendung wird jedoch nicht mehr empfohlen.

Adressdateien mit scraddr erschließen

7.1. Überblick

Das Paket scraddr ist eine kleine Beigabe zur Briefklasse von KOMA-Script. Ziel ist, die Benutzung von Adressdateien zu vereinfachen und ihre Anwendung flexibler zu gestalten. Im Grunde stellt das Paket nur einen Lademechanismus für Adressdateien bereit, die aus `\adrentry`- und neueren `\addrentry`-Einträgen bestehen, wie sie in Kapitel 4 ab Seite 209 beschrieben sind.

`\InputAddressFile{`*Dateiname*`}`

Der Befehl `\InputAddressFile` ist der zentrale Ladebefehl von scraddr. Er erwartet als obligatorisches Argument den Namen der einzulesenden Adressdatei. Wird diese Datei nicht gefunden, wird ein Fehler ausgegeben.

Für jeden Eintrag dieser Adressdatei wird eine Reihe von Makros generiert, die es ermöglichen, auf die Daten der Adressdatei zuzugreifen. Es soll an dieser Stelle nicht verschwiegen werden, dass dies bei großen Adressdateien sehr viel TEX-Speicher kostet.

```
\adrentry{Name}{Vorname}{Adresse}{Tel.}{F1}{F2}{Kommentar}
         {Kürzel}
\addrentry{Name}{Vorname}{Adresse}{Tel.}{F1}{F2}{F3}{F4}
          {Kürzel}
\adrchar{Anfangsbuchstaben}
\addrchar{Anfangsbuchstaben}
```

Der Aufbau der Adresseinträge in der Adressdatei wurde in Abschnitt 4.22 ab Seite 209 ausführlich besprochen. Die ebenfalls dort erwähnte Unterteilung der Adressdatei mit Hilfe von `\adrchar` oder `\addrchar` hat für scraddr keine Bedeutung und wird vom Paket ignoriert.

Die Zugriffsbefehle sind mit englischen, den Argumenten entsprechenden Bezeichnungen versehen.

```
\Name{Kürzel}
\FirstName{Kürzel}
\LastName{Kürzel}
\Address{Kürzel}
\Telephone{Kürzel}
\FreeI{Kürzel}
\FreeII{Kürzel}
\Comment{Kürzel}
\FreeIII{Kürzel}
\FreeIV{Kürzel}
```

Der Zugriff erfolgt anhand des Kürzels im letzten Argument eines Eintrags, das heißt Argument Nummer 8 für `\adrentry`-Einträge beziehungsweise Argument Nummer 9 für `\addrentry`-

Einträge. Das bedeutet auch, dass dieses Argument nicht leer sein darf. Um eine sichere Funktionsweise zu garantieren, empfiehlt es sich, das Kürzel nur als Folge von Buchstaben aufzubauen, wobei jedoch keine Umlaute benutzt werden dürfen.

Weiterhin ist zu beachten, dass bei mehrmaligem Auftreten eines Kürzels in den Einträgen die Angaben beim letzten Auftreten die gültigen sind.

7.2. Benutzung

Um das Paket benutzen zu können, ist eine gültige Adressdatei zu erstellen. Diese, hier `lotr.adr` genannt, könnte beispielsweise folgendermaßen aussehen:

```
\addrentry{Beutlin}{Frodo}%
          {Der Bühl\\ Beutelsend/Hobbingen im Auenland}{}%
          {Bilbo Beutlin}{Rauchen von Pfeifenkraut}%
          {der Ringträger}{Bilbos Erbe}{FRODO}
\adrentry{Gamdschie}{Samweis}%
         {Beutelhaldenweg 3\\Hobbingen im Auenland}{}%
         {Rosie Kattun}{Knullen}%
         {des Ringträgers treuester Gefährte}{SAM}
\adrentry{Bombadil}{Tom}%
         {Im Alten Wald}{}%
         {Goldbeere}{trällern von Nonsensliedern}%
         {Meister von Wald, Wasser und Berg}{TOM}
```

Das vierte Argument, die Telefonnummer, wurde hier leer gelassen. Erstens hat es in dem Zusammenhang keinen Sinn, und zweitens sollte dies ja auch möglich sein.

Mit dem oben beschriebenen Ladebefehl lesen wir die Adressdatei in unser Briefdokument ein:

```
\InputAddressFile{lotr}
```

Mit Hilfe der vorgestellten Makros können wir dann einen Brief an den alten Tom Bombadil schreiben, in dem wir ihn fragen, ob er sich noch an zwei Gefährten aus alter Zeit erinnern kann.

```
\begin{letter}{\Name{TOM}\\\Address{TOM}}
  \opening{Lieber \FirstName{TOM} \LastName{TOM},}

  oder \FreeIII{TOM}, wie Dich Deine geliebte \FreeI{TOM}
  nennt. Kannst Du Dich noch an einen Herrn
  \LastName{FRODO}, genauer gesagt \Name{FRODO}, denn es gab
  ja auch noch den Herrn \FreeI{FRODO}, erinnern. Er war
  \Comment{FRODO} im dritten Zeitalter und \FreeIV{FRODO}.
  Begleitet wurde er von \Name{SAM}, \Comment{SAM}.

  Beider Vorlieben waren sehr weltlich. Der
```

```
    \FirstName{FRODO} genoss das \FreeII{FRODO}, sein Gefährte
    schätzte eine gute Mahlzeit mit \FreeII{SAM}.

    Weißt du noch? Mithrandir hat Dir bestimmt viel von ihnen
    erzählt.
    \closing{"‘O Frühling und Sommerzeit
                und danach wieder Frühling!\\
              O Wind auf dem Wasserfall
                und Lachen des Laubes!"’}
\end{letter}
```

Die in diesem Beispiel in `\opening` verwendete Zusammensetzung aus `\FirstName{`*`Kürzel`*`}` und `\LastName{`*`Kürzel`*`}` kann auch direkt mittels `\Name{`*`Kürzel`*`}` erhalten werden.

Das fünfte und sechste Argument von `\adrentry` und `\addrentry` steht zur freien Verfügung. Mit den Makros `\FreeI` und `\FreeII` kann auf diese Inhalte zugegriffen werden. Im vorliegenden Fall wurde das fünfte Argument für die Person benutzt, die der Person des Eintrags am nächsten steht. Das sechste Argument enthält im Beispiel die besondere Vorliebe der jeweiligen Person. Das siebente Argument ist ebenfalls ein freier Eintrag. Der Zugriff erfolgt per `\Comment` oder `\FreeIII`. Der Zugriff auf das vierte freie Argument mittels `\FreeIV` ist nur für `\addrentry`-Einträge gültig. Bei `\adrentry`-Einträgen ist seine Verwendung nicht zulässig. Näheres hierzu finde sich im nächsten Abschnitt.

7.3. Paketoptionen für Warnungen

Wie im vorherigen Abschnitt erwähnt, ist die Benutzung des Zugriffsbefehls `\FreeIV` bei `\adrentry`-Einträgen nicht zulässig. Wie scraddr darauf reagiert, ist allerdings durch Paketoptionen konfigurierbar.

```
adrFreeIVempty
adrFreeIVshow
adrFreeIVwarn
adrFreeIVstop
```

Diese vier Optionen erlauben die Auswahl aus vier verschiedenen Reaktionen zwischen *Ignorieren* bis *Abbruch* falls bei einem `\adrentry`-Eintrag der Zugriffsbefehl `\FreeIV` verwendet wird:

`adrFreeIVempty` – Der Befehl `\FreeIV` wird einfach ignoriert.

`adrFreeIVshow` – Es wird die Warnung: »(entry FreeIV undefined at *Kürzel*)«, in den Text geschrieben.

`adrFreeIVwarn` – In der Log-Datei erscheint eine Warnung.

`adrFreeIVstop` – Der LaTeX-Lauf wird mit einer Fehlermeldung unterbrochen.

Zur Auswahl der gewünschten Reaktion wird die entsprechende Option beim Laden des Pakets im optionalen Argument des `\usepackage`-Befehls übergeben. Die standardmäßige Einstellung für das Paket scraddr ist die Option `adrFreeIVshow`.

Adressdateien aus Adressdatenbanken

In früheren Versionen von KOMA-Script war das Paket addrconv ein fester Bestandteil des KOMA-Script-Systems. Die hauptsächliche Verflechtung mit KOMA-Script bestand darin, dass mit Hilfe dieses Paketes aus Adressdatenbanken im BibTeX-Format Adressdateien für die KOMA-Script-Briefklasse oder für das scraddr-Paket erstellt werden konnten.

```
@address{HMUS,
   name =       {Hans Mustermann},
   title =      {Mag. art.},
   city =       {Heimstatt},
   zip =        01234,
   country =    {Germany},
   street =     {Mauerstra{\ss}e 1},
   phone =      {01234 / 5 67 89},
   note =       {Alles nur Erfindung},
   key =        {HMUS},
}
```

Aus Einträgen wie dem oben stehenden können mit Hilfe von BibTeX und verschiedenen BibTeX-Stilen die Adressdateien erstellt werden. Weiterhin gibt es spezielle LaTeX-Dateien, die es ermöglichen, aus den Adressdateien Telefon- und Adressverzeichnisse zu erstellen.

Das Paket addrconv war aber eigentlich ein eigenständiger Teil, der auch noch über die Belange von KOMA-Script hinaus Möglichkeiten bietet. Deshalb ist addrconv bereits seit einiger Zeit nicht mehr in KOMA-Script enthalten. Das Paket adrconv, nur ein »d«, ersetzt addrconv vollständig. Es muss, falls nicht bereits in Ihrer TeX-Distribution enthalten, von [Kie99] separat bezogen und installiert werden.

Grundlegende Fähigkeiten der KOMA-Script-Klassen mit Hilfe des Pakets scrextend anderen Klassen erschließen

Es gibt einige Möglichkeiten, die allen KOMA-Script-Klassen gemeinsam sind. Dies betrifft in der Regel nicht nur die Klassen scrbook, scrreprt und scrartcl, die als Ersatz für die Standardklassen book, report und article für Bücher, Berichte und Artikel gedacht sind, sondern in weiten Teilen auch die KOMA-Script-Klasse scrlttr2, die als Nachfolger von scrlettr für Briefe gedacht ist. Diese grundlegenden Möglichkeiten, die in den genannten Klassen zu finden sind, werden von KOMA-Script ab Version 3.00 auch von dem Paket scrextend bereit gestellt. Dieses Paket kann nicht mit KOMA-Script-Klassen, wohl aber mit vielen anderen Klassen verwendet werden. Der Versuch, das Paket mit einer KOMA-Script-Klasse zu laden, wird von scrextend erkannt und mit einer Warnung abgelehnt.

Neben den in diesem Kapitel beschriebenen Möglichkeiten gibt es einige weitere Möglichkeiten, die jedoch hauptsächlich für Klassen- und Paketautoren gedacht sind. Diese sind in Kapitel 10, ab Seite 251 zu finden. Das dort dokumentierte Paket scrbase wird von allen KOMA-Script-Klassen und dem Paket scrextend verwendet.

Auch das Paket scrlfile aus Kapitel 11 ab Seite 269 wird von allen KOMA-Script-Klassen und dem Paket scrextend geladen. Daher stehen auch dessen Möglichkeiten bei Verwendung von scrextend zur Verfügung.

Im Unterschied dazu wird das ebenfalls für Klassen- und Paketautoren gedachte Paket tocbasic (siehe Kapitel 13 ab Seite 284) nur von den Klassen scrbook, scrreprt und scrartcl geladen, so dass die dort definierten Möglichkeiten auch nur in diesen Klassen und nicht in scrextend zu finden sind. Natürlich kann tocbasic aber auch zusammen mit scrextend verwendet werden.

9.1. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde.

9.2. Kompatibilität zu früheren Versionen von KOMA-Script

Es gilt sinngemäß, was in Abschnitt 2.5 geschrieben wurde. .

9.3. Optionale, erweiterte Möglichkeiten

Das Paket scrextend kennt optional verfügbare, erweiterte Möglichkeiten. Das sind Möglichkeiten, die in der Grundeinstellung nicht vorhanden sind, aber zusätzlich ausgewählt werden

Tabelle 9.1.: Übersicht über die optional verfügbaren, erweiterten Möglichkeiten von scrextend

title	die Titelseiten werden auf die Möglichkeiten der KOMA-Script-Klassen erweitert; dies betrifft neben den Anweisungen für die Titelseiten auch die Option `titlepage` (siehe Abschnitt 9.7, ab Seite 248)

können. Diese Möglichkeiten sind beispielsweise deshalb optional, weil sie in Konflikt mit den Möglichkeiten der Standardklassen oder häufig benutzter Pakete stehen.

`extendedfeature=`*`Möglichkeit`*

Mit dieser Option kann eine optionale Möglichkeit von scrextend ausgewählt werden. Diese Option steht nur während des Ladens von scrextend zur Verfügung. Anwender geben diese Option daher als optionales Argument von `\usepackage{scrextend}` an. Eine Übersicht über die verfügbaren optionalen Möglichkeiten bietet Tabelle 9.1.

9.4. Entwurfsmodus

Es gilt sinngemäß, was in Abschnitt 3.3 geschrieben wurde.

9.5. Wahl der Schriftgröße für das Dokument

Es gilt sinngemäß, was in Abschnitt 3.5 geschrieben wurde.

9.6. Textauszeichnungen

LaTeX verfügt über eine ganze Reihe von Anweisungen zur Textauszeichnung. Neben der Wahl der Schriftart gehören dazu auch Befehle zur Wahl einer Textgröße oder der Textausrichtung. Näheres zu den normalerweise definierten Möglichkeiten ist [SKPH99], [Tea05b] und [Tea05a] zu entnehmen.

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

Im LaTeX-Kern ist bereits die Anweisung `\textsuperscript` definiert, mit der *`Text`* höher gestellt werden kann. Leider bietet LaTeX selbst keine entsprechende Anweisung, um Text tief statt hoch zu stellen. KOMA-Script definiert dafür `\textsubscript`. Ein Anwendungsbeispiel finden Sie in Abschnitt 3.6, Seite 55.

```
\setkomafont{Element}{Befehle}
\addtokomafont{Element}{Befehle}
\usekomafont{Element}
```

v2.8p

Mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` ist es möglich, die *Befehle* festzulegen, mit denen die Schrift eines bestimmten *Element*s umgeschaltet wird. Theoretisch könnten als *Befehle* alle möglichen Anweisungen einschließlich Textausgaben verwendet werden. Sie sollten sich jedoch unbedingt auf solche Anweisungen beschränken, mit denen wirklich nur ein Schriftattribut umgeschaltet wird. In der Regel werden dies Befehle wie `\normalfont`, `\rmfamily`, `\sffamily`, `\ttfamily`, `\mdseries`, `\bfseries`, `\upshape`, `\itshape`, `\slshape`, `\scshape` oder einer der Befehle `\Huge`, `\huge`, `\LARGE`, `\Large`, `\large`, `\normalsize`, `\small`, `\footnotesize`, `\scriptsize` und `\tiny` sein. Die Erklärung zu diesen Befehlen entnehmen Sie bitte [SKPH99], [Tea05b] oder [Tea05a]. Auch Farbumschaltungen wie `\normalcolor` sind möglich (siehe [Car99d] und [Ker07]). Die Verwendung anderer Anweisungen, inbesondere solcher, die Umdefinierungen vornehmen oder zu Ausgaben führen, ist nicht vorgesehen. Seltsames Verhalten ist in diesen Fällen möglich und stellt keinen Fehler dar.

Mit `\setkomafont` wird die Schriftumschaltung eines Elements mit einer völlig neuen Definition versehen. Demgegenüber wird mit `\addtokomafont` die existierende Definition lediglich erweitert. Es wird empfohlen, beide Anweisungen nicht innerhalb des Dokuments, sondern nur in der Dokumentpräambel zu verwenden. Beispiele für die Verwendung entnehmen Sie bitte den Abschnitten zu den jeweiligen Elementen. Namen und Bedeutung der einzelnen Elemente sind in Tabelle 3.2, Seite 57 aufgelistet. Allerdings werden davon in scrextend nur die Elemente für den Dokumenttitel, den schlauen Spruch, die Fußnoten und die `labeling`-Umgebung behandelt. Das Element `disposition` ist zwar auch verfügbar, wird jedoch von scrextend ebenfalls nur für den Dokumenttitel verwendet. Die Voreinstellungen sind den jeweiligen Abschnitten zu entnehmen.

Mit der Anweisung `\usekomafont` kann die aktuelle Schriftart auf diejenige umgeschaltet werden, die für das angegebene *Element* definiert ist.

Beispiel: Angenommen, Sie wollen, dass der Titel in Serifenschrift und rot gesetzt wird. Das erreichen Sie einfach mit:

```
\setkomafont{title}{\color{red}}
```

Für die Anweisung `\color{red}` wird das Paket color oder xcolor benötigt. Die zusätzliche Angabe von `\normalfont` ist in diesem Beispiel deshalb nicht notwendig, weil diese Anweisung bereits in der Definition des Titels enthalten ist. Das Beispiel setzt voraus, dass `extendedfeature=title` gesetzt ist (siehe Abschnitt 9.3, Seite 246).

9.7. Dokumenttitel

Es gilt sinngemäß, was in Abschnitt 3.7 geschrieben wurde. Es gibt jedoch einen Unterschied:

Die Möglichkeiten von scrextend zum Dokumenttitel gehören zu den optionalen, erweiterten Möglichkeiten und stehen daher nur zur Verfügung, wenn beim Laden des Pakets `extendedfeature=title` gewählt wurde (siehe Abschnitt 9.3, Seite 246). Darüber hinaus kann scrextend nicht mit einer KOMA-Script-Klasse zusammen verwendet werden. In allen Beispielen aus Abschnitt 3.7 muss daher bei Verwendung von scrextend

```
\documentclass{scrbook}
```

durch

```
\documentclass{book}
\usepackage[extendedfeature=title]{scrextend}
```

ersetzt werden.

9.8. Erkennung von rechten und linken Seiten

Es gilt sinngemäß, was in Abschnitt 3.11 geschrieben wurde.

9.9. Wahl eines vordefinierten Seitenstils

Eine der allgemeinen Eigenschaften eines Dokuments ist der Seitenstil. Bei LaTeX versteht man unter dem Seitenstil in erster Linie den Inhalt der Kopf- und Fußzeilen. Das Paket scrextend definiert selbst keine Seitenstile, nutzt aber erwartete Seitenstile.

`\titlepagestyle`

Auf einigen Seiten wird mit Hilfe von `\thispagestyle` automatisch ein anderer Seitenstil gewählt. Bei scrextend betrifft dies bisher nur die Titelseiten und auch dies nur, wenn mit `extendedfeature=title` gearbeitet wird (siehe Abschnitt 9.3, Seite 246). Welcher Seitenstil dies ist, wird dem Makro `\titlepagestyle` entnommen. In der Voreinstellung ist der Seitenstil `plain`. Dieser Seitenstil wird bereits im LaTeX-Kern vordefiniert und sollte daher immer verfügbar sein.

9.10. Vakatseiten

Es gilt sinngemäß, was in Abschnitt 3.13 geschrieben wurde.

9.11. Fußnoten

Die Fußnoten-Möglichkeiten der KOMA-Script-Klassen, die in Abschnitt 3.14 beschrieben sind, werden von scrextend ebenfalls bereitgestellt. In der Voreinstellung wird die Formatierung der Fußnoten jedoch der verwendeten Klasse überlassen. Dies ändert sich, sobald die Anweisung `\deffootnote` verwendet wird, die auf Seite 85 näher erläutert wird.

9.12. Schlauer Spruch

Es gilt sinngemäß, was in Abschnitt 3.17 geschrieben wurde. Allerdings werden von scrextend die Anweisungen `\setchapterpreamble` und `\setpartpreamble` nicht definiert. Ob die verwendete Klasse eine entsprechende Anweisung bietet, ist der Anleitung zur jeweiligen Klasse zu entnehmen.

9.13. Listen

Es gilt sinngemäß, was in Abschnitt 3.14 geschrieben wurde. Allerdings werden vom Paket scrextend nur die Umgebungen `labeling`, `addmargin` und `addmargin*` definiert. Alle anderen Listenumgebungen bleiben der Verantwortung und Kontrolle der verwendeten Klasse überlassen.

9.14. Randnotizen

Es gilt sinngemäß, was in Abschnitt 3.21 geschrieben wurde.

Teil II.

KOMA-Script für fortgeschrittene Anwender und Experten

In diesem Teil sind die Informationen für die Autoren von LaTeX-Paketen und -Klassen zu finden. Dies betrifft nicht nur Anweisungen, die bei der Implementierung neuer Pakete und Klassen nützlich sind, sondern auch Schnittstellen, die weitere Eingriffe in KOMA-Script erlauben. Darüber hinaus sind in diesem Teil auch obsolete Optionen und Anweisungen ebenso wie Hintergründe zur Implementierung von KOMA-Script zu finden.

Dieser Teil ist als Ergänzung zu den Informationen für Autoren von Artikeln, Berichten, Büchern und Briefen in Teil I gedacht. Nähere Informationen und Beispiele für diese Anwender sind jenem Teil zu entnehmen.

Grundlegende Funktionen im Paket scrbase

Das Paket scrbase stellt einige grundlegende Funktionen bereit, die sich an Autoren von Paketen und Klassen richten. Dabei kann es nicht nur für Wrapper-Klassen genutzt werden, die ihrerseits eine KOMA-Script-Klasse laden. Auch Autoren von Klassen, die ansonsten nichts mit KOMA-Script zu tun haben, können von der Funktionalität von scrbase profitieren.

10.1. Laden des Pakets

Während Anwender ein Paket mit Hilfe von `\usepackage` laden, verwenden Paket- und Klassenautoren `\RequirePackage`. Autoren von Wrapper-Paketen nutzen auch `\RequirePackageWithOptions`. Bei Verwendung von `\RequirePackage` können wie bei `\usepackage` Optionen angegeben werden. Demgegenüber erhält das Paket bei `\RequirePackageWithOptions` alle Optionen, mit denen zuvor das Wrapper-Paket geladen wurde. Näheres zu diesen Anweisungen ist [Tea06] zu entnehmen.

Das Paket scrbase benötigt intern die Funktionalität des Pakets keyval. Diese kann auch vom Paket xkeyval zur Verfügung gestellt werden. Bei Bedarf lädt scrbase selbst keyval.

Das Paket keyval erlaubt es, Schlüssel zu definieren und diesen Werten zuzuweisen. Auch die Optionen, die scrbase bereitstellt, verwenden die keyval-Syntax: *Schlüssel*=*Wert*.

`internalonly=`*Wert*

Von scrbase werden einige Verzweigungsanweisungen bereitgestellt. Dabei verwendet es primär die Bezeichnungen `\scr@`*Name*. Es handelt sich somit um interne Anweisungen. Diese werden auch intern von KOMA-Script verwendet. Paket- und Klassenautoren können diese Anweisungen ebenfalls verwenden, sollten sie aber nicht umdefinieren. Da einige dieser Anweisungen auch für Benutzer nützlich sein können, werden die gleichen Anweisungen normalerweise auch als `\`*Name* bereitgestellt. Da eventuell andere Pakete gleichnamige Anweisungen mit anderer Syntax bereitstellen könnten und es so zu Konflikten kommen könnte, kann der Anwender die Definition von `\`*Name* verhindern. Dazu gibt er entweder die Option ohne Wertangabe an. In diesem Fall werden nur die internen Verzweigungsanweisungen definiert. Oder er gibt genau die Anweisungen, die nicht definiert werden sollen, als Wert an, wobei er »`\`« durch »`/`« ersetzt.

Paket- und Klassenautoren sollten diese Option normalerweise nicht verwenden. Anwender können sie mit oder ohne Wertangabe entweder als globale Option bei `\documentclass` oder per `\PassOptionsToPackage` angeben.

Beispiel: Der Anwender will nicht, dass die Anweisungen `\ifVTeX` und `\ifundefinedorrelax` von scrbase definiert werden. Also verwendet er beim Laden der Klasse:

```
\documentclass%
  [internalonly=/ifVTeX/ifundefinedorrelax]%
  {foo}
```

Der Klassenname foo wird hier als Platzhalter für irgend eine Klasse verwendet. Die Bedeutungen der Anweisungen `\ifVTeX` und `\ifundefinedorrelax` sowie weitere Verzweigungsanweisungen sind Abschnitt 10.3 zu entnehmen.

10.2. Schlüssel als Eigenschaften von Familien und deren Mitgliedern

Wie bereits in Abschnitt 10.1 erwähnt, setzt scrbase bei Schlüsseln und deren Werte auf das Paket keyval. Allerdings erweitert es dessen Funktionalität. Während bei keyval ein Schlüssel einer Familie gehört, kennt keyval zu jeder Familie auch noch Familienmitglieder. Ein Schlüssel kann dann sowohl einer Familie als auch einem oder mehreren Familienmitgliedern gehören. Außerdem kann ein Wert einem Schlüssel eines Familienmitglieds, einem Schlüssel einer Familie oder einem Schlüssel aller Familienmitglieder zugewiesen werden.

```
\DefineFamily{Familienname}
\DefineFamilyMember[Mitglied]{Familienname}
```

scrbase muss aus verschiedenen Gründen die Mitglieder einer Familie kennen. Daher ist es notwendig, eine neue Familie zunächst mit `\DefineFamily` zu definieren und so eine leere Mitgliederliste zu erzeugen. Ist die Familie bereits definiert, so geschieht schlicht nichts. Es wird also auch nicht eine bereits existierende Mitgliederliste überschrieben.

Ein neues Mitglied wird der Familie dann mit der Anweisung `\DefineFamilyMember` bekannt gegeben. Existiert die Familie nicht, so führt dies zu einer Fehlermeldung. Existiert das Mitglied bereits, so geschieht nichts. Wird kein Mitglied angegeben, so bleibt das Mitglied nicht etwa leer, sondern es wird »`.\@currname.\@currext`« angenommen. Während des Ladens einer Klasse oder eines Pakets sind `\@currname` und `\@currext` zusammen der Dateiname.

Theoretisch wäre es möglich, mit einem leeren optionalen Argument *`Mitglied`* auch ein Mitglied ohne Name zu definieren. Dies würde jedoch der Familie selbst entsprechen. Es wird empfohlen als *`Familienname`* nur Buchstaben und Ziffern zu verwenden und das *`Mitglied`* immer mit einem anderen Zeichen zu verwenden. Anderenfalls könnte es passieren, dass sich Mitglieder einer Familie mit Mitgliedern anderer Familien überdecken.

scrbase definiert selbst bereits die Familie »`KOMA`« und fügt ihr das Mitglied »`.scrbase.sty`« hinzu. Grundsätzlich ist die Familie »`KOMA`« KOMA-Script vorbehalten. Es wird empfohlen für eigene Pakete den Namen des Gesamtpakets als Familie und den Namen einzelner Pakete im Gesamtpaket als Mitglied zu verwenden.

Beispiel: Angenommen Sie schreiben ein neues Gesamtpaket »Fleischermeister«. Darin befinden sich die Pakete `Salami.sty`, `Mettwurst.sty` und `Krakauer.sty`. Daher ent-

scheiden Sie sich für den Familienname »Fleischermeister« und fügen in jedem der Pakete die Zeilen

```
\DefineFamily{Fleischermeister}
\DefineFamilyMember{Fleischermeister}
```

ein. Dadurch werden dann beim Laden der drei genannten Pakete der Familie »Fleischermeister« je nach Paket eines der drei Mitglieder ».Salami.sty«, ».Mettwurst.sty« und ».Krakauer.sty« zugefügt. Am Ende sind dann alle drei Mitglieder definiert.

```
\DefineFamilyKey[Mitglied]{Familie}{Schlüssel}[Säumniswert]
                {Aktion}
```

Mit dieser Anweisung wird ein *Schlüssel* definiert. Ist ein *Mitglied* angegeben, so ist der *Schlüssel* eine Eigenschaft dieses Mitglieds der angegebenen *Familie*Familie. Ist kein Mitglied angegeben, so wird wieder das Mitglied ».\@currname.\@currext« angenommen. Wird später dem *Schlüssel* ein Wert zugewiesen, so wird *Aktion* ausgeführt, wobei der Wert als Parameter übergeben wird. Innerhalb von *Aktion* steht also »#1« für den übergebenen Wert. Wurde kein Wert übergeben, so wird stattdessen der *Säumniswert* eingesetzt. Falls kein *Säumniswert* angegeben wird, kann später der *Schlüssel* nur mit Wertübergabe verwendet werden.

Letztlich führt

```
\DefineFamilyKey[Mitglied]{Familie}{Schlüssel}
                [Säumniswert]{Aktion}
```

zu dem Aufruf

```
\define@key{FamilieMitglied}{Schlüssel}
          [Säumniswert]{Aktion}
```

wobei \define@key im keyval-Paket definiert ist (siehe [Car99b]).

Beispiel: Nehmen wir an, jedes der drei Pakete aus dem letzten Beispiel soll einen Schlüssel Aufschnitt erhalten. Wird dieser aufgerufen, so soll in jedem der Pakete entsprechend dem Aufrufwert ein Schalter gesetzt werden. Für das Paket Salami könnte das beispielsweise so aussehen:

```
\newif\if@Salami@Aufschnitt
\DefineFamilyKey{Fleischermeister}%
                {Aufschnitt}[true]{%
  \expandafter\let\expandafter\if@Salami@Aufschnitt
  \csname if#1\endcsname
}
```

Als Wert sind daher beim Aufruf `true` oder `false` erlaubt. Ein Test auf unerlaubte Werte existiert in diesem Beispiel nicht. Wird der Schlüssel später verwendet, so

muss entweder einer der erlaubten Werte zugewiesen oder ein Aufruf ohne Wertzuweisung verwendet werden. In letzterem Fall wird der Säumniswert *true* zugewiesen.

Für die anderen beiden Pakete kann das fast identisch definiert werden. Lediglich die Zeichenfolge »Salami« ist jeweils zu ersetzen.

`\FamilyProcessOptions[`*Mitglied*`]{`*Familie*`}`

Grundsätzlich ist die Erweiterung der Schlüssel von Familien auf Familien und Familienmitglieder dazu gedacht, dass Schlüssel beziehungsweise die Wertzuweisung an Schlüssel als ganz normale Klassen- oder Paketoptionen verwendet werden kann. Diese Anweisung stellt daher ein Erweiterung von `\ProcessOptions*` aus dem LaTeX-Kern dar (siehe [Tea06]). Dabei verarbeitet die Anweisung nicht nur Optionen, die mit `\DeclareOption` definiert wurden. Es werden auch alle Schlüssel eines angegebenen Familienmitglieds abgearbeitet. Wird das optionale Argument *Mitglied* nicht angegeben, so wird wieder das Mitglied »`.\@currname.\@currext`« verwendet.

Eine Besonderheit sind Schlüssel, die nicht einem Familienmitglied, sondern der Familie selbst zugeordnet sind, bei der also das Mitglied leer geblieben ist. Diese werden ebenfalls gesetzt und zwar noch bevor der Schlüssel des Mitglieds gesetzt wird.

Beispiel: Wenn in den Paketen aus den zurückliegenden Beispielen die Zeile

```
\FamilyProcessOptions{Fleischermeister}
```

ergänzt wird, so kann der Anwender bereits beim Laden der Pakete, die Eigenschaft `Aufschnitt` wählen. Wird die Option global, also bei `\documentclass` angegeben, so wird die Eigenschaft automatisch bei allen drei Paketen gesetzt, wenn alle drei Pakete geladen werden.

Es wird darauf hingewiesen, dass bei Paketen globale Optionen vor den lokal dem Paket zugewiesenen Optionen ausgeführt werden. Während bei der Abarbeitung der globalen Optionen unbekannte Werte für Optionen dazu führen, dass darüber lediglich in der `log`-Datei informiert und die Option ansonsten ignoriert wird, führt dies bei lokalen Optionen zu einer Fehlermeldung.

Man kann `\FamilyProcessOptions` wahlweise als Erweiterung von `\ProcessOption*` oder als Erweiterung des *Schlüssel=Wert*-Mechanismus von keyval verstehen. Letztlich werden mit Hilfe von `\FamilyProcessOptions` aus *Schlüssel=Wert*-Paaren Optionen.

`\FamilyExecuteOptions[`*Mitglied*`]{`*Familie*`}{`*Optionenliste*`}`

Diese Anweisung stellt ein Erweiterung von `\ExecuteOptions` aus dem LaTeX-Kern dar (siehe [Tea06]). Dabei verarbeitet die Anweisung nicht nur Optionen, die mit `\DeclareOption`

definiert wurden. Es werden auch alle Schlüssel eines angegebenen Familienmitglieds abgearbeitet. Wird das optionale Argument *Mitglied* nicht angegeben, so wird wieder das Mitglied »`.\@currname.\@currext`« verwendet.

Eine Besonderheit sind Schlüssel, die nicht einem Familienmitglied, sondern der Familie selbst zugeordnet sind, bei der also das Mitglied leer geblieben ist. Diese werden ebenfalls gesetzt und zwar noch bevor der Schlüssel des Mitglieds gesetzt wird.

Beispiel: Angenommen, die Option `Aufschnitt` soll in den zurückliegenden Beispielen bereits als Voreinstellung gesetzt werden, so müssen die Pakete nur um die Zeile

```
\FamilyExecuteOptions{Fleischermeister}{Aufschnitt}
```

ergänzt werden.

`\FamilyOptions{`*Familie*`}{`*Optionenliste*`}`
`\Family@Options{`*Familie*`}{`*Optionenliste*`}{`*Fehleraktion*`}`

Die *Optionenliste* hat dabei die Form:

Schlüssel`=`*Wert*`,`*Schlüssel*`=`*Wert*`...`

wobei für *Schlüssel*, für die ein Säumniswert definiert ist, die Wertzuweisung natürlich auch entfallen kann.

Im Gegensatz zu normalen Optionen, die mit `\DeclareOption` definiert wurden, können die *Schlüssel* auch noch nach dem Laden der Klasse oder des Pakets gesetzt werden. Dazu verwendet der Anwender `\FamilyOptions`. Dabei werden die *Schlüssel* aller Mitglieder der angegebenen *Familie* gesetzt. Existiert ein *Schlüssel* auch als Eigenschaft der Familie selbst, so wird dieser Familien-Schlüssel zuerst gesetzt. Danach folgen die Mitglieder-Schlüssel in der Reihenfolge, in der die Mitglieder definiert wurden. Existiert ein angegebener *Schlüssel* weder für die Familie noch für ein Mitglied der Familie, so wird von `\FamilyOptions` ein Fehler ausgegeben. Bei `\Family@Options` kann man hingegen die Fehleraktion selbst wählen. Diese interne Anweisung ist Klassen- und Paketautoren vorbehalten.

Beispiel: Sie ergänzen das Fleischermeister-Projekt um ein weiteres Paket Wurstsalat. Wird dieses Paket verwendet, so sollen alle Wurstpakete zunächst einmal Aufschnitt produzieren:

```
\ProvidesPackage{Wurstsalat}%
                 [2008/05/06 nonsense package]
\DefineFamily{Fleischermeister}
\DefineFamilyMember{Fleischermeister}
\FamilyProcessOptions{Fleischermeister}\relax
\FamilyOptions{Fleischermeister}{Aufschnitt}
```

Sollte noch kein Wurst-Paket geladen sein, so würde nun eine Fehlermeldung wegen der nicht definierten Option »`Aufschnitt`« ausgegeben. Das kann vermieden werden, wenn die letzte Zeile in

```
\Family@Options{Fleischermeister}{Aufschnitt}{}
```

geändert wird. Allerdings produzieren so Wurst-Pakete, die nach Fleischwurst geladen werden, keinen Aufschnitt. Dies kann man ebenfalls ändern, wenn die letzte Zeile erneut überarbeitet wird:

```
\AtBeginDocument{%
  \Family@Options{Fleischermeister}{Aufschnitt}{%
    \PackageWarning{Wurstsalat}{%
      Wurstsalat benötigt mindestens ein
      Wurst-Paket}%
  }%
}%
```

Statt auf jegliche Meldung zu verzichten, falls kein Wurst-Paket geladen ist, wird nun auch gleich noch eine Warnung ausgegeben.

`\FamilyOption{`*Familie*`}{`*Option*`}{`*Werteliste*`}`
`\Family@Option{`*Familie*`}{`*Option*`}{`*Werteliste*`}{`*Fehleraktion*`}`

Neben Optionen, die sich gegenseitig ausschließende Werte besitzen, kann es auch Optionen geben, die gleichzeitig mehrere Werte annehmen können. Für diese wäre es bei Verwendung von `\FamilyOptions` notwendig, der Option mehrfach einen Wert zuzuweisen und dabei die Option selbst mehrfach anzugeben. Stattdessen kann man einfach mit `\FamilyOption` einer einzigen *Option* eine ganze *Werteliste* zuweisen. Die *Werteliste* ist dabei eine durch Komma separierte Liste von Werten:

Wert`,`*Wert*`...`

In diesem Zusammenhang sei darauf hingewiesen, dass die Verwendung eines Kommas in einem Wert möglich ist, wenn man den Wert in geschweifte Klammern setzt. Die weiteren Funktionsweise ist der vorhergehenden Erklärung zu `\FamilyOptions` und `\Family@Options` zu entnehmen.

Beispiel: Das Paket Wurstsalat soll eine Option bekommen, über die man weitere Zutaten bestimmen kann. Für jede Zutat wird dabei wieder ein Schalter gesetzt.

```
\newif\if@salatmit@Zwiebeln
\newif\if@salatmit@Gurken
\newif\if@salatmit@Peperoni
\DefineFamilyKey{Fleischermeister}{SalatZusatz}{%
  \csname @salatmit@#1true\endcsname
}
```

Es wurden hier die drei Zutaten »Zwiebeln«, »Gurken« und »Peperoni« definiert. Eine Fehlerbehandlung für den Fall, dass der Anwender unbekannte Zutaten fordert, existiert nicht.

Für einen Salat mit Zwiebeln und Gurken, kann der Anwender

```
\FamilyOptions{Fleischermeister}{%
  SalatZusatz=Zwiebeln,SalatZusatz=Gurken}
```

oder einfach

```
\FamilyOption{Fleischermeister}%
             {SalatZusatz}{Zwiebeln,Gurken}
```

verwenden.

`\FamilyBoolKey[`*Mitglied*`]{`*Familie*`}{`*Schlüssel*`}{`*Schaltername*`}`
`\FamilySetBool{`*Familie*`}{`*Schlüssel*`}{`*Schaltername*`}{`*Wert*`}`

In den vorherigen Beispielen wurden schon mehrfach Schalter gesetzt. Im Beispiel der Option `Aufschnitt` war es dabei notwendig, dass der Anwender als Werte `true` oder `false` angibt. Es existierte keine Fehlerbehandlung, falls der Anwender einen falschen Wert verwendet. Da solche boolschen Schalter ein häufiger Anwendungsfall sind, kann man sie bei scrbase einfach mit `\FamilyBoolKey` definieren. Dabei sind die Argumente *Mitglied*, *Familie* und *Schlüssel* die gleichen wie bei `\DefineFamilyKey` (siehe Seite 253). Das Argument *Schaltername* ist der Name eines Schalter ohne den Präfix `\if`. Existiert dieser Schalter noch nicht, so wird er automatisch definiert und mit *false* voreingestellt. Intern verwendet `\FamilyBoolKey` dann `\FamilySetBool` als *Aktion* für `\DefineFamilyKey`. Der Säumniswert für eine solche Option ist immer `true`.

`\FamilySetBool` wiederum versteht als *Wert* neben `true` auch die Werte `on` und `yes` zum Einschalten und neben `false` auch die Werte `off` und `no` zum Ausschalten. Wird ein unbekannter Wert übergeben, so wird die Anweisung `\FamilyUnkownKeyValue` mit den Argumenten *Familie*, *Schlüssel* und *Wert* aufgerufen. Dadurch wird dann eine Fehlermeldung über unbekannte Wertzuweisung ausgegeben (siehe auch Seite 260).

Beispiel: Der Schlüssel `Aufschnitt` soll in den Wurstpaketen etwas robuster definiert werden. Außerdem sollen alle Wurstpakete denselben Schalter verwenden, so dass entweder alle Wurstpakete Aufschnitt produzieren oder keines.

```
\FamilyBoolKey{Fleischermeister}{Aufschnitt}%
                                {@Aufschnitt}
```

Ein Test, ob Aufschnitt produziert wird sähe dann so aus:

```
\if@Aufschnitt
   ...
\else
   ...
\fi
```

Dies wäre dann in allen drei Wurstpaketen identisch. Damit könnte man prinzipiell die Eigenschaft »Aufschnitt« auch als Eigenschaft der Familie definieren:

```
\@ifundefined{if@Aufschnitt}{%
```

```
    \expandafter\newif\csname if@Aufschnitt\endcsname
  }{}%
  \define@key{Fleischermeister}{Aufschnitt}[true]{%
    \FamilySetBool{Fleischermeister}{Aufschnitt}%
                                    {@Aufschnitt}%
                                    {#1}%
  }
```

oder einfacher

```
\FamilyBoolKey[]{Fleischermeister}{Aufschnitt}%
                                  {@Aufschnitt}
```

unter Ausnutzung des weiterführenden Hinweises auf Seite 253, der nicht nur für `\DefineFamilyKey`, sondern entsprechend auch für `\FamilyBoolKey` gilt.

```
\FamilyNumericalKey[Mitglied]{Familie}{Schlüssel}
                   [Säumniswert]{Makroname}{Werteliste}
\FamilySetNumerical{Familie}{Schlüssel}
                   {Makroname}{Werteliste}{Wert}
```

Während Schalter nur zwei Werte annehmen können, gibt es auch Schlüssel, die mehrere Werte kennen. So kann beispielsweise eine Ausrichtung nicht nur links oder nicht links, sondern beispielsweise links, mittig oder rechts sein. Intern unterscheidet man solche Einstellungen dann gerne mit Hilfe von `\ifcase`. Diese TEX-Anweisung erwartet wiederum einen nummerischen Wert. Daher heißt bei scrbase die Anweisung, mit der man via *Schlüssel* einem Makro eine Definition zuweisen kann, entsprechend `\FamilyNumericalKey`. Die *Werteliste* hat dabei die Form

{Wert}{Definition},{Wert}{Definition},...

Über diese *Werteliste* werden so nicht nur die erlaubten Werte für den *Schlüssels* angegeben. Für jeden erlaubten *Wert* wird auch gleich angegeben, wie bei Verwendung desselben das Makro *\Makroname* definiert werden soll. Üblicherweise werden als *Definition* schlicht Zahlenwerte angegeben. Es sind aber auch andere Angaben möglich. Derzeit gibt es aber die Einschränkung, dass *Definition* voll expandierbar sein muss und bei der Zuweisung auch expandiert wird.

Beispiel: Die Wurst für den Wurstsalat kann unterschiedlich geschnitten werden. So wäre es denkbar, dass der Aufschnitt einfach ungeschnitten bleibt oder in grobe oder feine Streifen geschnitten werden soll. Diese Information soll in der Anweisung `\Schnitt` gespeichert werden.

```
\FamilyNumericalKey{Fleischermeister}%
                   {SalatSchnitt}{Schnitt}{%
                     {Kein}{Kein},{Nein}{Kein},%
```

```
    {Grob}{Grob},%
    {Fein}{Fein}%
}
```

Dass nicht geschnitten werden soll, kann in diesem Fall vom Anwender sowohl mit

```
\FamilyOptions{Fleischermeister}{SalatSchnitt=Kein}
```

als auch mit

```
\FamilyOptions{Fleischermeister}{SalatSchnitt=Nein}
```

angegeben werden. In beiden Fällen würde `\Schnitt` mit dem Inhalt `Kein` definiert. Es kann durchaus sinnvoll sein, wie in diesem Beispiel dem Anwender mehrere Werte für denselben Zweck anzubieten.

Nun ist es sehr wahrscheinlich, dass die Schnittart nicht ausgegeben, sondern später ausgewertet werden soll. In diesem Fall sind die textuellen Definitionen aber eher unpraktisch. Definiert man den Schlüssel hingegen als

```
\FamilyNumericalKey{Fleischermeister}%
                   {SalatSchnitt}{Schnitt}{%
    {Kein}{0},{Nein}{0},%
    {Grob}{1},%
    {Fein}{2}%
}
```

so kann später einfach in der Form

```
\ifcase\Schnitt
  % ungeschnitten
\or
  % grob geschnitten
\else
  % fein geschnitten
\fi
```

unterschieden werden.

Intern wird von `\FamilyNumericalKey` dann `\DefineFamilyKey` mit der Anweisung `\FamilySetNumerical` verwendet. Wird an einen solchen Schlüssel ein unbekannter Wert übergeben, so wird von `\FamilySetNumerical` die Anweisung `\FamilyUnkownKeyValue` mit den Argumenten *Familie*, *Schlüssel* und *Wert* aufgerufen. Dadurch wird dann eine Fehlermeldung über unbekannte Wertzuweisung ausgegeben.

`\FamilyStringKey[`*Mitglied*`]{`*Familie*`}{`*Schlüssel*`}`
`[`*Säumniswert*`]{`*Makro*`}`

v3.08

Hier wird nun ein Schlüssel definiert, der jeden beliebigen Wert annehmen kann. Der Wert wird in dem angegebenen *Makro* gespeichert. Wird das optionales Argument für den *Säumniswert* weg gelassen, so entspricht der Aufruf

```
\DefineFamilyKey[Mitglied]{Familie}{Schlüssel}
                {\defMakro{#1}}.
```

Existiert das optionale Argument für den *Säumniswert* so entspricht der Aufruf

```
\DefineFamilyKey[Mitglied]{Familie}{Schlüssel}
                [Säumniswert]{\defMakro{#1}}.
```

Ist *Makro* noch nicht definiert, so wird es als leeres Makro definiert.

Beispiel: In der Voreinstellung sollen 250 g Wurstsalat erzeugt werden. Die Menge soll jedoch einfach per Option geändert werden können. Dazu wird die zu erstellende Menge im Makro `\Salatgewicht` gespeichert. Die Option, über die das Gewicht geändert werden kann, soll ebenfalls `Salatgewicht` heißen:

```
\newcommand*{\Salatgewicht}{250g}
\FamilyStringKey{Fleischermeister}%
                {Salatgewicht}[250g]{\Salatgewicht}
```

Soll nach einer vorherigen Änderung wieder die Standardmenge hergestellt werden, so kann der Anwender die Option einfach ohne Gewichtsangabe aufrufen:

```
\FamilyOptions{Fleischermeister}{Salatgewicht}
```

Das ist möglich, weil die Standardmenge bei der Definition auch als Säumniswert angegeben wurde.

In diesem Fall existieren keine unbekannten Werte, da alle Werte schlicht für eine Makrodefinition verwendet werden. Es ist jedoch zu beachten, dass in der Wertzuweisung an den Schlüssel keine Absätze enthalten sein dürfen.

```
\FamilyUnkownKeyValue{Familie}{Schlüssel}{Wert}{Werteliste}
\FamilyElseValues
```

Die Anweisung `\FamilyUnkownKeyValue` gibt eine Fehlermeldung über einen unbekannten Wert für einen bekannten Schlüssel aus. Als *Werteliste* wird dabei eine durch Komma separierte Liste von erlaubten Werten der Form

‘*Wert*’, ‘*Wert*’ ...

erwartet. Darüber hinaus kann die Anweisung `\FamilyElseValues` weitere erlaubte Werte in der Form

, ‘*Wert*’, ‘*Wert*’ ...

enthalten, die dann ebenfalls in der Fehlermeldung mit ausgegeben werden. Sowohl `\FamilySetBool` als auch `\FamilySetNumerical` leeren das Makro `\FamilyElseValues` am Ende automatisch – unabhängig davon, ob ein Fehler zu melden war oder nicht.

Beispiel: Für den Aufschnitt soll nun zusätzlich wählbar sein, ob er grob oder fein geschnitten werden soll. Dabei ist grob die Voreinstellung, die auch dann verwendet werden soll, wenn nicht angegeben wird, wie der Aufschnitt zu schneiden ist.

```
\@ifundefined{if@Feinschnitt}{%
  \expandafter
  \newif\csname if@Feinschnitt\endcsname}{}%
\@ifundefined{if@Aufschnitt}{%
  \expandafter
  \newif\csname if@Aufschnitt\endcsname}{}
\DefineFamilyKey{Fleischermeister}%
                {Aufschnitt}[true]{%
  \ifstr{#1}{fein}{%
    \@Aufschnitttrue
    \@Feinschnittrue
  }{%
    \@Feinschnittfalse
    \def\FamilyElseValue{, ‘fein’}%
    \FamilySetBool{Fleischermeister}{Aufschnitt}%
                                    {@Aufschnitt}%
                                    {#1}%
  }%
}%
```

Die hierbei verwendete Anweisung `\ifstr` ist auf Seite 263 in Abschnitt 10.3 erklärt.

10.3. Verzweigungen

Das Paket scrbase stellt eine ganze Reihe von Verzweigungsanweisungen zur Verfügung. Dabei wird nicht auf die TEX-Syntax von Verzweigungen, wie beispielsweise

```
\iftrue
  ...
\else
  ...
\fi
```

gebaut, sondern es wird die LATEX-Syntax mit Argumenten eingesetzt, wie man sie auch von LATEX-Anweisungen wie `\IfFileExists`, `\@ifundefined`, `\@ifpackageloaded` und vielen weiteren kennt. Einige Paketautoren ziehen es allerdings vor, die TEX-Syntax auch für Anwender in die LATEX-Ebene zu bringen. Da es sich bei den Verzweigungen von scrbase um recht grundlegende Möglichkeiten handelt, ist die Wahrscheinlichkeit gegeben, dass dabei gleichnamige Anwenderanweisungen verwendet würden. Dies würde selbst bei eigentlich gleicher Semantik zwangsläufig zu einem Problem aufgrund der unterschiedlichen Syntax führen. Deshalb definiert scrbase einige dieser Anweisungen zunächst als interne Anweisungen mit dem Präfix

\scr@. Für den Anwender werden dann gleichbedeutende Anweisungen ohne diesen Präfix bereit gestellt. Letzteres kann jedoch mit Hilfe der Option `internalonly` (siehe Abschnitt 10.1, Seite 251) verhindert werden.

Paket- und Klassenautoren sollten wie KOMA-Script selbst auch die internen Namen verwenden. Zur Vollständigkeit sind aber auch die Anwenderanweisungen angegeben.

```
\scr@ifundefinedorrelax{Name}{Dann-Teil}{Sonst-Teil}
\ifundefinedorrelax{Name}{Dann-Teil}{Sonst-Teil}
```

Diese Anweisung funktioniert prinzipiell wie `\@ifundefined` aus dem LaTeX-Kern (siehe [BCJ+05]). Es wird also der *Dann-Teil* ausgeführt, wenn *Name* der Name einer undefinierten Anweisung oder `\`*Name* derzeit `\relax` ist. Im Unterschied zu `\@ifundefined` wird allerdings `\`*Name* nicht als `\relax` definiert, wenn es zuvor undefiniert war. Unter Verwendung von ε-TeX wird in diesem Fall noch nicht einmal Hash-Speicher belegt.

```
\scr@ifpdftex{Dann-Teil}{Sonst-Teil}
\ifpdftex{Dann-Teil}{Sonst-Teil}
```

Wird mit pdfTeX gearbeitet, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Dabei ist es unerheblich, ob tatsächlich eine PDF-Datei ausgegeben werden soll, oder nicht. Diese Unterscheidung ist nur sehr selten wirklich von Nutzen. In der Regel sollte man eher auf die gewünschte Anweisung testen.

```
\scr@ifVTeX{Dann-Teil}{Sonst-Teil}
\ifVTeX{Dann-Teil}{Sonst-Teil}
```

Wird mit VTeX gearbeitet, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Diese Unterscheidung ist nur sehr selten wirklich von Nutzen. In der Regel sollte man eher auf die gewünschte Anweisung testen.

```
\scr@ifpdfoutput{Dann-Teil}{Sonst-Teil}
\ifpdfoutput{Dann-Teil}{Sonst-Teil}
```

Wird eine PDF-Datei erzeugt, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Dabei ist es unerheblich, ob die PDF-Datei mit Hilfe von pdfTeX oder mit Hilfe von VTeX erzeugt wird.

```
\scr@ifpsoutput{Dann-Teil}{Sonst-Teil}
\ifpsoutput{Dann-Teil}{Sonst-Teil}
```

Wird eine PostScript-Datei erzeugt, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. VTeX kann PostScript direkt erzeugten, was hier auch erkannt wird. Wird hingegen kein VTeX verwendet, ist aber ein Schalter `\if@dvips` definiert, so wird die Entscheidung darüber getroffen. KOMA-Script stellt `\if@dvips` in typearea bereit.

`\scr@ifdvioutput{`*Dann-Teil*`}{`*Sonst-Teil*`}`
`\ifdvioutput{`*Dann-Teil*`}{`*Sonst-Teil*`}`

Wird eine DVI-Datei erzeugt, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Es wird immer dann davon ausgegangen, dass eine DVI-Datei erzeugt wird, wenn keine direkte Ausgabe einer PDF- oder Postscript-Datei erkannt werden kann.

`\ifnotundefined{`*Name*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

Bei Verwendung von ε-TEX wird hier tatsächlich getestet, ob die Anweisung mit dem angegebenen *Name* noch undefiniert ist. Dies ist der Normalfall, da die Verwendung von ε-TEX für LATEX offiziell empfohlen ist. Wird trotzdem kein ε-TEX verwendet, ist dies hingegen schlicht die Umkehrung von `\scr@ifundefinedorrelax`. Von dieser Anweisung gibt es keine interne Variante.

`\ifstr{`*Zeichenfolge*`}{`*Zeichenfolge*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

Die beiden Argumente *Zeichenfolge* werden expandiert und dann verglichen. Sind sie dabei gleich, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Von dieser Anweisung gibt es keine interne Variante.

`\ifnumber{`*Zeichenfolge*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

Hier werden keine Zahlen verglichen. Der *Dann-Teil* wird vielmehr ausgeführt, wenn die Expansion der *Zeichenfolge* nur aus Ziffern besteht. Anderenfalls wird der *Sonst-Teil* verwendet. Von dieser Anweisung gibt es keine interne Variante.

`\ifdimen{`*Zeichenfolge*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

Hier werden keine Längen verglichen. Der *Dann-Teil* wird vielmehr ausgeführt, wenn die Expansion der *Zeichenfolge* eine gültige Länge mit einer gültigen Längeneinheit ist. Anderenfalls wird der *Sonst-Teil* verwendet. Von dieser Anweisung gibt es keine interne Variante.

`\if@atdocument` *Dann-Teil* `\else` *Sonst-Teil* `\fi`

Diese Verzweigung existiert ganz bewusst nur als interne Anweisung. Dabei entspricht `\if@atdocument` in der Dokumentpräambel `\iffalse`, nach `\begin{document}` entspricht `\if@document` hingegen `\iftrue`. Klassen und Paketautoren können dieses Anweisung manchmal sinnvoll nutzen, wenn sich Anweisungen in der Dokumentpräambel anders verhalten sollen als innerhalb des Dokuments. Es ist zu beachten, dass es sich bei dieser Anweisung um eine Verzweigung in TEX-Syntax und nicht in LATEX-Syntax handelt!

10.4. Definition sprachabhängiger Bezeichner

Normalerweise muss man zur Definition oder zur Änderung sprachabhängiger Begriffe Anweisungen wie `\captionsngerman` so umdefinieren, dass zusätzlich zu den bisherigen Begriffen auch die neuen oder geänderten definiert werden. Erschwert wird dieses Vorhaben dadurch, dass beim Laden eines Paketes wie german oder ngerman diese Anweisungen von den Paketen erneut definiert werden. Bei den genannten Paketen geschieht dies leider in einer Form, die alle zuvor gemachten Änderungen zunichte macht. Aus diesem Grund ist es sinnvoll die Änderungen mit Hilfe von `\AtBeginDocument` bis `\begin{document}`, also bis nach dem Laden aller Pakete, zu verzögern. Auch der Anwender muss entweder von `\AtBeginDocument` Gebrauch machen oder aber seine Änderungen nicht in die Dokumentpräambel, sondern hinter `\begin{document}` einfügen. Das Paket scrbase bietet ihm für die Definition selbst einige zusätzliche Anweisungen.

```
\providecaptionname{Sprache}{Begriff}{Inhalt}
\newcaptionname{Sprache}{Begriff}{Inhalt}
\renewcaptionname{Sprache}{Begriff}{Inhalt}
```

Mit Hilfe dieser drei Anweisungen ist es möglich, einem *Begriff* in Abhängigkeit der *Sprache* einen *Inhalt* zuzuweisen. Der *Begriff* ist dabei immer ein Makro. Die Arbeitsweise der drei Anweisungen unterscheidet sich je nachdem, ob eine *Sprache* und ein *Begriff* innerhalb dieser *Sprache* zum Zeitpunkt des Aufrufs bereits definiert ist.

Ist eine *Sprache* nicht definiert, so tut `\providecaptionname` nichts weiter, als dies in der log-Datei zu vermerken. Dabei wird für jede Sprache nur einmal eine entsprechende Information in die log-Datei geschrieben. Ist die Sprache definiert, enthält aber bisher keinen entsprechenden *Begriff*, so wird er mit dem angegebenen *Inhalt* definiert. Ist der *Begriff* hingegen in der *Sprache* bereits definiert, so wird er nicht umdefiniert, sondern ein entsprechender Hinweis in die log-Datei geschrieben.

Die Anweisung `\newcaptionname` verhält sich etwas anders. Ist bei ihr die *Sprache* nicht definiert, dann wird diese neu definiert, indem eine entsprechende Anweisung definiert wird. Für die *Sprache* ngerman wäre das beispielsweise `\captionsngerman`. Außerdem wird darüber auch in der log-Datei informiert. Ist die *Sprache* definiert, der *Begriff* in dieser *Sprache* aber noch nicht vorhanden, so wird er mit dem gewünschten *Inhalt* definiert. Ist der *Begriff* in der *Sprache* bereits vorhanden, so wird eine Fehlermeldung ausgegeben.

Die Anweisung `\renewcaptionname` verhält sich noch einmal anders. Ist die *Sprache* nicht definiert, so wird eine Fehlermeldung ausgegeben. Ist die *Sprache* definiert, der *Begriff* in dieser *Sprache* jedoch nicht, so wird ebenfalls eine Fehlermeldung ausgegeben. Ist der *Begriff* in der *Sprache* definiert, so wird er auf den gewünschten *Inhalt* umdefiniert.

KOMA-Script selbst verwendet `\providecaptionname` um die Begriffe aus Abschnitt 17.4 zu definieren.

Beispiel: Möchten Sie statt »Abbildung« in den Abbildungsunterschriften lieber »Abb.« stehen haben, so erreichen Sie dies mit

```
\renewcaptionname{ngerman}{\figurename}{Abb.}
```

Da mit `\renewcaptionname` nur bereits vorhandene Begriffe in vorhandenen Sprachen umdefiniert werden können, müssen Sie diese Anweisung nach `\begin{document}` setzen oder mit Hilfe von `\AtBeginDocument` verzögern. Außerdem werden Sie eine Fehlermeldung erhalten, wenn Sie kein Paket zur Sprachumschaltung auf `ngerman` verwenden.

In Tabelle 10.1 ist ein Überblick über die üblicherweise von Klassen und Sprachpaketen definierten Begriffe und deren Verwendung zu finden.

Tabelle 10.1.: Überblick über sprachabhängige Begriffe in den üblichen Sprachpaketen

`\abstractname`
: Überschrift für die Zusammenfassung

`\alsoname`
: »Siehe auch« bei ergänzenden Verweisen im Stichwortverzeichnis

`\appendixname`
: »Anhang« in der Kapitelüberschrift eines Anhangs

`\bibname`
: Überschrift für das Literaturverzeichnis

`\ccname`
: Spitzmarke für den Verteiler in Briefen

`\chaptername`
: »Kapitel« in der Kapitelüberschrift

`\contentsname`
: Überschrift für das Inhaltsverzeichnis

`\enclname`
: Spitzmarke für die Anlagen bei Briefen

`\figurename`
: Spitzmarke in der Abbildungsunterschrift

`\glossaryname`
: Überschrift für das Glossar

...

Tabelle 10.1.: Überblick über übliche sprachabhängige Begriffe (*Fortsetzung*)

`\headtoname`
: »An« im Briefkopf

`\indexname`
: Überschrift für das Stichwortverzeichnis

`\listfigurename`
: Überschrift für das Abbildungsverzeichnis

`\listtablename`
: Überschrift für das Tabellenverzeichnis

`\pagename`
: »Seite« in der Seitennummer von Briefen

`\partname`
: »Teil« in der Teileüberschrift

`\prefacename`
: Überschrift für das Vorwort

`\proofname`
: Spitzmarke bei Beweisen

`\refname`
: Überschrift für das Quellenverzeichnis

`\seename`
: »Siehe« bei Verweisen im Stichwortverzeichnis

`\tablename`
: Spitzmarke in der Tabellenunter- bzw. -überschrift

10.5. Identifikation von KOMA-Script

Auch wenn scrbase unabhängig von KOMA-Script auch für andere Pakete und Klassen verwendet werden kann, so ist es dennoch ein KOMA-Script-Paket. Als solches enthält es auch Anweisungen, die es als KOMA-Script-Paket identifizieren.

`\KOMAScript`

Diese Anweisung setzt schlicht die Wortmarke »KOMA-Script« in serifenloser Schrift und mit leichter Sperrung des in Versalien gesetzten Teils. \KOMA-Script wird übrigens bei

Bedarf von allen KOMA-Script-Klassen und -Paketen definiert. Die Definition erfolgt mit `\DeclareRobustCommand`.

`\KOMAScriptVersion`

Bei KOMA-Script ist in dieser Anweisung die Hauptversion von KOMA-Script in der Form »*Datum Version* KOMA-Script« abgelegt. Diese Hauptversion ist für alle KOMA-Script-Klassen und alle KOMA-Script-Pakete, die von den Klassen verwendet werden, gleich. Daher kann sie auch nach dem Laden von scrbase abgefragt werden. Diese Anleitung wurde beispielsweise mit der KOMA-Script-Version »2012/03/08 v3.10a BETA KOMA-Script« erstellt.

10.6. Erweiterungen des LaTeX-Kerns

In einigen Fällen stellt der LaTeX-Kern selbst Anweisungen zur Verfügung, lässt aber ganz ähnliche Anweisungen, die ebenfalls häufiger benötigt werden oder eigentlich nahe liegen, vermissen. Einige wenige solcher Anweisungen für Klassen- und Paketautoren stellt scrbase zur Verfügung.

`\ClassInfoNoLine{`*Klassenname*`}{`*Information*`}`
`\PackageInfoNoLine{`*Paketname*`}{`*Information*`}`

Der LaTeX-Kern bietet dem Klassen- und Paketautor zwar Anweisungen wie `\PackageInfo` und `\ClassInfo`, um Informationen mit aktueller Zeilennummer in die Log-Datei zu schreiben. Er bietet neben `\PackageWarning` und `\ClassWarning`, die Warnungen mit aktueller Zeilennummer ausgeben, auch die beiden Anweisungen `\PackageWarningNoLine` und `\ClassWarningNoLine`, um Warnungen ohne Zeilennummer auszugeben. Die nahe liegenden Anweisungen `\ClassInfoNoLine` und `\PackageInfoNoLine`, um auch Informationen ohne Zeilennummer in die Log-Datei zu schreiben, fehlen jedoch. Diese werden von scrbase bereit gestellt.

`\l@addto@macro{`*Anweisung*`}{`*Erweiterung*`}`

Der LaTeX-Kern bietet mit `\g@addto@macro` eine interne Anweisung, um die Definition eines Makro *Anweisung* global um den Code *Erweiterung* zu erweitern. Dies funktioniert in dieser Form nur für Makros ohne Argumente. Dennoch könnte man diese Anweisung in einigen Fällen auch in einer Form benötigen, die lokal zur aktuellen Gruppe arbeitet. Diese wird mit `\l@addto@macro` von scrbase bereit gestellt. Eine Alternative stellt hier die Verwendung des Pakets etoolbox dar, das eine ganze Reihe solcher Anweisungen für unterschiedliche Zwecke bietet (siehe [Leh11]).

10.7. Erweiterungen der mathematischen Fähigkeiten von ε-TEX

Das für LaTeX inzwischen verwendete und von KOMA-Script vorausgesetzte ε-TEX besitzt mit `\numexpr` erweiterte Möglichkeiten zur Berechnung einfacher Ausrücke mit TEX-Zählern und ganze Zahlen. Als Operationen werden dabei die vier Grundrechenarten und Klammern unterstützt. Bei der Division wird korrekt gerundet. Manchmal sind weitere Operationen nützlich.

`\XdivY{`*Dividend*`}{`*Divisor*`}`
`\XmodY{`*Dividend*`}{`*Divisor*`}`

v3.05a

Die Anweisung `\XdivY` liefert den Wert des Ganzzahlquotienten, die Anweisung `\XmodY` den Wert des Rests der Division mit Rest. Diese Art der Division ist nach der Gleichung

$$\mathit{Dividend} = \mathit{Divisor} \cdot \mathit{Ganzzahlquotient} + \mathit{Rest}$$

definiert, wobei *Dividend* und *Rest* ganze Zahlen und *Rest* außerdem größer oder gleich 0 und kleiner als *Divisor* ist. Der *Divisor* ist eine natürliche Zahl (ohne die 0).

Der Wert kann jeweils zur Zuweisung an einen Zähler oder direkt innerhalb eines Ausdrucks mit `\numexpr` verwendet werden. Zur Ausgabe als arabische Zahl ist `\the` voran zu stellen.

Paketabhängigkeiten mit scrlfile beherrschen

Die Einführung von LaTeX 2ε brachte 1994 eine Menge Neuerungen im Umgang mit LaTeX-Erweiterungen. So stehen dem Paketautor heute eine ganze Reihe von Befehlen zur Verfügung, um festzustellen, ob ein anderes Paket oder eine bestimmte Klasse verwendet wird und ob dabei bestimmte Optionen zur Anwendung kommen. Der Paketautor kann selbst andere Pakete laden oder diesen Optionen mit auf den Weg geben für den Fall, dass sie später noch geladen werden. Es bestand daher die Hoffnung, dass es künftig unerheblich wäre, in welcher Reihenfolge Pakete geladen werden. Diese Hoffnung hat sich leider nicht erfüllt.

11.1. Die Sache mit den Paketabhängigkeiten

Immer häufiger definieren unterschiedliche Pakete den gleichen Befehl neu oder um. Dabei ist es dann sehr entscheidend, in welcher Reihenfolge die Pakete geladen werden. Manchmal ist das für den Anwender kaum zu überschauen. In manchen Fällen ist es auch notwendig, einfach nur in irgendeiner Form auf das Laden eines anderen Paketes zu reagieren. Auch das ist nicht immer ganz einfach.

Nehmen wir als einfaches Beispiel das Laden des longtable-Paketes bei Verwendung von KOMA-Script. Das longtable-Paket definiert seine eigene Form von Tabellenüberschriften. Diese passen perfekt zu den Standardklassen. Sie passen aber überhaupt nicht zu den Voreinstellungen für die Tabellenüberschriften von KOMA-Script und reagieren auch nicht auf die entsprechenden Möglichkeiten der Konfiguration. Um dieses Problem zu lösen, müssen die Befehle von longtable, die für die Tabellenüberschriften zuständig sind, von KOMA-Script umdefiniert werden. Allerdings sind die KOMA-Script-Klassen bereits abgearbeitet, wenn das Paket geladen wird.

Bisher bestand die einzige Möglichkeit, dieses Problem zu lösen darin, die Umdefinierung mit Hilfe von `\AtBeginDocument` auf einen späteren Zeitpunkt zu verschieben. Will der Anwender die entsprechende Anweisung jedoch selbst umdefinieren, so sollte er dies eigentlich ebenfalls in der Präambel tun. Das kann er jedoch nicht, weil KOMA-Script ihm dabei in die Quere kommt. Er müsste die Umdefinierung also ebenfalls mit Hilfe von `\AtBeginDocument` durchführen.

Aber eigentlich müsste KOMA-Script die Abarbeitung gar nicht auf den Zeitpunkt von `\begin{document}` verschieben. Es würde genügen, wenn sie bis unmittelbar nach dem Laden von longtable verzögert werden könnte. Leider fehlen entsprechende Anweisungen im LaTeX-Kern. Das Paket scrlfile bringt hier Abhilfe.

Ebenso wäre es denkbar, dass man vor dem Laden eines bestimmten Paketes gerne die Bedeutung eines Makros in einem Hilfsmakro retten und nach dem Laden des Paketes wieder restaurieren will. Auch das geht mit scrlfile.

Die Anwendung von scrlfile ist nicht auf die Abhängigkeit von Paketen beschränkt. Auch Ab-

hängigkeiten von anderen Dateien können berücksichtigt werden. So kann beispielsweise dafür gesorgt werden, dass das nicht unkritische Laden einer Datei wie `french.ldf` automatisch zu einer Warnung führt.

Obwohl das Paket in erster Linie für andere Paketautoren interessant sein dürfte, gibt es durchaus auch Anwendungen für normale LaTeX-Benutzer. Deshalb sind in diesem Kapitel auch für beide Gruppen Beispiele aufgeführt.

11.2. Aktionen vor und nach dem Laden

Mit scrlfile können vor und nach dem Laden von Dateien Aktionen ausgelöst werden. Bei den dazu verwendeten Befehlen wird zwischen allgemeinen Dateien, Klassen und Paketen unterschieden.

```
\BeforeFile{Datei}{Anweisungen}
\AfterFile{Datei}{Anweisungen}
```

Mit Hilfe von `\BeforeFile` kann dafür gesorgt werden, dass die *Anweisungen* vor dem nächsten Laden einer bestimmten *Datei* ausgeführt werden. Vergleichbar arbeitet `\AfterFile`. Nur werden die *Anweisungen* hier erst nach dem Laden der *Datei* ausgeführt. Wird die Datei nie geladen, so werden die *Anweisungen* in beiden Fällen natürlich auch nie ausgeführt. Bei *Datei* sind etwaige Dateiendungen wie bei `\input` alt Teil des Dateinamens anzugeben.

Um die Funktionalität bereitstellen zu können, bedient sich scrlfile der bekannten LaTeX-Anweisung `\InputIfFileExists`. Diese wird hierzu umdefiniert. Falls die Anweisung nicht die erwartete Definition hat, gibt scrlfile eine Warnung aus. Dies geschieht für den Fall, dass die Anweisung in späteren LaTeX-Versionen geändert wird oder bereits von einem anderen Paket umdefiniert wurde.

Die Anweisung `\InputIfFileExists` wird von LaTeX immer verwendet, wenn eine Datei geladen werden soll. Dies geschieht unabhängig davon, ob die Datei mit `\LoadClass`, `\documentclass`, `\usepackage`, `\RequirePackage`, `\include` oder vergleichbaren Anweisungen geladen wird. Lediglich

```
\input foo
```

lädt die Datei `foo` ohne Verwendung von `\InputIfFileExists`. Sie sollten daher stattdessen immer

```
\input{foo}
```

verwenden. Beachten Sie die Klammern um den Dateinamen!

```
\BeforeClass{Klasse}{Anweisungen}
\BeforePackage{Paket}{Anweisungen}
```

Diese beiden Befehle arbeiten vergleichbar zu `\BeforeFile` mit dem einen Unterschied, dass die *Klasse* beziehungsweise das *Paket* mit seinem Namen und nicht mit seinem Dateinamen angegeben wird. Die Endungen »`.cls`« und »`.sty`« entfallen hier also.

```
\AfterClass{Klasse}{Anweisungen}
\AfterAtEndOfClass{Klasse}{Anweisungen}
\AfterClass*{Klasse}{Anweisungen}
\AfterClass+{Klasse}{Anweisungen}
\AfterClass!{Klasse}{Anweisungen}
\AfterPackage{Paket}{Anweisungen}
\AfterPackage*{Paket}{Anweisungen}
\AfterPackage+{Paket}{Anweisungen}
\AfterPackage!{Paket}{Anweisungen}
\AfterAtEndOfPackage{Klasse}{Anweisungen}
```

Die Anweisungen `\AfterClass` und `\AfterPackage` arbeiten weitgehend wie `\AfterFile` mit dem winzigen Unterschied, dass die *Klasse* beziehungsweise das *Paket* mit seinem Namen und nicht mit seinem Dateinamen angegeben wird. Die Endungen ».cls« und ».sty« entfallen hier also.

Bei den Sternvarianten gibt es eine zusätzliche Funktionalität. Wurde oder wird die entsprechende Klasse oder das entsprechende Paket bereits geladen, so werden die *Anweisungen* nicht nach dem nächsten Laden, sondern unmittelbar ausgeführt.

v3.09 Bei der Plusvariante werden die *Anweisungen* sicher erst dann ausgeführt, wenn die Klasse oder das Paket vollständig geladen wurden. Der Unterschied zwischen der Stern- und der Plusvariante kommt nur zum Tragen, wenn die Anweisung verwendet wird, wenn das Laden der Klassen bzw. des Pakets zwar bereits begonnen hat, aber noch nicht beendet wurde. Wenn das Laden der Klasse bzw. des Pakets noch nicht abgeschlossen wurde, werden die *Anweisungen* in allen Fällen vor den in der Klasse bzw. dem Paket mit `\AtEndOfClass` oder `\AtEndOfPackage` verzögerten Anweisungen ausgeführt.

Um eine Ausführung nach den in der Klasse oder dem Paket selbst mit `\AtEndOfClass` oder `\AtEndOfPackage` verzögerten Anweisungen sicher zu stellen, ist die Variante mit Ausrufezeichen zu verwenden. In diesem Fall werden die *Anweisungen* nicht mehr im Kontext der angegebenen Klasse bzw. des angegebenen Pakets ausgeführt.

Will man nur für den Fall, dass die Klasse bzw. das Paket noch nicht geladen wurde, erreichen, dass *Anweisungen* nach der Klasse bzw. dem Paket und außerhalb des Kontextes der angegebenen Klasse bzw. des angegebenen Pakets ausgeführt werden, so verwendet man

v3.09 für Klassen die Anweisung `\AfterAtEndOfClass` und für Pakete `\AfterAtEndOfPackage`.

Beispiel: Als Beispiel für Paket- oder Klassenautoren will ich zunächst erklären, wie KOMA-Script selbst Gebrauch von den neuen Anweisungen macht. Dazu findet sich in scrbook beispielsweise Folgendes:

```
\AfterPackage{hyperref}{%
  \@ifpackagelater{hyperref}{2001/02/19}{}{%
    \ClassWarningNoLine{scrbook}{%
      You are using an old version of hyperref
      package!\MessageBreak%
      This version has a buggy hack at many
```

```
      drivers\MessageBreak%
      causing \string\addchap\space to behave
      strange.\MessageBreak%
      Please update hyperref to at least version
      6.71b}%
  }%
}
```

Alte Versionen von hyperref definierten ein Makro von scrbook in einer Weise um, so dass es mit neueren Versionen von KOMA-Script nicht mehr funktioniert. Neuere Versionen von hyperref unterlassen dies, wenn sie eine neuere Version von KOMA-Script erkennen. Für den Fall, dass hyperref zu einem späteren Zeitpunkt geladen wird, sorgt also scrbook dafür, dass unmittelbar nach dem Laden des Paketes überprüft wird, ob es sich um eine verträgliche Version handelt. Falls dies nicht der Fall ist, wird eine Warnung ausgegeben.

An anderer Stelle findet sich in drei KOMA-Script-Klassen Folgendes:

```
\AfterPackage{caption2}{%
  \renewcommand*{\setcapindent}{%
```

Nach dem Laden von caption2 und nur falls das Paket geladen wird, wird hier die KOMA-Script eigene Anweisung `\setcapindent` umdefiniert. Der Inhalt der Umdefinierung ist für dieses Beispiel unerheblich. Es sei nur erwähnt, dass caption2 die Kontrolle über die `\caption`-Anweisung übernimmt und daher die normale Definition von `\setcapindent` keinerlei Wirkung mehr hätte. Die Umdefinierung verbessert dann die Zusammenarbeit mit caption2.

Es gibt aber auch Beispiele für den sinnvollen Einsatz der neuen Anweisungen durch normale Anwender. Angenommen, Sie erstellen ein Dokument, aus dem sowohl eine PS-Datei mit LaTeX und dvips als auch eine PDF-Datei mit pdfLaTeX erstellt werden soll. Das Dokument soll außerdem Hyperlinks aufweisen. Im Tabellenverzeichnis haben Sie Einträge, die über mehrere Zeilen gehen. Nun gibt es zwar mit pdfLaTeX bei der PDF-Ausgabe keine Probleme, da dort Links umbrochen werden können. Bei Verwendung des hyperref-Treibers für dvips oder hyperTeX ist dies jedoch nicht möglich. In diesem Fall hätten Sie gerne, dass bei hyperref die Einstellung `linktocpage` verwendet wird. Die Entscheidung, welcher Treiber geladen wird, wird bei Ihnen automatisch von `hyperref.cfg` erledigt. Dazu sieht die Datei beispielsweise wie folgt aus:

```
\ProvidesFile{hyperref.cfg}
\@ifundefined{pdfoutput}{\ExecuteOptions{dvips}}
                        {\ExecuteOptions{pdftex}}
\endinput
```

Alles weitere kann nun `\AfterFile` überlassen werden:

```
\documentclass{article}
\usepackage{scrlfile,ngerman}
\AfterFile{hdvips.def}{\hypersetup{linktocpage}}
\AfterFile{hypertex.def}{\hypersetup{linktocpage}}
\usepackage{hyperref}
\begin{document}
\listoffigures
\clearpage
\begin{figure}
  \caption{Dies ist ein Beispiel mit einer
    Abbildungsunterschrift, die etwas länglich ist
    und bei der trotzdem auf die Verwendung des
    optionalen Arguments verzichtet wurde}
\end{figure}
\end{document}
```

Egal, ob nun der hyperref-Treiber `hypertex` oder `dvips` zu Anwendung kommt, wird die dann nützliche Einstellung `linktocpage` verwendet. Wenn Sie jedoch mit pdfLaTeX eine PDF-Datei erstellen, wird darauf verzichtet, da in diesem Fall der hyperref-Treiber `hpdftex.def` verwendet wird. Das bedeutet, dass weder die Treiberdatei `hdvips.def` noch `hypertex.def` geladen wird.

Übrigens können Sie das Laden von scrlfile und die obigen Anweisungen `\AfterFile` auch in Ihre private `hyperref.cfg` (siehe oben) einfügen. Verwenden Sie dabei jedoch zum Laden des Paketes besser `\RequirePackage` an Stelle von `\usepackage` (siehe [Tea06]). Die neuen Zeilen müssen in obigem Beispiel unmittelbar nach der `\ProvidesFile`-Zeile, also unbedingt vor der Ausführung der Optionen `dvips` oder `pdftex`, eingefügt werden.

`\BeforeClosingMainAux{`*Anweisungen*`}`
`\AfterReadingMainAux{`*Anweisungen*`}`

Diese Anweisungen unterscheidet sich in einem Detail von den zuvor erklärten Anweisungen. Jene ermöglichen Aktionen vor und nach dem Laden von Dateien. Dies ist hier nicht der Fall. Paketautoren haben des Öfteren das Problem, dass Sie Anweisungen in die `aux`-Datei schreiben wollen, nachdem die letzte Seite des Dokuments ausgegeben wurde. Dies wird häufig mit

```
\AtEndDocument{%
  \if@filesw
    \write\@auxout{%
      \protect\writethistoaux%
    }%
  \fi
}
```

versucht. Dies ist jedoch keine echte Lösung des Problems. Wurde die letzte Seite vor `\end{document}` bereits ausgegeben, so führt obiges zu keiner Ausgabe in die `aux`-Datei mehr.

Würde man zur Lösung dieses Problems nun ein `\immediate` vor `\write` setzen, so hätte man das umgekehrte Problem: wurde die letzte Seite bei `\end{document}` noch nicht ausgegeben, so wird `\writethistoaux` zu früh in die `aux`-Datei geschrieben. Man sieht daher häufig Lösungsversuche wie:

```
\AtEndDocument{%
  \if@filesw
    \clearpage
    \immediate\write\@auxout{%
      \protect\writethistoaux%
    }%
  \fi
}
```

Diese Lösung hat jedoch den Nachteil, dass damit die Ausgabe der letzten Seite erzwungen wird. Eine Anweisung wie:

```
\AtEndDocument{%
  \par\vspace*{\fill}%
  Hinweis am Ende des Dokuments.\par
}
```

führt dann nicht mehr dazu, dass der Hinweis am Ende der letzten Seite des Dokuments ausgegeben wird, sie würde stattdessen am Ende der nächsten Seite ausgegeben. Gleichzeitig würde `\writethistoaux` wieder eine Seite zu früh in der `aux`-Datei landen.

Die beste Lösung des Problems wäre nun, wenn man unmittelbar in die `aux`-Datei schreiben könnte, nachdem das finale `\clearpage` innerhalb von `\end{document}` ausgeführt, aber bevor die `aux`-Datei geschlossen wird. Dies ist das Ziel von `\BeforeClosingMainAux`:

```
\BeforeClosingMainAux{%
  \if@filesw
    \immediate\write\@auxout{%
      \protect\writethistoaux%
    }%
  \fi
}
```

Dies ist auch dann erfolgreich, wenn das finale `\clearpage` innerhalb von `\end{document}` tatsächlich zu keiner Ausgabe einer Seite mehr führt oder wenn `\clearpage` innerhalb einer `\AtEndDocument`-Anweisung verwendet wurde.

Es gibt jedoch für `\BeforeClosingMainAux` eine Einschränkung: Im Argument `\Anweisungen` sollte keine Satzanweisung verwendet werden. Es darf also mit `\BeforeClosingMainAux` kein zusätzliches Material gesetzt werden! Wird diese Einschränkung nicht beachtet, so ist das Ergebnis ebenso unvorhersehbarer wie bei den gezeigten Problemen mit `\AtEndDocument`.

v3.03 Die Anweisung `\AfterReadingMainAux` führt sogar *Anweisungen* nach dem Schließen und Einlesen der `aux`-Datei innerhalb von `\end{document}` aus. Dies ist nur in einigen wenigen, sehr seltenen Fällen sinnvoll, beispielsweise, wenn man statistische Informationen in die `log`-Datei schreiben will, die erst nach dem Einlesen der `aux`-Datei gültig sind oder zur Implementierung zusätzlicher *Rerun*-Aufforderungen. Satzanweisungen sind an dieser Stelle noch kritischer zu betrachten als bei `\BeforeClosingMainAux`.

11.3. Dateien beim Einlesen ersetzen

In den bisherigen Abschnitten wurden Anweisungen erklärt, mit denen es möglich ist, vor oder nach dem Einlesen einer bestimmten Datei, eines bestimmten Pakets oder einer Klasse, Aktionen auszuführen. Es ist mit scrlfile aber auch möglich, eine ganz andere Datei als die angeforderte einzulesen.

`\ReplaceInput{`*Dateiname*`}{`*Ersatzdatei*`}`

v2.96 Mit dieser Anweisung wird eine Ersetzung der Datei mit dem als erstes angegebenen *Dateiname* definiert. Wenn LaTeX anschließend angewiesen wird, diese Datei zu laden, wird stattdessen *Ersatzdatei* geladen. Die Definition der Ersatzdatei wirkt sich auf alle Dateien aus, die vom Anwender oder intern von LaTeX mit Hilfe von `\InputIfFileExists` geladen werden. Dazu ist es allerdings erforderlich, dass scrlfile diese Anweisung umdefiniert.

Beispiel: Sie wollen, dass an Stelle der Datei `\jobname.aux`, die Datei `\jobname.xua` geladen wird. Dazu verwenden Sie:

```
\ReplaceInput{\jobname.aux}{\jobname.xua}
```

Wenn Sie nun zusätzlich `\jobname.xua` auch noch durch `\jobname.uxa` ersetzen:

```
\ReplaceInput{\jobname.xua}{\jobname.uxa}
```

dann wird `\jobname.aux` am Ende durch `\jobname.uxa` ersetzt. Es wird also die komplette Ersetzungskette abgearbeitet.

Einer Ersetzung im Kreis:

```
\ReplaceInput{\jobname.aux}{\jobname.xua}
\ReplaceInput{\jobname.xua}{\jobname.aux}
```

würde jedoch zu einem *stack size error* führen. Es ist also nicht möglich, eine einmal ersetzte Datei wieder durch ihren Ursprung zu ersetzen.

Theoretisch wäre es auch möglich, auf diesem Wege ein Paket durch ein anderes oder eine Klasse durch eine andere zu ersetzen. Dabei würde LaTeX aber erkennen, dass die angeforderten Dateinamen nicht zum Namen des Pakets oder der Klasse passen. Eine Lösung dieses Problems finden Sie nachfolgend.

\ReplaceClass{*Klasse*}{*Ersatzklasse*}
\ReplacePackage{*Paket*}{*Ersatzpaket*}

v2.96 Eine Klasse oder ein Paket sollte niemals mit Hilfe der oben erklären Anweisung \ReplaceInput ersetzt werden. In diesem Fall würde LaTeX eine Warnung über nicht übereinstimmende Klassen- oder Paketnamen melden.

Beispiel: Sie ersetzen das Paket fancyhdr leichtfertig durch das nicht kompatible Paket scrpage2, indem Sie

```
\ReplaceInput{fancyhdr.sty}{scrpage2.sty}
```

verwenden. Dies wird beim Laden von fancyhdr zu der Warnung

```
LaTeX warning: You have requested 'scrpage2',
               but the package provides 'fancyhdr'.
```

führen. Für den Anwender wäre diese Warnung mehr als verwirrend, hat er doch gar nicht scrpage2, sondern tatsächlich fancyhdr angefordert, das jedoch durch scrpage2 ersetzt wurde.

Eine Lösung dieses Problems besteht nun darin, statt \ReplaceInput eine der Anweisungen \ReplaceClass oder \ReplacePackage zu verwenden. Es ist zu beachten, dass dabei wie bei \documentclass und \usepackage der Name der Klasse oder des Pakets und nicht deren kompletter Dateiname anzugeben ist.

Die Ersetzung funktioniert für Klassen, die mit \documentclass, \LoadClassWithOptions oder \LoadClass geladen werden. Für Pakete funktioniert die Ersetzung beim Laden mit \usepackage, \RequirePackageWithOptions und \RequirePackage.

Es ist zu beachten, dass die *Ersatzklasse* oder das *Ersatzpaket* mit denselben Optionen geladen wird, mit denen die ursprünglich geforderte Klasse oder das ursprünglich geforderte Paket geladen würden. Wird ein Paket oder eine Klasse durch ein Paket oder eine Klasse ersetzt, das eine geforderte Option nicht unterstützt, würde dies zu den üblichen Warnungen und Fehlern führen. Es ist jedoch möglich, solche in der *Ersatzklasse* oder dem *Ersatzpaket* fehlenden Optionen per \BeforeClass oder \BeforePackage neu zu definieren.

Beispiel: Angenommen, das Paket oldfoo soll beim Laden durch das Paketnewfoo ersetzt werden. Dies wird mit:

```
\ReplacePackage{oldfoo}{newfoo}
```

erreicht. Das alte Paket hat eine Option oldopt, die das neue Paket jedoch nicht hat. Mit

```
\BeforePackage{newfoo}{%
  \DeclareOption{oldopt}{%
    \PackageInfo{newfoo}%
                {option 'oldopt' not supported}%
  }}%
```

wird diese Option nun für das Paket newfoo nachdefiniert. Dadurch wird vermieden, dass beim Laden des Pakets oldfoo ein Fehler über die im Paket newfoo nicht unterstützte Option gemeldet wird.

Existiert hingegen eine Option `newopt`, die an Stelle der Option `oldopt` verwendet werden soll, so kann dies ebenfalls erreicht werden:

```
\BeforePackage{newfoo}{%
  \DeclareOption{oldopt}{%
    \ExecuteOptions{newopt}%
  }}%
```

Es ist sogar möglich, festzulegen, dass beim Laden des neuen Pakets andere Voreinstellung gelten sollen:

```
\BeforePackage{newfoo}{%
  \DeclareOption{oldopt}{%
    \ExecuteOptions{newopt}%
  }%
  \PassOptionsToPackage{newdefoptA,newdefoptB}%
                       {newfoo}%
}
```

oder auch direkt:

```
\BeforePackage{newfoo}{%
  \DeclareOption{oldopt}{%
    \ExecuteOptions{newopt}%
  }%
}%
\PassOptionsToPackage{newdefoptA,newdefoptB}%
                     {newfoo}%
```

Damit Klassen ersetzt werden können, ist es erforderlich scrlfile mit Hilfe von `\RequirePackage` vor der Klasse zu laden.

11.4. Dateien gar nicht erst einlesen

v3.08 Gerade in Klassen und Paketen, die innerhalb von Firmen oder Instituten verwendet werden, findet man häufig, dass sehr viele Pakete nur deshalb geladen werden, weil die Anwender diese Pakete häufig verwenden. Wenn es dann mit einem dieser automatisch geladenen Paketen zu einem Problem kommt, muss man irgendwie das Laden des problematischen Paketes verhindern. Auch hier bietet scrlfile eine einfache Lösung.

`\PreventPackageFromLoading{`*Paketliste*`}`

v3.08 Wird diese Anweisung vor dem Laden eines Paket mit `\usepackage`, `\RequirePackage` oder `\RequirePackageWithOptions` aufgerufen, so wird das Laden des Pakets effektiv verhindert,

falls es in der *Paketliste* zu finden ist.

Beispiel: Angenommen Sie arbeiten in einer Firma, in der alle Dokumente mit Latin Modern erzeugt werden. In der Firmenklasse, firmenci, befinden sich daher die Zeilen:

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

Nun wollen Sie zum ersten Mal ein Dokument mit XeLaTeX oder LuaLaTeX setzen. Da hierbei ohnehin Latin-Modern voreingestellt ist und das Laden von fontenc eher störend wäre, wollen Sie das Laden beider Pakete verhindern. Sie laden die Klasse deshalb nun in Ihrem eigenen Dokument wie folgt:

```
\RequirePackage{scrlfile}
\PreventPackageFromLoading{fontenc,lmodern}
\documentclass{firmenci}
```

Wie im Beispiel zu sehen ist, kann man das Paket scrlfile auch bereits vor der Klasse laden. In diesem Fall muss das Laden dann aber mit Hilfe von \RequirePackage erfolgen, da \usepackage vor \documentclass verboten ist.

```
\StorePreventPackageFromLoading{\Anweisung}
\ResetPreventPackageFromLoading
```

v3.08 \Anweisung wird mit \StorePreventPackageFromLoading als die aktuelle Liste der Pakete definiert, für die das Laden verhindert werden soll. Dagegen setzt v3.08 \ResetPreventPackageFromLoading die Liste der Pakete, für die das Laden verhindert werden soll, zurück. Danach können wieder alle Pakete geladen werden.

Beispiel: Angenommen, Sie sind innerhalb eines Pakets unbedingt auf das Laden eines anderen Pakets angewiesen und wollen nicht, dass der Anwender das Laden dieses Pakets mit \PreventPackageFromLoading verhindern kann. Also setzen Sie die Paketliste für die Ausnahmen zuvor zurück:

```
\ResetPreventPackageFromLoading
\RequirePackage{foo}
```

Allerdings hat dies den Nachteil, dass ab diesem Zeitpunkt die komplette Ausnahmeliste des Anwenders verloren ist. Also speichern Sie die Liste zunächst zwischen und reaktivieren Sie später wieder:

```
\newcommand*{\Users@PreventList}{}%
\StorePreventPackageFromLoading\Users@PreventList
\ResetPreventPackageFromLoading
\RequirePackage{foo}
\PreventPackageFromLoading{\Users@PreventList}
```

Es ist zu beachten, dass `\Users@PreventList` durch die Anweisung `\StorePreventPackageFromLoading` auch definiert werden würde, wenn diese bereits anderweitig definiert wäre. Eine vorhandene Definition würde also ohne Rücksicht überschrieben werden. In diesem Beispiel wurde deshalb mit einem vorherigen `\newcommand*` sichergestellt, dass in diesem Fall zur Sicherheit eine Fehlermeldung ausgegeben wird.

An dieser Stelle sei darauf hingewiesen, dass Sie bei Manipulationen an der mit `\StorePreventPackageFromLoading` zwischengespeicherten Liste selbst die Verantwortung für eine korrekte Wiederherstellbarkeit tragen. So muss die Liste unbedingt mit Komma separiert sein, sollte keine Leerzeichen oder Gruppenklammern enthalten und muss voll expandierbar sein.

Dateien mit scrwfile sparen und ersetzen

Eines der Probleme, die auch durch die Einführung von ε-TEX nicht gelöst wurde, ist die Tatsache, dass TEX nur 18 Dateien gleichzeitig zum Schreiben geöffnet haben kann. Diese Zahl erscheint zunächst recht groß. Allerdings ist zu berücksichtigen, dass bereits LATEX selbst einige dieser Dateien belegt. Inhaltsverzeichnis, Tabellenverzeichnis, Abbildungsverzeichnis, Index, Glossar und jedes weitere Verzeichnis, das von LATEX aus erzeugt wird, belegt in der Regel eine weitere Datei. Dazu kommen Hilfsdateien von Paketen wie hyperref oder minitoc.

Im Endeffekt kann es daher geschehen, dass irgendwann die Meldung

```
! No room for a new \write .
\ch@ck ...\else \errmessage {No room for a new #3}
                                                \fi
```

erscheint.

Dass LATEX bei Verzeichnissn wie dem Inhaltsverzeichnis, dem Tabellenverzeichnis und dem Abbildungsverzeichnis immer sofort eine neue Datei zum Schreiben öffnet, hat aber auch noch einen weiteren Nachteil. Solche Verzeichnisse werden durch deren Befehle nicht nur direkt gesetzt, sie können auch kein weiteres Mal gesetzt werden, da die zugehörige Hilfsdatei nach dem jeweiligen Befehl bis zum Ende des Dokuments leer ist.

Das Paket scrwfile bietet hier eine grundsätzliche Änderung im LATEX-Kern, durch die beide Probleme gelöst werden können.

12.1. Grundsätzliche Änderungen am LATEX-Kern

LATEX-Klassen verwenden zum Öffnen eines Verzeichnisses beispielsweise mit `\tableofcontents` oder `\listoffigure` die LATEX-Kern-Anweisung `\@starttoc`. LATEX selbst lädt bei dieser Anweisung nicht nur die zugehörige Hilfsdatei, sondern öffnet diese Hilfsdatei auch neu zum Schreiben. Werden anschließend mit `\addtocontents` oder `\addcontentsline` Einträge in dieses Verzeichnis vorgenommen, so wird jedoch nicht direkt in die geöffnete Hilfsdatei geschrieben. Stattdessen schreibt LATEX `\@writefile`-Anweisungen in die `aux`-Datei. Erst beim Einlesen der `aux`-Dateien am Ende des Dokuments wird dann über diese `\@writefile`-Anweisungen in die tatsächlichen Hilfsdateien geschrieben. Die Hilfsdateien werden von LATEX auch nicht explizit geschlossen. Stattdessen verlässt sich LATEX hier darauf, dass TEX die Dateien am Ende ohnehin schließt.

Dieses Vorgehen sorgt dafür, dass die Hilfsdateien zwar erst innerhalb von `\end{document}` tatsächlich beschrieben werden, aber trotzdem während des gesamten LATEX-Laufs gleichzeitig offen sind. scrwfile hat nun genau hier einen Ansatzpunkt: die Umdefinierung von `\@starttoc` und `\@writefile`.

Natürlich besitzen Änderungen am LATEX-Kern immer das Potential, dass es zu Unverträg-

lichkeiten mit anderen Paketen kommen kann. Betroffen können in erster Linie Pakete sein, die ebenfalls `\@starttoc` oder `\@writefile` umdefinieren. In einigen Fällen kann es helfen, die Reihenfolge der Pakete zu ändern.

Tatsächlich sind bisher keine solchen Probleme im Zusammenhang mit scrwfile bekannt, obwohl das Paket vor der Veröffentlichung von verschiedenen Anwendern über ein Jahr lang getestet wurde. Wenn Sie jedoch auf ein solches Problem stoßen, sollten Sie sich an den KOMA-Script-Autor wenden.

12.2. Das Eindateiensystem

Bereits beim Laden des Pakets mit

```
\usepackage{scrwfile}
```

wird `\@starttoc` von scrwfile so umdefiniert, dass davon selbst keine Datei mehr zum Schreiben angefordert und geöffnet wird. Unmittelbar vor dem Schließen der `aux`-Datei in `\end{document}` wird dann `\@writefile` so umdefiniert, dass diese Anweisung statt in die eigentlichen Hilfsdateien in eine neue Hilfsdatei mit der Endung `wrt` schreibt. Nach dem Einlesen der `aux`-Dateien wird dann noch die `wrt`-Datei abgearbeitet und zwar ein Mal für jede der Hilfsdateien, in die mit `\@writefile` geschrieben wird. Dabei muss dann aber nicht jede dieser Hilfsdateien gleichzeitig geöffnet sein. Stattdessen ist immer nur eine zum Schreiben geöffnet und wird auch wieder explizit geschlossen. Da dabei eine interne Schreibdatei von LaTeX wiederverwendet wird, benötigt scrwfile keine einzige eigene Schreibdatei für diese Art von Verzeichnissen.

Selbst, wenn bisher nur mit einem Inhaltsverzeichnis gearbeitet wird, steht nach dem Laden des Pakets bereits eine Schreibdatei mehr für Literaturverzeichnisse, Stichwortverzeichnisse, Glossare und ähnliche Verzeichnisse, die nicht mit `\@starttoc` arbeiten, zur Verfügung. Darüber hinaus können beliebig viele Verzeichnisse, die mit `\@starttoc` arbeiten, angelegt werden.

12.3. Das Klonen von Dateieinträgen

Nachdem `\@writefile` für das Eindateiensystem aus dem vorherigen Abschnitt bereits so geändert wurde, dass es nicht direkt in die entsprechende Hilfsdatei schreibt, lag eine weitere Idee nahe. Beim Kopieren der `\@writefile`-Anweisungen in die `wrt`-Datei können diese auch für andere Zielendungen übernommen werden.

```
\TOCclone[Verzeichnisüberschrift]{Quellendung}{Zielendung}
\listofZielendung
```

Durch dieses Klonen von Dateieinträgen werden so ganze Verzeichnisse geklont. Dazu muss man nur die Endung der Hilfsdatei des Verzeichnisses kennen, dessen Einträge kopiert werden sollen. Zusätzlich muss man die Endung einer Zieldatei angeben. In diese werden die Einträge

dann kopiert. Natürlich kann man in dieses geklonte Verzeichnis auch zusätzliche Einträge schreiben.

Die *Zielendung* der Zieldatei wird mit Hilfe von tocbasic (siehe Kapitel 13) verwaltet. Steht eine solche Datei bereits unter Kontrolle von tocbasic wird eine Warnung ausgegeben. Anderenfalls wird mit Hilfe von tocbasic ein neues Verzeichnis für diese Endung angelegt. Die Überschrift dieses Verzeichnisses kann man über das optionale Argument *Verzeichnisüberschrift* bestimmen.

Ausgeben kann man dieses neue Verzeichnis dann beispielsweise über die Anweisung `\listof`*Zielendung*. Die Verzeichniseigenschaften `leveldown`, `numbered`, `onecolumn` und `totoc` (siehe Anweisung `\setuptoc` in Abschnitt 13.2, Seite 292) werden automatisch in das Zielverzeichnis übernommen, falls sie für das Quellverzeichnis bereits gesetzt waren. Die Eigenschaft `nobabel` wird für geklonte Verzeichnisse immer gesetzt, da die entsprechenden babel-Einträge in das Quellverzeichnis ohnehin bereits kopiert werden.

Beispiel: Angenommen, Sie wollen eine Gliederungsübersicht, in der nur die Kapitel angezeigt werden, zusätzlich zum normalen Inhaltsverzeichnis.

```
\usepackage{scrwfile}
\TOCclone[Gliederungsübersicht]{toc}{stoc}
```

Hierdurch wird zunächst ein neues Verzeichnis mit der Überschrift »Gliederungsübersicht« angelegt. Das neue Verzeichnis verwendet die Dateiendung `stoc`. Alle Einträge in die Datei mit der Endung `toc` werden auch in dieses Verzeichnis kopiert.

Damit dieses neue Verzeichnis nun nur die Kapitelebene ausgibt, verwenden wir:

```
\addtocontents{stoc}{\protect\value{tocdepth}=0}
```

Während normalerweise erst ab `\begin{document}` Einträge in ein Verzeichnis vorgenommen werden können, funktioniert dies nach Laden von scrwfile bereits in der Dokumentpräambel. Durch die hier gezeigte unkonventionelle Art, den Zähler `tocdepth` innerhalb der Verzeichnisdatei zu ändern, bleibt diese Änderung nur für dieses Verzeichnis wirksam.

Später im Dokument wird das Verzeichnis der Endung `stoc` dann mit

```
\listofstoc
```

ausgegeben und zeigt nur die Teile und Kapitel des Dokuments.

Etwas schwieriger wird es, wenn das Inhaltsverzeichnis in der Gliederungsübersicht angezeigt werden soll. Dies wäre zwar mit

```
\addtocontents{toc}{%
  \protect\addcontentsline
    {stoc}{chapter}{\protect\contentsname}%
}
```

möglich. Da jedoch alle Einträge in `toc` auch nach `stoc` kopiert werden, würde so von der Gliederungsübersicht dieser Eintrag ebenfalls übernommen. Also darf der Eintrag nicht aus der Verzeichnisdatei heraus erzeugt werden. Da das Paket tocbasic zum Einsatz kommt, kann aber

```
\AfterStartingTOC[toc]{%
  \addcontentsline{stoc}{chapter}
                  {\protect\contentsname}}
```

verwendet werden. Natürlich setzt dies voraus, dass die Datei mit Endung `toc` auch unter der Kontrolle von tocbasic steht. Dies ist bei allen KOMA-Script-Klassen der Fall. Näheres zur Anweisung `\AfterStartingTOC` ist in Abschnitt 13.2 auf Seite 291 zu finden.

12.4. Hinweis zum Entwicklungsstand

Obwohl das Paket bereits von mehreren Anwendern getestet wurde und vielfach im Einsatz ist, ist seine Entwicklung noch nicht abgeschlossen. Deshalb ist es theoretisch möglich, dass insbesondere an der internen Funktionsweise des Pakets noch Änderungen vorgenommen werden. Sehr wahrscheinlich sind auch künftige Erweiterungen. Teilweise befindet sich bereits Code für solche Erweiterungen im Paket. Da jedoch noch keine Benutzeranweisungen existieren, mit denen diese Möglichkeiten genutzt werden könnten, wurde hier auf eine Dokumentation derselben verzichtet.

Verzeichnisse verwalten mit Hilfe von tocbasic

Der Hauptzweck des Pakets tocbasic besteht darin, Paket- und Klassenautoren die Möglichkeit zu geben, eigene Verzeichnisse vergleichbar mit dem Abbildungs- und Tabellenverzeichnis zu erstellen und dabei Klassen und anderen Paketen einen Teil der Kontrolle über diese Verzeichnisse zu erlauben. Dabei sorgt das Paket tocbasic auch dafür, dass diese Verzeichnisse von babel (siehe [Bra01]) bei der Sprachumschaltung mit berücksichtigt werden. Durch Verwendung von tocbasic soll dem Paketautor die Mühe genommen werden, selbst solche Anpassungen an andere Pakete oder an Klassen vornehmen zu müssen.

KOMA-Script verwendet tocbasic sowohl für das Inhaltsverzeichnis als auch für die bereits erwähnten Verzeichnisse für Abbildungen und Tabellen.

13.1. Grundlegende Anweisungen

Die grundlegenden Anweisungen dienen in erster Linie dazu, eine Liste aller bekannten Dateierweiterungen, die für Verzeichnisse stehen, zu verwalten. Einträge in Dateien mit solchen Dateierweiterungen werden typischerweise mit `\addtocontents` oder `\addcontentsline` vorgenommen. Darüber hinaus gibt es Anweisungen, mit denen eine Aktion für all diese Dateierweiterungen durchgeführt werden können. Außerdem gibt es Anweisungen, um Einstellungen für die Datei vorzunehmen, die zu einer gegebenen Dateierweiterung gehört. Typischerweise hat so eine Dateierweiterung auch einen Besitzer. Dieser Besitzer kann eine Klasse oder ein Paket oder ein Bezeichner sein, den der Autor der Klasse oder des Pakets, das tocbasic verwendet, selbst gewählt hat. KOMA-Script selbst verwendet beispielsweise den Besitzer `float` für die Dateierweiterungen `lof` und `lot`, die für das Abbildungs- und das Tabellenverzeichnis stehen. Für das Inhaltsverzeichnis verwendet KOMA-Script als Besitzer den Dateiname der Klasse.

`\ifattoclist{`*`Dateierweiterung`*`}{`*`Dann-Teil`*`}{`*`Sonst-Teil`*`}`

Mit dieser Anweisung wird überprüft, ob die *`Dateierweiterung`* bereits in der Liste der bekannten Dateierweiterungen vorhanden ist, oder nicht. Ist die *`Dateierweiterung`* bereits über diese Liste bekannt, so wird der *`Dann-Teil`* ausgeführt. Anderenfalls wird der *`Sonst-Teil`* ausgeführt.

Beispiel: Angenommen, Sie wollen wissen, ob die Dateierweiterung »`foo`« bereits verwendet wird, um in diesem Fall einen Fehlermeldung auszugeben, weil diese damit nicht mehr verwendet werden kann:

```
\ifattoclist{foo}{%
  \PackageError{bar}{%
    extension ‘foo’ already in use%
```

```
  }{%
    Each extension may be used only
    once.\MessageBreak
    The class or another package already
    uses extension ‘foo’.\MessageBreak
    This error is fatal!\MessageBreak
    You should not continue!}%
}{%
  \PackageInfo{bar}{using extension ‘foo’}%
}
```

`\addtotoclist[`*Besitzer*`]{`*Dateierweiterung*`}`

Diese Anweisung fügt die *Dateierweiterung* zu der Liste der bekannten Dateierweiterungen. Ist die *Dateierweiterung* bereits bekannt, so wird hingegen ein Fehler gemeldet, um die doppelte Verwendung derselben *Dateierweiterung* zu verhindern.

Wenn das optionale Argument [*Besitzer*] angegeben wurde, wird der angegebene *Besitzer* für diese Dateierweiterung mit gespeichert. Wurde das optionale Argument weg gelassen, dann versucht tocbasic den Dateinamen der aktuell abgearbeiteten Klasse oder des Pakets herauszufinden und als *Besitzer* zu speichern. Dies funktioniert nur, wenn `\addtotoclist` während des Ladens der Klasse oder des Pakets aufgerufen wird. Es funktioniert nicht, wenn `\addtoclist` erst später Aufgrund der Verwendung einer Anweisung durch den Benutzer aufgerufen wird. In diesem Fall wird als *Besitzer* ».« eingetragen.

Beachten Sie, dass ein leeres Argument *Besitzer* nicht das gleiche ist, wie das Weglassen des kompletten optionalen Arguments einschließlich der eckigen Klammern. Ein leeres Argument würde auch einen leeren *Besitzer* ergeben.

Beispiel: Angenommen, Sie wollen die Dateierweiterung »`foo`« zu der Liste der bekannten Dateierweiterungen hinzufügen, während Ihr Paket mit dem Dateinamen »`bar.sty`« geladen wird:

```
\addtotoclist{foo}
```

Dies fügt die Dateierweiterung »`foo`« mit dem Besitzer »`bar.sty`« zur Liste der bekannten Dateierweiterung, wenn diese Erweiterung nicht bereits in der Liste ist. Wenn bereits die verwendete Klasse oder ein anderes Paket diese Dateierweiterung angemeldet hat, erhalten Sie den Fehler:

```
Package tocbasic Error: file extension ‘foo’ cannot be used twice


See the tocbasic package documentation for explanation.
Type H <return> for immediate help.
```

Wenn Sie dann tatsächlich die Taste »`H`« gefolgt von der Return-Taste drücken, erhalten Sie als Hilfe:

```
File extension ‘foo’ is already used by a toc-file, while bar.sty
tried to use it again for a toc-file.
This may be either an incompatibility of packages, an error at a ↩
  package,
or a mistake by the user.
```

Vielleicht stellt Ihr Paket auch eine Anweisung bereit, die ein Verzeichnis dynamisch erzeugt. In diesem Fall sollten Sie das optionale Argument von `\addtotoclist` verwenden, um den *Besitzer* anzugeben:

```
\newcommand*{\createnewlistofsomething}[1]{%
  \addtotoclist[bar.sty]{#1}%
  % Weitere Aktionen, um dieses Verzeichnis
  % verfügbar zu machen
}
```

Wenn jetzt der Anwender diese Anweisung aufruft, beispielsweise mit

```
\createnewlistofsomething{foo}
```

dann wird die Dateierweiterung »`foo`« ebenfalls mit dem Besitzer »`bar.sty`« zur Liste der bekannten Dateierweiterungen hinzugefügt oder aber ein Fehler gemeldet, wenn diese Dateierweiterung bereits verwendet wird.

Sie können als *Besitzer* angeben, was immer Sie wollen, aber es sollte eindeutig sein! Wenn Sie beispielsweise der Autor des Pakets float wären, könnten Sie als *Besitzer* auch »`float`« an Stelle von »`float.sty`« angeben. In diesem Fall würden die KOMA-Script-Optionen für das Verzeichnis der Abbildungen und das Verzeichnis der Tabellen auch Ihre Verzeichnisse betreffen, die zum Zeitpunkt der Verwendung der Option bereits zur Liste der bekannten Dateierweiterungen hinzugefügt sind. Das liegt daran, dass KOMA-Script die Dateierweiterungen »`lof`« für das Abbildungsverzeichnis und »`lot`« für das Tabellenverzeichnis mit dem Besitzer »*`float`*« anmeldet und die Optionen für diese Besitzer setzt.

`\AtAddToTocList[`*`Besitzer`*`]{`*`Anweisungen`*`}`

Auf diese Weise können die *`{Anweisungen}`* zu einer internen Liste von Anweisungen hinzugefügt werden, die immer dann auszuführen sind, wenn eine Dateierweiterung mit dem angegebenen *`Besitzer`* zur Liste der bekannten Dateierweiterungen hinzugefügt wird. Bezüglich des optionalen Argument wird wie in der Erklärung von `\addtotoclist` beschrieben verfahren. Wird das optionale Argument leer gelassen, werden in diesem Fall die Aktionen unabhängig vom Besitzer immer ausgeführt, wenn die Dateierweiterung zu der Liste der bekannten Dateierweiterungen hinzugefügt wird. Während der Ausführung der *`Anweisungen`* ist außerdem `\@currext` die Dateierweiterung, die gerade hinzugefügt wird.

Beispiel: tocbasic selbst verwendet

```
\AtAddToTocList[]{%
  \expandafter\tocbasic@extend@babel
  \expandafter{\@currext}}
```

um jede Dateierweiterung zu der in tocbasic vorhandenen Erweiterung für das Paket babel hinzuzufügen.

Die zweimalige Verwendung von `\expandafter` ist im Beispiel erforderlich, weil das Argument von `\tocbasic@extend@babel` zwingend bereits expandiert sein muss. Siehe dazu auch die Erklärung zu `\tocbasic@extend@babel` in Abschnitt 13.3, Seite 294.

`\removefromtoclist[`*Besitzer*`]{`*Dateierweiterung*`}`

Man kann eine *Dateierweiterung* auch wieder aus der Liste der bekannten Dateierweiterungen entfernen. Ist das optionale Argument `[`*Besitzer*`]` angegeben, so wird die Dateierweiterung nur entfernt, wenn sie für den angegebenen *Besitzer* angemeldet wurde. Wie der Besitzer beim Weglassen des optionalen Argument bestimmt wird, ist der Erklärung zu `\addtotoclist` zu entnehmen. Wird ein leerer *Besitzer* angegeben, findet kein Besitzertest statt, sondern die *Dateierweiterung* wird unabhängig vom Besitzer entfernt.

`\doforeachtocfile[`*Besitzer*`]{`*Anweisungen*`}`

Bisher haben Sie nur Anweisungen kennen gelernt, die für Klassen- und Paketautoren zwar zusätzliche Sicherheit, aber auch eher zusätzlichen Aufwand bedeuten. Mit `\doforeachtocfile` kann man die erste Ernte dafür einfahren. Diese Anweisung erlaubt es die angegebenen *Anweisungen* für jede von dem *Besitzer* angemeldete Dateierweiterung auszuführen. Während der Ausführung der *Anweisungen* ist `\@currext` die aktuell verarbeitete Dateierweiterung. Wird das optionale Argument `[`*Besitzer*`]` weggelassen, so werden alle Dateierweiterungen unabhängig vom Besitzer abgearbeitet. Ein leeres optionales Argument würde hingegen nur die Dateierweiterungen mit leerem Besitzer verarbeiten.

Beispiel: Wenn Sie die Liste aller bekannten Dateierweiterungen auf das Terminal und in die `log`-Datei ausgeben wollen, ist dies einfach mit

```
\doforeachtocfile{\typeout{\@currext}}
```

möglich. Wollen Sie hingegen nur die Dateierweiterungen des Besitzer »`foo`«, dann geht das einfach mit:

```
\doforeachtocfile[foo]{\typeout{\@currext}}
```

Die KOMA-Script-Klassen scrbook und scrreprt verwenden diese Anweisung übrigens, um für jedes Verzeichnis, für das die Eigenschaft `chapteratlist` gesetzt ist, optional einen vertikalen Abstand oder die Kapitelüberschrift in dieses Verzeichnis einzutragen. Wie Sie diese Eigenschaft setzen können, ist in Abschnitt 13.2, Seite 292 zu finden.

`\tocbasicautomode`

Diese Anweisung definiert die Anweisung `\@starttoc`, die der LaTeX-Kern bereit stellt, so um, dass bei jedem Aufruf von `\@starttoc` die dabei angegebene Dateierweiterung in die Liste der bekannten Dateierweiterungen eingefügt wird, soweit sie dort noch nicht vorhanden ist. Außerdem wird dann `\tocbasic@starttoc` an Stelle von `\@starttoc` verwendet. Näheres zu `\tocbasic@starttoc` und `\@starttoc` ist Abschnitt 13.3, Seite 295 zu entnehmen.

Mit Hilfe von `\tocbasicautomode` wird also sozusagen jedes Verzeichnis, das mit Hilfe von `\@starttoc` erstellt wird, automatisch zumindest teilweise unter die Kontrolle von tocbasic gestellt. Ob das zum gewünschten Ergebnis führt, hängt jedoch sehr von den jeweiligen Verzeichnissen ab. Immerhin funktioniert damit schon einmal die Erweiterung für das babel-Paket für alle Verzeichnisse. Es ist jedoch vorzuziehen, wenn der Paketautor selbst tocbasic explizit verwendet. Er kann dann auch die weiteren Vorteile nutzen, die ihm das Paket bietet und die in den nachfolgenden Abschnitten beschrieben werden.

13.2. Erzeugen eines Verzeichnisses

Im vorherigen Abschnitt haben Sie erfahren, wie eine Liste bekannter Dateierweiterungen geführt werden kann und wie automatisch Anweisungen beim Hinzufügen von Dateierweiterungen zu dieser Liste ausgeführt werden können. Des Weiteren haben Sie eine Anweisung kennen gelernt, mit der man für jede einzelne bekannte Dateierweiterung oder einen spezifischen Teil davon Anweisungen ausführen kann. In diesem Abschnitt werden Sie Anweisungen kennen lernen, die sich auf die Datei beziehen, die mit dieser Dateierweiterung verbunden ist.

`\addtoeachtocfile[`*`Besitzer`*`]{`*`Inhalt`*`}`

Die Anweisung `\addtoeachtocfile` schreibt *`Inhalt`* mit Hilfe von `\addtocontents` aus dem LaTeX-Kern in jede Datei, die für den angegebenen *`Besitzer`* in der Liste der bekannten Dateierweiterungen zu finden ist. Wird das optionale Argument weggelassen, wird in jede Datei aus der Liste der bekannten Dateierweiterungen geschrieben. Der konkrete Dateiname setzt sich dabei übrigens aus `\jobname` und der Dateierweiterung zusammen. Während des Schreibens von *`Inhalt`* ist `\@currext` die Dateierweiterung der Datei, in die aktuell geschrieben wird.

Beispiel: Sie wollen einen vertikalen Abstand von einer Zeile in alle Dateien aus der Liste der bekannten Dateierweiterungen schreiben.

```
\addtoeachtocfile{%
  \protect\addvspace{\protect\baselineskip}%
}%
```

Wenn Sie das hingegen nur für die Dateien mit dem definierten Besitzer »`foo`« machen wollen, verwenden Sie:

```
\addtoeachtocfile[foo]{%
  \protect\addvspace{\protect\baselineskip}%
}
```

Anweisungen, die nicht bereits beim Schreiben expandiert werden sollen, sollte wie bei `\addtocontents` mit `\protect` geschützt werden.

`\addcontentslinetoeachtocfile[`*Besitzer*`]{`*Ebene*`}{`*Inhalt*`}`

Diese Anweisung ist mit `\addcontentsline` aus dem LaTeX-Kern vergleichbar. Der Unterschied besteht darin, dass diese Anweisung *Inhalt* nicht nur in eine einzelne Datei, sondern in alle Dateien eines angegebenen *Besitzers* und bei Verzicht auf das optionale Argument in alle Dateien aus der Liste der bekannten Dateierweiterungen schreibt.

Beispiel: Angenommen, Sie sind Klassen-Autor und wollen den Kapiteleintrag nicht nur in das Inhaltsverzeichnis, sondern in alle Verzeichnisdateien schreiben. Nehmen wir weiter an, dass aktuell *#1* den Titel enthält, der geschrieben werden soll.

```
\addcontentslinetoeachtocfile{chapter}{%
  \protect\numberline{\thechapter}#1%
}%
```

In diesem Fall soll natürlich die aktuelle Kapitelnummer direkt beim Schreiben in die Verzeichnisdatei expandiert werden, weshalb sie nicht mit `\protect` vor der Expansion geschützt wurde.

Während des Schreibens von *Inhalt* ist auch hier wie schon bei `\addtoeachtocfile` `\@currext` die Dateierweiterung der Datei, in die aktuell geschrieben wird.

`\listoftoc[`*Titel*`]{`*Dateierweiterung*`}`
`\listoftoc*{`*Dateierweiterung*`}`
`\listofeachtoc[`*Besitzer*`]`
`\listof`*Dateierweiterung*`name`

Mit diesen Anweisungen werden die Verzeichnisse ausgegeben. Die Sternvariante `\listoftoc*` benötigt als einziges Argument die *Dateierweiterung* der Datei mit den Daten zu dem Verzeichnis. Die Anweisung setzt zunächst die vertikalen und horizontalen Abstände, die innerhalb von Verzeichnissen gelten sollen, führt die Anweisungen aus, die vor dem Einlesen der Datei ausgeführt werden sollen, liest dann die Datei und führt zum Schluss die Anweisungen aus, die nach dem Einlesen der Datei ausgeführt werden sollen. Damit kann `\listoftoc*` als direkter Ersatz der LaTeX-Kern-Anweisung `\@starttoc` verstanden werden.

Die Version von `\listoftoc` ohne Stern setzt das komplette Verzeichnis und veranlasst auch einen optionalen Eintrag in das Inhaltsverzeichnis und den Kolumnentitel. Ist das optionale Argument `[`*Titel*`]` gegeben, so wird diese Angabe sowohl als Überschrift als auch als optionaler Eintrag in Inhaltsverzeichnis und Kolumnentitel verwendet. Ist das Argument

Titel lediglich leer, so wird auch eine leere Angabe verwendet. Wird hingegen das komplette Argument einschließlich der eckigen Klammern weggelassen, so wird die Anweisung `\listof`*Dateierweiterung*`name` verwendet, wenn diese definiert ist. Ist sie nicht definiert, wird ein Standard-Ersatzname verwendet und eine Warnung ausgegeben.

Die Anweisung `\listofeachtoc` gibt alle Verzeichnisse des angegebenen Besitzers oder alle Verzeichnisse aller bekannten Dateinamenerweiterungen aus. Dabei sollte `\listof`*Dateierweiterung*`name` definiert sein, damit der korrekte Titel ausgegeben werden kann.

Da eventuell auch der Anwender selbst `\listoftoc` ohne optionales Argument oder `\listofeachtoc` verwenden könnte, wird empfohlen `\listof`*Dateierweiterung*`name` immer passend zu definieren.

Beispiel: Angenommen, Sie haben ein neues »Verzeichnis der Algorithmen« mit der Dateierweiterung »`loa`« und wollen dieses anzeigen lassen:

```
\listoftoc[Verzeichnis der Algorithmen]{loa}
```

erledigt das für Sie. Wollen Sie das Verzeichnis hingegen ohne Überschrift ausgegeben haben, dann genügt:

```
\listoftoc*{loa}
```

In zweiten Fall würde natürlich auch ein optional aktivierter Eintrag in das Inhaltsverzeichnis nicht gesetzt. Näheres zur Eigenschaft des Eintrags in das Inhaltsverzeichnis ist bei der Anweisung `\setuptoc`, Seite 292 zu finden.

Wenn Sie zuvor

```
\newcommand*{\listofloaname}{%
  Verzeichnis der Algorithmen%
}
```

definiert haben, genügt auch

```
\listoftoc{loa}
```

um ein Verzeichnis mit der gewünschten Überschrift zu erzeugen. Für den Anwender ist es eventuell einprägsamer, wenn Sie dann außerdem noch

```
\newcommand*{\listofalgorithms}{\listoftoc{loa}}
```

als einfache Verzeichnisanweisung definieren.

Da LaTeX bei der Ausgabe eines Verzeichnisses auch gleich eine neue Verzeichnisdatei zum Schreiben öffnet, kann der Aufruf jeder dieser Anweisungen zu einer Fehlermeldung der Art

```
! No room for a new \write .
\ch@ck ...\else \errmessage {No room for a new #3}
                                                \fi
```

führen, wenn keine Schreibdateien mehr zur Verfügung stehen. Abhilfe kann in diesem Fall das Lades des in Kapitel 12 beschriebenen Pakets scrwfile bieten.

```
\BeforeStartingTOC[Dateierweiterung]{Anweisungen}
\AfterStartingTOC[Dateierweiterung]{Anweisungen}
```

Manchmal ist es nützlich, wenn unmittelbar vor dem Einlesen der Datei mit den Verzeichnisdaten *Anweisungen* ausgeführt werden können. Mit Hilfe von `\BeforeStartingTOC` können Sie genau solche *Anweisungen* wahlweise für eine einzelne *Dateierweiterung* oder alle Dateien, die mit Hilfe von `\listoftoc*`, `\listoftoc` oder `\listofeachtoc` eingelesen werden, erreichen. Ebenso können Sie *Anweisungen* nach dem Einlesen der Datei ausführen, wenn Sie diese mit `\AfterStartingTOC` definieren. Während der Ausführung der *Anweisungen* ist `\@currext` die Dateierweiterung der Datei, die eingelesen wird bzw. gerade eingelesen wurde.

Ein Beispiel zur Verwendung von `\AfterStartingTOC` ist in Abschnitt 12.3 auf Seite 283 zu finden.

```
\BeforeTOCHead[Dateierweiterung]{Anweisungen}
\AfterTOCHead[Dateierweiterung]{Anweisungen}
```

Es können auch *Anweisungen* definiert werden, die unmittelbar vor oder nach dem Setzen der Überschrift bei Verwendung von `\listoftoc` oder `\listofeachtoc`, ausgeführt werden. Bezüglich des optionalen Arguments und der Bedeutung von `\@currext` gilt, was bereits bei `\BeforeStartingTOC` und `\AfterStartingTOC` oben erklärt wurde.

```
\MakeMarkcase
```

Wann immer tocbasic eine Marke für einen Kolumnentitel setzt, dann erfolgt dies als Argument der Anweisung `\MakeMarkcase`. Diese Anweisung ist dazu gedacht, bei Bedarf die Groß-/Kleinschreibung des Kolumnentitels zu ändern. In der Voreinstellung ist diese Anweisung bei Verwendung einer KOMA-Script-Klasse `\@firstofone`, also das unveränderte Argument selbst. Bei Verwendung einer anderen Klasse ist `\MakeMarkcase` im Gegensatz dazu `\MakeUppercase`. Die Anweisung wird von tocbasic jedoch nur definiert, wenn sie noch nicht bereits definiert ist. Sie kann also in einer Klasse in der gewünschten Weise vorbelegt werden und wird dann von tocbasic nicht umdefiniert, sondern verwendet wie vorgefunden.

Beispiel: Sie wollen aus unerfindlichen Gründen, dass die Kolumnentitel in Ihrer Klasse in Kleinbuchstaben ausgegeben werden. Damit dies auch für die Kolumnentitel gilt, die von tocbasic gesetzt werden, definieren Sie:

```
\let\MakeMarkcase\MakeLowercase
```

Erlauben Sie mir einen Hinweis zu `\MakeUppercase`. Diese Anweisung ist zum einen nicht voll expandierbar. Das bedeutet, dass sie im Zusammenspiel mit anderen Anweisungen zu Problemen führen kann. Darüber hinaus sind sich alle Typografen einig, dass beim Versalsatz, also

beim Satz kompletter Wörter oder Passagen in Großbuchstaben, Sperrung unbedingt notwendig ist. Dabei darf jedoch kein fester Abstand zwischen den Buchstaben verwendet werden. Vielmehr muss zwischen unterschiedlichen Buchstaben auch ein unterschiedlicher Abstand gesetzt werden, weil sich unterschiedliche Buchstabenkombinationen unterschiedlich verhalten. Gleichzeitig bilden einige Buchstaben von sich aus bereits Löcher, was bei der Sperrung ebenfalls zu berücksichtigen ist. Pakete wie ulem oder soul können das ebenso wenig leisten wie der Befehl `\MakeUppercase` selbst. Auch die automatische Sperrung mit Hilfe des microtype-Pakets ist diesbezüglich nur eine näherungsweise Notlösung, da die von der konkreten Schrift abhängige Form der Buchstaben auch hier nicht näher betrachtet wird. Da Versalsatz also eher etwas für die absoluten Experten ist und fast immer Handarbeit bedeutet, wird Laien empfohlen, darauf eher zu verzichten oder ihn nur vorsichtig und nicht an so exponierter Stelle wie im Kolumnentitel zu verwenden.

`\deftocheading{`*`Dateierweiterung`*`}{`*`Definition`*`}`

Das Paket tocbasic enthält eine Standarddefinition für das Setzen von Überschriften von Verzeichnissen. Diese Standarddefinition ist durch verschiedene Eigenschaften, die bei der Anweisung `\setuptoc` erläutert werden, konfigurierbar. Sollte diese Möglichkeit einmal nicht ausreichen, so besteht die Möglichkeit, mit `\deftocheading` eine alternative Überschriftenanweisung für ein Verzeichnis mit einer bestimmten *`Dateierweiterung`* zu definieren. Die Definition kann als einzigen Parameter *`#1`* enthalten. Beim Aufruf der Anweisung innerhalb von `\listoftoc` oder `\listofeachtoc` wird als dieses Argument dann der Titel für das Verzeichnis übergeben.

`\setuptoc{`*`Dateierweiterung`*`}{`*`Liste von Eigenschaften`*`}`
`\unsettoc{`*`Dateierweiterung`*`}{`*`Liste von Eigenschaften`*`}`

Mit diesen beiden Anweisungen können *`Eigenschaften`* für eine *`Dateierweiterung`* bzw. das Verzeichnis, das dazu gehört, gesetzt werden. Die *`Liste von Eigenschaften`* ist dabei eine durch Komma getrennte Aneinanderreihung von *`Eigenschaften`*. Das Paket tocbasic wertet folgende Eigenschaften aus:

`leveldown` bedeutet, dass das Verzeichnis nicht mit der obersten Gliederungsebene unterhalb von `\part` – wenn vorhanden `\chapter`, sonst `\section` – erstellt wird, sondern mit einer Überschrift der nächst tieferen Gliederungsebene. Diese Eigenschaft wird von der internen Überschriftenanweisung ausgewertet. Wird hingegen eine eigene Überschriftenanweisung mit `\deftocheading` definiert, liegt die Auswertung der Eigenschaft in der Verantwortung dessen, der die Definition vornimmt. Die KOMA-Script-Klassen setzen diese Eigenschaft bei Verwendung der Option `listof=leveldown` für alle Dateierweiterungen des Besitzers `float`.

`nobabel` bedeutet, dass die normalerweise automatisch verwendete Erweiterung für die Sprachumschaltung mit babel für diese Dateierweiterung nicht verwendet wird. Diese Eigenschaft sollte nur für Verzeichnisse verwendet werden, die nur in einer festen Sprache erstellt werden, in denen also Sprachumschaltungen im Dokument nicht zu berücksichtigen

sind. Sie wird außerdem von Pakage scrwfile für Klonziele verwendet, da die Erweiterungen dort bereits durch das Klonen selbst aus der Klonquelle übernommen werden.

numbered bedeutet, dass das Verzeichnis nummeriert und damit ebenfalls in das Inhaltsverzeichnis aufgenommen werden soll. Diese Eigenschaft wird von der internen Überschriftenanweisung ausgewertet. Wird hingegen eine eigene Überschriftenanweisung mit `\deftocheading` definiert, liegt die Auswertung der Eigenschaft in der Verantwortung dessen, der die Definition vornimmt. Die KOMA-Script-Klassen setzen diese Eigenschaft bei Verwendung der Option `listof=numbered` für alle Dateierweiterungen des Besitzers float.

v3.01 **onecolumn** bedeutet, dass für dieses Verzeichnis automatisch der LaTeX-interne Einspaltenmodus mit `\onecolumn` verwendet wird. Dies gilt jedoch nur, falls dieses Verzeichnis nicht mit der oben beschriebenen Eigenschaft `leveldown` um eine Gliederungsebene nach unten verschoben wurde. Die KOMA-Script-Klassen scrbook und scrreprt setzen diese Eigenschaft per `\AtAddToTocList` (siehe Seite 286) für alle Verzeichnisse mit dem Besitzer `float` oder mit sich selbst als Besitzer. Damit werden beispielsweise das Inhaltsverzeichnis, das Abbildungsverzeichnis und das Tabellenverzeichnis bei diesen beiden Klassen automatisch einspaltig gesetzt. Der Mehrspaltenmodus des multicol-Pakets ist von der Eigenschaft ausdrücklich nicht betroffen.

totoc bedeutet, dass der Titel des Verzeichnisses in das Inhaltsverzeichnis aufgenommen werden soll. Diese Eigenschaft wird von der internen Überschriftenanweisung ausgewertet. Wird mit `\deftocheading` hingegen eine eigene Überschriftenanweisung definiert, liegt die Auswertung der Eigenschaft in der Verantwortung dessen, der die Definition vornimmt. Die KOMA-Script-Klassen setzen diese Eigenschaft bei Verwendung der Option `listof=totoc` für alle Dateierweiterungen des Besitzers float.

Die KOMA-Script-Klassen kennen eine weitere Eigenschaft:

chapteratlist sorgt dafür, dass in dieses Verzeichnis bei jedem neuen Kapitel eine optionale Gliederung eingefügt wird. In der Voreinstellung ist diese Untergliederung dann ein vertikaler Abstand. Näheres zu den Möglichkeiten ist Option `listof` in Abschnitt 3.20, Seite 134 zu entnehmen.

Beispiel: Da KOMA-Script für das Abbildungs- und das Tabellenverzeichnis auf tocbasic aufbaut, gibt es nun eine weitere Möglichkeit, jegliche Kapiteluntergliederung dieser Verzeichnisse zu verhindern:

```
\unsettoc{lof}{chapteratlist}
\unsettoc{lot}{chapteratlist}
```

Wollen Sie hingegen, dass das von Ihnen definierte Verzeichnis mit der Dateierweiterung »*loa*« ebenfalls von der Kapiteluntergliederung der KOMA-Script-Klassen betroffen ist, so verwenden Sie

```
\setuptoc{loa}{chapteratlist}
```

Wollen Sie außerdem, dass bei Klassen, die `\chapter` als oberste Gliederungsebene verwenden, das Verzeichnis automatisch einspaltig gesetzt wird, so verwenden Sie außerdem

```
\ifundefinedorrelax{chapter}{}{%
  \setuptoc{loa}{onecolumn}%
}
```

Die Verwendung von `\ifundefinedorrelax` setzt das Paket scrbase voraus (siehe Abschnitt 10.3, Seite 262).

Sollte Ihr Paket mit einer anderen Klasse verwendet werden, so schadet es trotzdem nicht, dass Sie diese Eigenschaften setzen, im Gegenteil: Wertet eine andere Klasse diese Eigenschaften ebenfalls aus, so nutzt Ihr Paket automatisch die Möglichkeiten jener Klasse.

Wie Sie hier sehen, unterstützt ein Paket, das tocbasic verwendet, bereits ohne nennenswerten Aufwand diverse Möglichkeiten für die dadurch realisierten Verzeichnisse, die sonst einigen Implementierungsaufwand bedeuteten und deshalb in vielen Paketen leider fehlen.

`\iftocfeature{`*`Dateierweiterung`*`}{`*`Eigenschaft`*`}{`*`Dann-Teil`*`}{`*`Sonst-Teil`*`}`

Hiermit kann man für jede *`Eigenschaft`* feststellen, ob sie für eine *`Dateierweiterung`* gesetzt ist. Ist dies der Fall, wird der *`Dann-Teil`* ausgeführt, anderenfalls der *`Sonst-Teil`*. Das kann beispielsweise nützlich sein, wenn Sie eigene Überschriftenanweisungen mit `\deftocheading` definieren, aber die oben beschriebenen Eigenschaften `totoc`, `numbered` oder `leveldown` unterstützen wollen.

13.3. Interne Anweisungen für Klassen- und Paketautoren

Das Paket tocbasic bietet einige interne Anweisungen, deren Benutzung durch Klassen- und Paketautoren freigegeben ist. Diese Anweisungen beginnen alle mit `\tocbasic@`. Aber auch Klassen- und Paketautoren sollten diese Anweisungen nur verwenden und nicht etwas umdefinieren! Ihre interne Funktion kann jederzeit geändert oder erweitert werden, so dass Umdefinierung der Anweisungen die Funktion von tocbasic erheblich beschädigen könnte!

`\tocbasic@extend@babel{`*`Dateierweiterung`*`}`

Das Paket babel (siehe [Bra01]) bzw. ein LaTeX-Kern, der um die Sprachverwaltung von babel erweitert wurde, schreibt bei jeder Sprachumschaltung am Anfang oder innerhalb eines Dokuments in die Dateien mit den Dateierweiterungen `toc`, `lof` und `lot` Anweisungen, um

diese Sprachumschaltung in diesen Dateien mit zu führen. tocbasic erweitert diesen Mechanismus so, dass mit Hilfe von `\tocbasic@extend@babel` auch andere *`Dateierweiterungen`* davon profitieren. Das Argument *`Dateierweiterung`* sollte dabei vollständig expandiert sein! Anderenfalls besteht die Gefahr, dass etwa die Bedeutung eines Makros zum Zeitpunkt der tatsächlichen Auswertung bereits geändert wurde.

In der Voreinstellung wird diese Anweisung normalerweise für alle *`Dateierweiterungen`*, die mit `\addtotoclist` zur Liste der bekannten Dateierweiterungen hinzugefügt werden, aufgerufen. Über die Eigenschaft `nobabel` (siehe `\setuptoc`, Abschnitt 13.2, Seite 292) kann dies unterdrückt werden. Für die Dateinamenerweiterungen `toc`, `lof` und `lot` unterdrückt tocbasic dies bereits selbst, damit die Umschaltung nicht mehrfach in die zugehörigen Dateien eingetragen wird.

Normalerweise gibt es keinen Grund, diese Anweisung selbst aufzurufen. Es sind allerdings Verzeichnisse denkbar, die nicht unter der Kontrolle von tocbasic stehen, also nicht in der Liste der bekannten Dateierweiterungen geführt werden, aber trotzdem die Spracherweiterung für babel nutzen sollen. Für diese ist die Anweisung explizit aufzurufen. Bitte achten Sie jedoch darauf, dass dies für jede Dateierweiterung nur einmal geschieht!

`\tocbasic@starttoc{`*`Dateierweiterung`*`}`

Diese Anweisung ist der eigentlich Ersatz der Anweisung `\@starttoc` aus dem LaTeX-Kern. Es ist die Anweisung, die sich hinter `\listoftoc*` (siehe Abschnitt 13.2, Seite 289) verbirgt. Klassen- oder Paketautoren, die Vorteile von tocbasic nutzen wollen, sollten zumindest diese Anweisung, besser jedoch `\listoftoc` verwenden. Die Anweisung baut selbst auf `\@starttoc` auf, setzt allerdings zuvor `\parskip` und `\parindent` auf 0 und `\parfillskip` auf 0 bis unendlich. Außerdem wird `\@currext` auf die aktuelle Dateierweiterung gesetzt, damit diese in den nachfolgend ausgeführten Haken ausgewertet werden können.

Da LaTeX bei der Ausgabe eines Verzeichnisses auch gleich eine neue Verzeichnisdatei zum Schreiben öffnet, kann der Aufruf dieser Anweisung zu einer Fehlermeldung der Art

```
! No room for a new \write .
\ch@ck ...\else \errmessage {No room for a new #3}
                                                \fi
```

führen, wenn keine Schreibdateien mehr zur Verfügung stehen. Abhilfe kann in diesem Fall das Lades des in Kapitel 12 beschriebenen Pakets scrwfile bieten.

`\tocbasic@@before@hook`
`\tocbasic@@after@hook`

Der Haken `\tocbasic@@before@hook` wird unmittelbar vor dem Einlesen der Verzeichnisdatei, noch vor den mit `\BeforeStartingTOC` definierten Anweisungen ausgeführt. Es ist erlaubt, diesen Haken mit Hilfe von `\g@addto@macro` zu erweitern.

Ebenso wird `\tocbasic@@after@hook` unmittelbar nach der Verzeichnisdatei aber noch vor den mit `\AfterStartingTOC` definieren Anweisungen ausgeführt. Es ist erlaubt, diesen Haken mit Hilfe von `\g@addto@macro` zu erweitern.

KOMA-Script nutzt diese Haken, um die Verzeichnisse mit dynamischer Anpassung an die Breite der Gliederungsnummern zu ermöglichen. Ihre Verwendung ist Klassen und Paketen vorbehalten. Anwender sollten sich auf `\BeforeStartingTOC` und `\AfterStartingTOC` beschränken. Paketautoren sollten ebenfalls vorzugsweise diese beiden Anwenderanweisungen verwenden! Ausgaben innerhalb der beiden Haken sind nicht gestattet!

Wird keine der Anweisungen `\listofeachtoc`, `\listoftoc` und `\listoftoc*` für die Ausgabe der Verzeichnisse verwendet, sollten die beiden Anweisungen für die Haken trotzdem aufgerufen werden.

`\tb@`*`Dateierweiterung`*`@before@hook`
`\tb@`*`Dateierweiterung`*`@after@hook`

Diese Anweisungen werden direkt nach `\tocbasic@@before@hook` bzw. vor `\tocbasic@@after@hook` für das jeweilige Verzeichnis mit der entsprechenden *`Dateierweiterung`* ausgeführt. Sie dürfen keinesfalls von Klassen- und Paketautoren verändert werden. Werden für die Ausgabe der Verzeichnisse die Anweisungen `\listoftoc`, `\listoftoc*` und `\listofeachtoc` nicht verwendet, sollte diese die beiden Anweisungen für die Haken trotzdem aufgerufen werden, soweit sie definiert sind. Die Anweisungen können auch undefiniert sein.

`\tocbasic@listhead{`*`Titel`*`}`

Diese Anweisung wird von `\listoftoc` und `\listofeachtoc` verwendet, um die Anweisung zum Setzen der Überschrift eines Verzeichnisses aufzurufen. Das kann entweder die vordefinierte Anweisung des Pakets tocbasic oder eine individuelle Anweisung sein. Wenn Sie Ihre eigene Anweisung für die Überschrift definieren, können Sie ebenfalls `\tocbasic@listhead` verwenden. In diesem Fall sollte vor dem Aufruf von `\tocbasic@listhead` die Anweisung `\@currext` auf die Dateinamenerweiterung, die zu diesem Verzeichnis gehört, gesetzt werden.

`\tocbasic@listhead@`*`Dateierweiterung`*`{`*`Titel`*`}`

Ist diese individuelle Anweisung für das Setzen einer Verzeichnisüberschrift definiert, so verwendet `\tocbasic@listhead` diese. Anderenfalls definiert `\tocbasic@listhead` diese und ruft sie dann auf.

13.4. Ein komplettes Beispiel

In diesem Abschnitt finden Sie ein komplettes Beispiel, wie eine eigene Gleitumgebung einschließlich Verzeichnis und KOMA-Script-Integration mit Hilfe von tocbasic definiert werden

kann. In diesem Beispiel werden interne Anweisungen, also solche mit »@« im Namen verwendet. Das bedeutet, dass die Anweisungen entweder in einem eigenen Paket, einer Klasse oder zwischen `\makeatletter` und `\makeatother` verwendet werden müssen.

Als erstes wird eine Umgebung benötigt, die diese neue Gleitumgebung bereitstellt. Das geht ganz einfach mit:

```
\newenvironment{remarkbox}{%
  \@float{remarkbox}%
}{%
  \end@float
}
```

Die neue Umgebung heißt also `remarkbox`.

Jede Gleitumgebung hat eine Standardplatzierung. Diese setzt sich aus bekannten Platzierungsoptionen zusammen:

```
\newcommand*{\fps@remarkbox}{tbp}
```

Die neue Gleitumgebung soll also in der Voreinstellung nur oben, unten oder auf einer eigenen Seite platziert werden dürfen.

Gleitumgebungen haben außerdem einen nummerischen Gleittyp. Umgebungen, bei denen das gleiche Bit im Gleittyp gesetzt ist, dürfen sich nicht gegenseitig überholen. Abbildungen und Tabellen haben normalerweise den Typ 1 und 2. Sie dürfen sich also gegenseitig überholen.

```
\newcommand*{\ftype@remarkbox}{4}
```

Die neue Umgebung hat den Typ 4, darf also Tabellen und Abbildungen überholen und von diesen überholt werden.

Gleitumgebungen haben außerdem eine Nummer.

```
\newcounter{remarkbox}
\newcommand*{\remarkboxformat}{%
  Merksatz~\theremarkbox\csname autodot\endcsname}
\newcommand*{\fnum@remarkbox}{\remarkboxformat}
```

Hier wird zunächst ein neuer Zähler definiert, der unabhängig von Kapiteln oder sonstigen Gliederungszählern ist. Dabei definiert LaTeX auch gleich `\theremarkbox` mit der Standardausgabe als arabische Zahl. Diese wird dann in der Definition der formatierten Ausgabe verwendet. Die formatierte Ausgabe wird wiederum als Gleitumgebungsnummer für die Verwendung in `\caption` definiert.

Gleitumgebungen haben Verzeichnisse und diese haben eine Datei mit dem Namen `\jobname` und einer Dateierweiterung.

```
\newcommand*{\ext@remarkbox}{lor}
```

Als Dateierweiterung verwenden wir also »`lor`«.

Die Umgebung selbst steht damit. Es fehlt allerdings das Verzeichnis. Damit wir dabei möglichst wenig selbst machen müssen, verwenden wir das Paket tocbasic. Dieses wird in

Dokumenten mit

```
\usepackage{tocbasic}
```

geladen. Ein Klassen- oder Paketautor würde hingegen

```
\RequirePackage{tocbasic}
```

verwenden.

Nun machen wir die neue Dateierweiterung dem Paket tocbasic bekannt.

```
\addtotoclist[float]{lor}
```

Dabei verwenden wir als Besitzer `float`, damit sich alle anschließend aufgerufenen Optionen von KOMA-Script, die sich auf Verzeichnisse von Gleitumgebungen beziehen, auch auf das neue Verzeichnis beziehen.

Jetzt definieren wir noch einen Titel für dieses Verzeichnis:

```
\newcommand*{\listoflorname}{Verzeichnis der Merksätze}
```

Normalerweise würde man in einem Paket übrigens zunächst einen englischen Titel definieren und dann beispielsweise mit Hilfe des Pakets scrbase Titel für andere Sprachen.

Jetzt müssen wir nur noch definieren, wie ein einzelner Eintrag in dem Verzeichnis aussehen soll:

```
\newcommand*{\l@remarkbox}{\l@figure}
```

Weil das die einfachste Lösung ist, wurde hier festgelegt, dass Einträge in das Verzeichnis der Merksätze genau wie Einträge in das Abbildungsverzeichnis aussehen sollen. Man hätte auch eine explizite Festlegung wie

```
\newcommand*{\l@remarkbox}{\@dottedtocline{1}{1em}{1.5em}}
```

verwenden können.

Außerdem wollen Sie, das Kapiteleinträge sich auf das Verzeichnis auswirken.

```
\setuptoc{lor}{chapteratlist}
```

Das Setzen dieser Eigenschaft ermöglicht dies bei Verwendung einer KOMA-Script-Klasse und jeder anderen Klasse, die diese Eigenschaft unterstützt. Leider gehören die Standardklassen nicht dazu.

Das genügt bereits. Der Anwender kann nun bereits wahlweise mit Hilfe der Optionen einer KOMA-Script-Klasse oder `\setuptoc` verschiedene Formen der Überschrift (ohne Inhaltsverzeichniseintrag, mit Inhaltsverzeichniseintrag, mit Nummerierung) wählen und das Verzeichnis mit `\listoftoc{lor}` ausgeben. Mit einem schlichten

```
\newcommand*{\listofremarkboxes}{\listoftoc{lor}}
```

kann man die Anwendung noch etwas vereinfachen.

Wie Sie gesehen haben, beziehen sich gerade einmal fünf einzeilige Anweisungen, von denen nur drei bis vier wirklich notwendig sind, auf das Verzeichnis selbst. Trotzdem bietet dieses Verzeichnis bereits die Möglichkeit, es zu nummerieren oder auch nicht nummeriert in

das Inhaltsverzeichnis aufzunehmen. Es kann sogar per Eigenschaft bereits eine tiefere Gliederungsebene gewählt werden. Kolumnentitel werden für KOMA-Script, die Standardklassen und alle Klassen, die tocbasic explizit unterstützen, angepasst gesetzt. Unterstützende Klassen beachten das neue Verzeichnis sogar beim Wechsel zu einem neuen Kapitel. Sprachumschaltungen durch babel werden in dem Verzeichnis ebenfalls berücksichtigt.

Natürlich kann ein Paketautor weiteres hinzufügen. So könnte er explizit Optionen anbieten, um die Verwendung von `\setuptoc` vor dem Anwender zu verbergen. Andererseits kann er auch auf diese Anleitung zu tocbasic verweisen, wenn es darum geht, die entsprechenden Möglichkeiten zu erklären. Vorteil ist dann, dass der Anwender automatisch von etwaigen zukünftigen Erweiterungen von tocbasic profitiert. Soll der Anwender aber nicht mit der Tatsache belastet werden, dass für die Merksätze die Dateierweiterung `lor` verwendet wird, so genügt

```
\newcommand*{\setupremarkboxes}{\setuptoc{lor}}
```

um eine als Argument an `\setupremarkboxes` übergebene Liste von Eigenschaften direkt als Liste von Eigenschaften für `lor` zu setzen.

13.5. Alles mit einer Anweisung

Das Beispiel aus dem vorherigen Abschnitt hat gezeigt, dass es mit mit tocbasic recht einfach ist, eigene Gleitumgebungen mit eigenen Verzeichnissen zu definieren. In diesem Beispiel wird gezeigt, dass es sogar noch einfacher gehen kann.

`\DeclareNewTOC[`*`Optionenliste`*`]{`*`Dateierweiterung`*`}`

v3.06

Mit dieser Anweisung wird in einem einzigen Schritt ein neues Verzeichnis, dessen Überschrift und die Bezeichnung für die Einträge unter Kontrolle von tocbasic definiert. Optional können dabei gleichzeitig gleitende oder nicht gleitende Umgebungen definiert werden, innerhalb derer `\caption` Einträge für dieses neue Verzeichnis erzeugt. Auch die Erweiterungen `\captionabove`, `\captionbelow` und `captionbeside` aus den KOMA-Script-Klassen (siehe Abschnitt 3.20 können dann verwendet werden.

`Dateierweiterung` definiert dabei die Dateiendung der Hilfsdatei, die das Verzeichnis repräsentiert, wie dies in Abschnitt 13.1 bereits erläutert wurde. Dieser Parameter muss angegeben werden und darf nicht leer sein!

`Optionenliste` ist eine durch Komma getrennte Liste, wie dies auch von `\KOMAoptions` (siehe Abschnitt 2.4) bekannt ist. Diese Optionen können jedoch *nicht* mit `\KOMAoptions` gesetzt werden! Eine Übersicht über die möglichen Optionen bietet Tabelle 13.1.

v3.06

Tabelle 13.1.: Optionen für die Anweisung `\DeclareNewTOC`

`counterwithin=`*LaTeX-Zähler*
: Falls eine neue Gleitumgebung oder eine nicht gleitende Umgebung definiert wird, so wird für diese auch ein neuer Zähler *Eintragstyp* (siehe Option `type`) angelegt. Dieser Zähler kann in gleicher Weise wie beispielsweise der Zähler `figure` bei `book`-Klassen von Zähler `chapter` abhängt von einem LaTeX-Zähler ahängig gemacht werden.

`float`
: Es wird nicht nur ein neuer Verzeichnistyp definiert, sondern auch Gleitumgebungen *Eintragstyp* (siehe Option `type`) und *Eintragstyp*`*` (vgl. `figure` und `figure*`).

`floatpos=`*Gleitverhalten*
: Jede Gleitumgebung hat ein voreingestelltes *Gleitverhalten*, das über das optionale Argument der Gleitumgebung überladen werden kann. Mit dieser Option wird das *Gleitverhalten* für die optional erstellbare Gleitumgebung (siehe Option `float`) festgelegt. Die Syntax und Semantik ist dabei mit der des optionalen Arguments für die Gleitumgebung identisch. Wird die Option nicht verwendet, so ist das voreingestellte Gleitverhalten `tbp`, also *top*, *bottom*, *page*.

`floattype=`*Gleittyp*
: Jede Gleitumgebung hat einen nummerischen Typ. Gleitumgebungen, bei denen in diesem *Gleittyp* nur unterschiedliche Bits gesetzt sind, können sich gegenseitig überholen. Die Gleitumgebungen `figure` und `table` haben normalerweise die Typen 1 und 2, können sich also gegenseitig überholen. Es sind Typen von 1 bis 31 (alle Bits gesetzt, kann also keinen anderen Typ überholen und von keinem anderen Typen überholt werden) zulässig. Wird kein Typ angegeben, so wird mit 16 der höchst mögliche Ein-Bit-Typ verwendet.

`forcenames`
: Siehe Option `name` und `listname`.

`hang=`*Einzug*
: Mehrzeilige Verzeichniseinträge in hierarchisch angeordneten Verzeichnissen besitzen ab der zweiten Zeile einen Einzug von links. Dieser Einzug wird auch für die Position des Textes in der ersten Zeile eines nummerierten Eintrags verwendet. Der Wert des Einzugs kann über diese Option bestimmt werden. Ohne diese Option wird als *Einzug* 1,5 em verwendet.

...

Tabelle 13.1.: Optionen für die Anweisung `\DeclareNewTOC` (*Fortsetzung*)

`indent=`*Einzug*
: In der hierarchischen Form besitzt jeder Verzeichniseintrag einen Einzug von links. Der Wert des Einzugs kann über diese Option bestimmt werden. Ohne diese Option wird als *Einzug* 1 em verwendet.

`level=`*Gliederungsebene*
: Jeder Eintrag in ein Verzeichnis hat eine nummerische *Gliederungsebene*, die über diese Option gesetzt werden kann. Ist diese Option nicht angegeben, so wird der Standardwert 1 verwendet.

`listname=`*Verzeichnistitel*
: Jedes Verzeichnis hat eine Überschrift, die durch diese Option bestimmt werden kann. Ist die Option nicht angegeben, so wir als Verzeichnistitel »`List of` *Mehrzahl des Eintragstyps*« (siehe Option *types*) verwendet, wobei das erste Zeichen der *Mehrzahl des Eintragstyps* in einen Großbuchstaben gewandelt wird. Es wird auch ein Makro `\list`*Eintragstyp*`name` mit diesem Wert definiert, der jederzeit geändert werden kann. Dieses Makro wird jedoch nur definiert, wenn es nicht bereits definiert ist oder zusätzlich Option `forcenames` gesetzt ist.

`name=`*Eintragsname*
: Sowohl als optionaler Präfix für die Einträge im Verzeichnis als auch für die Beschriftung in einer Gleitumgebung (siehe Option `float`) oder einer nicht gleitenden Umgebung (siehe Option `nonfloat`) wird ein Name für einen Eintrag in das Verzeichnis benötigt. Ohne diese Option wird als *Eintragsname* der *Eintragstyp* (siehe Option `type`) verwendet, bei dem das erste Zeichen in einen Großbuchstaben gewandelt wird. Es wird auch ein Makro `\`*Eintragstyp*`name` mit diesem Wert definiert, der jederzeit geändert werden kann. Dieses Makro wird jedoch nur definiert, wenn es nicht bereits definiert ist oder zusätzlich Option `forcenames` gesetzt ist.

`nonfloat`
: Es wird nicht nur ein neuer Verzeichnistyp definiert, sondern auch eine nicht gleitende Umgebungen *Eintragstyp*`-` (siehe Option `type`), die ähnlich wie eine Gleitumgebung verwendet werden kann, jedoch nicht gleitet und auch nicht die Grenzen der aktuell gültigen Umgebung durchbricht.

...

Tabelle 13.1.: Optionen für die Anweisung `\DeclareNewTOC` (*Fortsetzung*)

`owner=`*`Besitzer`*
: Jedes neue Verzeichnis hat bei tocbasic einen Besitzer (siehe Abschnitt 13.1). Dieser kann hier angegeben werden. Ist kein Besitzer angegeben, so wird der Besitzer »`float`« verwendet, den auch die KOMA-Script-Klassen für das Abbildungs- und das Tabellenverzeichnis verwenden.

`type=`*`Eintragstyp`*
: *`Eintragstyp`* gibt den Typ der Einträge in das entsprechende Verzeichnis an. Der Typ wird auch als Basisname für verschiedene Makros und gegebenenfalls Umgebungen und Zähler verwendet. Er sollte daher nur aus Buchstaben bestehen. Wird diese Option nicht verwendet, so wird für *`Eintragstyp`* die *`Dateierweiterung`* aus dem obligatorischen Argument verwendet.

`types=`*`Mehrzahl des Eintragstyps`*
: An verschiedenen Stellen wird auch die Mehrzahlform des Eintragstyps verwendet, beispielsweise um eine Anweisung `\listof`*`Mehrzahl des Eintragstyps`* zu definieren. Wird diese Option nicht verwendet, so wird als *`Mehrzahl des Eintragstyps`* einfach »*`Eintragstyp`*`s`« verwendet.

Beispiel: Das Beispiel aus Abschnitt 13.4 kann mit Hilfe der neuen Anweisung deutlich verkürzt werden:

```
\DeclareNewTOC[%
  type=remarkbox,%
  types=remarkboxes,%
  float,% Gleitumgebungen sollen definiert werden.
  floattype=4,%
  name=Merksatz,%
  listname={Verzeichnis der Merks\"atze}%
]{lor}
\setuptoc{lor}{chapteratlist}
```

Neben den Umgebungen `remarkbox` und `remarkbox*` sind damit auch der Zähler `remarkbox`, die zur Ausgabe gehörenden Anweisungen `\theremarkbox`, `\remarkboxname` und `\remarkboxformat`, die für das Verzeichnis benötigten `\listremarkboxname` und `\listofremarkboxes`, sowie einige intern verwendete Anweisungen mit Bezug auf die Dateiendung `lor` definiert. Soll der Gleitumgebungstyp dem Paket überlassen werden, so kann Option `floattype` im Beispiel zusätzlich entfallen. Wird zusätzlich die Option `nonfloat` angegeben, wird außerdem eine nicht gleitende Umgebung `remarkbox-` definiert, in der ebenfalls `\caption` verwendet werden kann.

Fremdpakete verbessern mit scrhack

Einige Pakete außerhalb von KOMA-Script arbeiten nicht sehr gut mit KOMA-Script zusammen. Für den KOMA-Script-Autor ist es dabei oftmals sehr mühsam, die Autoren der jeweiligen Pakete von einer Verbesserung zu überzeugen. Das betrifft auch Pakete, deren Entwicklung eingestellt wurde. Deshalb wurde das Paket scrhack begonnen. Dieses Paket ändert Anweisungen und Definitionen anderer Pakete, damit sie besser mit KOMA-Script zusammenarbeiten.

14.1. Entwicklungsstand

Obwohl das Paket bereits seit längerer Zeit Teil von KOMA-Script ist und auch bereits von vielen Anwendern genutzt wird, hat es auch ein Problem. Bei der Umdefinierung von Makros fremder Pakete ist es von der genauen Definition und Verwendung dieser Makros abhängig. Damit ist es gleichzeitig auch von bestimmten Versionen dieser Pakete abhängig. Wird eine unbekannte Version eines der entsprechenden Pakete verwendet, kann scrhack den notwendigen Patch eventuell nicht ausführen. Im Extremfall kann aber umgekehrt der Patch einer unbekannten Version auch zu einem Fehler führen.

Da also scrhack immer wieder an neue Versionen fremder Pakete angepasst werden muss, kann es niemals als fertig angesehen werden. Daher existiert von scrhack dauerhaft nur eine Beta-Version. Obwohl die Benutzung in der Regel einige Vorteile mit sich bringt, kann die Funktion nicht dauerhaft garantiert werden.

14.2. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde.

14.3. Verwendung von tocbasic

In den Anfängen von KOMA-Script gab es von Anwenderseite den Wunsch, dass Verzeichnisse von Gleitumgebungen, die mit dem Paket float erzeugt werden, genauso behandelt werden, wie das Abbildungsverzeichnis oder das Tabellenverzeichnis, das von den KOMA-Script-Klassen selbst angelegt wird. Damals setzte sich der KOMA-Script-Autor mit dem Autor von float in Verbindung, um diesem eine Schnittstelle für entsprechende Erweiterungen zu unterbreiten. In etwas abgewandelter Form wurde diese in Gestalt der beiden Anweisungen `\float@listhead` und `\float@addtolists` realisiert.

Später zeigte sich, dass diese beiden Anweisungen nicht genug Flexibilität für eine umfangreiche Unterstützung aller KOMA-Script-Möglichkeiten boten. Leider hatte der Autor von

float die Entwicklung aber bereits eingestellt, so dass hier keine Änderungen mehr zu erwarten sind.

Andere Paketautoren haben die beiden Anweisungen ebenfalls übernommen. Dabei zeigte sich, dass die Implementierung in einigen Paketen, darunter auch float, dazu führt, dass all diese Pakete nur in einer bestimmten Reihenfolge geladen werden können, obwohl sie ansonsten in keinerlei Beziehung zu einander stehen.

Um all diese Nachteile und Probleme zu beseitigen, unterstützt KOMA-Script diese alte Schnittstelle offiziell nicht mehr. Stattdessen wird bei Verwendung dieser Schnittstelle von KOMA-Script gewarnt. Gleichzeitig wurde in KOMA-Script das Paket tocbasic (siehe Kapitel 13) als zentrale Schnittstelle für die Verwaltung von Verzeichnissen entworfen und realisiert. Die Verwendung dieses Pakets bietet weit mehr Vorteile und Möglichkeiten als die beiden alten Anweisungen.

Obwohl der Aufwand zur Verwendung dieses Pakets sehr gering ist, haben bisher die Autoren der Pakete, die auf die beiden alten Anweisungen gesetzt haben, keine Anpassung vorgenommen. Daher enthält scrhack selbst entsprechende Anpassungen für die Pakete float, floatrow und listings. Allein durch das Laden von scrhack reagieren diese Pakete dann nicht nur auf die Einstellungen von Option `listof`, sondern beachtet auch Sprachumschaltungen durch das babel-Paket. Näheres zu den Möglichkeiten, die durch die Umstellung der Pakete auf tocbasic nun zur Verfügung stehen, ist Abschnitt 13.2 zu entnehmen.

Sollte diese Änderung für eines der Pakete nicht erwünscht sein oder zu Problemen führen, so kann sie selektiv mit den Einstellungen `float=false`, `floatrow=false` und `listings=false` abgeschaltet werden. Wichtig dabei ist, dass eine Änderung der Optionen nach dem Laden des zugehörigen Pakets keinen Einfluss mehr hat!

14.4. Sonderfall hyperref

Ältere Versionen von hyperref vor 6.79h haben bei den Sternformen der Gliederungsbefehle hinter, statt vor oder auf die Gliederungsüberschriften verlinkt. Inzwischen ist dieses Problem auf Vorschlag des KOMA-Script-Autors beseitigt. Da die entsprechende Änderung aber über ein Jahr auf sich warten lies, wurde in scrhack ein entsprechender Patch aufgenommen. Zwar kann dieser ebenfalls durch `hyperref=false` deaktiviert werden, empfohlen wird jedoch stattdessen die aktuelle Version von hyperref zu verwenden. In diesem Fall wird die Änderung durch scrhack automatisch verhindert.

Zusätzliche Informationen zum Paket typearea.sty

In diesem Kapitel finden Sie zusätzliche Informationen zum Paket typearea. Einige Teile des Kapitels sind dabei dem KOMA-Script-Buch [KM08] vorbehalten. Dies sollte kein Problem sein, denn der normale Anwender, der das Paket einfach nur verwenden will, wird diese Informationen normalerweise nicht benötigen. Ein Teil der Informationen richtet sich an Anwender, die ausgefallene Aufgaben lösen oder eigene Pakete schreiben wollen, die auf typearea basieren. Ein weiterer Teil der Informationen behandelt Möglichkeiten von typearea, die aus Gründen der Kompatibilität zu den Standardklassen oder früheren Versionen von KOMA-Script existieren. Die Teile, die nur aus Gründen der Kompatibilität zu früheren Versionen von KOMA-Script existieren, sind in serifenloser Schrift gesetzt und sollten nicht mehr verwendet werden.

15.1. Anweisungen für Experten

In diesem Abschnitt werden Anweisungen beschrieben, die für den normalen Anwender kaum oder gar nicht von Interesse sind. Experten bieten diese Anweisungen zusätzliche Möglichkeiten. Da die Beschreibungen für Experten gedacht sind, sind sie kürzer gefasst.

`\activateareas`

Diese Anweisung wird von typearea genutzt, um die Einstellungen für Satzspiegel und Ränder in die internen LaTeX-Längen zu übertragen, wenn der Satzspiegel innerhalb des Dokuments, also nach `\begin{document}` neu berechnet wurde. Wurde die Option `pagesize` verwendet, so wird diese anschließend mit demselben Wert neu aufgerufen. Damit kann beispielsweise innerhalb von PDF-Dokumenten die Seitengröße tatsächlich variieren.

Experten können diese Anweisung auch verwenden, wenn Sie aus irgendwelchen Gründen, Längen wie `\textwidth` oder `\textheight` innerhalb des Dokuments manuell geändert haben. Der Experte ist dabei für eventuell notwendige Seitenumbrüche vor oder nach der Verwendung jedoch selbst verantwortlich! Darüber hinaus sind alle von `\activateareas` durchgeführten Änderungen lokal!

`\storeareas{`*`Anweisung`*`}`

Mit Hilfe von `\storeareas` wird eine *`Anweisung`* definiert, über die alle aktuellen Seitenspiegeleinstellungen wieder hergestellt werden können. So ist es möglich, die aktuellen Einstellungen zu speichern, anschließend die Einstellungen zu ändern und dann die gespeicherten Einstellungen wieder zu reaktivieren.

Beispiel: Sie wollen in einem Dokument im Hochformat eine Seite im Querformat unterbringen. Mit `\storeareas` kein Problem:

```
\documentclass[pagesize]{scrartcl}

\begin{document}
\rule{\textwidth}{\textheight}

\storeareas\meinegespeichertenWerte
\KOMAoptions{paper=landscape,DIV=current}
\rule{\textwidth}{\textheight}

\clearpage
\meinegespeichertenWerte
\rule{\textwidth}{\textheight}
\end{document}
```

Wichtig ist dabei die Anweisung `\clearpage` vor dem Aufruf von `\meinegespeichertenWerte`, damit die Wiederherstellung erst auf der nächsten Seite erfolgt.

Bei der Verwendung von `\storeareas` ist zu beachten, dass sowohl `\storeareas` als auch die damit definierte *Anweisung* nicht innerhalb einer Gruppe aufgerufen werden sollten. Die Definition der *Anweisung* erfolgt intern mit `\newcommand`. Bei erneuter Verwendung einer bereits definierte *Anweisung* wird eine entsprechende Fehlermeldung ausgegeben.

`\AfterCalculatingTypearea{`*Anweisungen*`}`

Diese Anweisung dient der Verwaltung eines *Hooks* und ermöglicht es dem Experten jedes Mal, nachdem typearea eine neue Aufteilung in Satzspiegel und Ränder berechnet hat, also nach jeder impliziten oder expliziten Ausführung von `\typearea`, *Anweisungen* ausführen zu lassen. Diese *Anweisungen* werden dabei unmittelbar vor `\activateareas` ausgeführt.

15.2. Lokale Einstellungen durch die Datei `typearea.cfg`

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

15.3. Mehr oder weniger obsolete Optionen und Anweisungen

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

Zusätzliche Informationen zu den Hauptklassen scrbook, scrreprt und scrartcl sowie dem Paket scrextend

In diesem Kapitel finden Sie zusätzliche Informationen zu den KOMA-Script-Klassen scrbook, scrreprt und scrartcl und einigen Anweisungen, die auch in scrextend vorhanden sind. Einige Teile des Kapitels sind dabei dem KOMA-Script-Buch [KM08] vorbehalten. Dies sollte kein Problem sein, denn der normale Anwender, der die Klassen einfach nur verwenden will, wird diese Informationen normalerweise nicht benötigen. Ein Teil der Informationen richtet sich an Anwender, die ausgefallene Aufgaben lösen oder eigene Klassen schreiben wollen, die auf einer der drei KOMA-Script-Klassen basieren. Ein weiterer Teil der Informationen behandelt Möglichkeiten der Klassen, die aus Gründen der Kompatibilität zu den Standardklassen oder früheren Versionen von KOMA-Script existieren. Die Teile, die nur aus Gründen der Kompatibilität zu früheren Versionen von KOMA-Script existieren, sind in serifenloser Schrift gesetzt und sollten nicht mehr verwendet werden.

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

16.1. Ergänzende Hinweise zu Benutzeranweisungen

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

16.2. Zusammenspiel von KOMA-Script und anderen Paketen

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

16.3. Anweisungen für Experten

In diesem Abschnitt werden Anweisungen beschrieben, die für den normalen Anwender kaum oder gar nicht von Interesse sind. Experten bieten diese Anweisungen zusätzliche Möglichkeiten. Da die Beschreibungen für Experten gedacht sind, sind sie kürzer gefasst.

`\KOMAClassName`
`\ClassName`

In `\KOMAClassName` ist der Name der aktuell verwendeten KOMA-Script-Klasse abgelegt. Will man also wissen, ob eine oder welche KOMA-Script-Klasse verwendet wird, so kann man einfach auf diese Anweisung testen. Demgegenüber gibt `\ClassName` Auskunft, welche Standardklasse durch diese KOMA-Script-Klasse ersetzt wird.

Es sei in diesem Zusammenhang darauf hingewiesen, dass demgegenüber die Existenz von `\KOMAScript` nicht als Indiz für die Verwendung einer KOMA-Script-Klasse dienen kann.

Zum einen definieren alle KOMA-Script-Pakete diese Anweisung, zum anderen können auch andere Pakete es für sinnvoll erachten, das KOMA-Script-Piktogramm unter diesem Namen zu definieren.

`\addtocentrydefault{`*`Ebene`*`}{`*`Nummer`*`}{`*`Überschrift`*`}`

v3.08

Die KOMA-Script-Klassen verwenden `\addcontentsline` nicht direkt, um Einträge ins Inhaltsverzeichnis vorzunehmen. Stattdessen wird `\addtocentrydefault` mit ganz ähnlichen Argumenten aufgerufen. Die Anweisung kann sowohl für nummerierte als auch für nicht nummerierte Einträge verwendet werden. Dabei gibt *`Ebene`* die Gliederungsebene in Textform also `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph` oder `subparagraph` an. Die formatierte Gliederungsnummer wird über das zweite Argument *`Nummer`* übergeben. Dieses Argument darf auch leer sein. Der Text des Eintrags wird mit *`Überschrift`* angegeben. Es wird empfohlen zerbrechliche Befehle in diesem Argument mit `\protect` zu schützen.

Für das Argument *`Nummer`* gilt noch eine Besonderheit. Ist das Argument leer, so signalisiert dies, dass ein nicht nummerierter Eintrag erzeugt werden soll. In der Voreinstellung wird dies mit

```
\addcontentsline{toc}{Ebene}{Überschrift}
```

erreicht. Ist das Argument jedoch nicht leer, so soll ein nummerierter Eintrag erzeugt werden und *`Nummer`* ist die vorformatierte Gliederungsnummer. In der Voreinstellung verwendet KOMA-Script

```
\addcontentsline{toc}{Ebene}{%
  \protect\numberline{Nummer}Überschrift%
}
```

zur Erzeugung dieses Eintrags.

Paketautoren und Autoren von Wrapper-Klassen können diese Anweisung umdefinieren, um Einfluss auf die Einträge zu nehmen. So wäre beispielsweise denkbar mit

```
\renewcommand{\addtocentrydefault}[3]{%
  \ifstr{#3}{}{%
    \ifstr{#2}{}{%
      \addcontentsline{toc}{#1}{#3}%
    }{%
      \addcontentsline{toc}{#1}{\protect\numberline{#2}#3}%
    }%
  }%
}%
```

dafür zu sorgen, dass Einträge mit leerer *`Überschrift`* erst gar nicht vorgenommen werden. Eine solche Änderung ist in der Praxis jedoch nicht notwendig, da die Unterdrückung leerer

Einträge bereits auf andere Weise in die KOMA-Script-Klassen eingebaut ist. Siehe hierzu auch die Erklärung zu den Gliederungsbefehlen in Abschnitt 3.16 ab Seite 93.

```
\addparttocentry{Nummer}{Überschrift}
\addchaptertocentry{Nummer}{Überschrift}
\addsectiontocentry{Nummer}{Überschrift}
\addsubsectiontocentry{Nummer}{Überschrift}
\addsubsubsectiontocentry{Nummer}{Überschrift}
\addparagraphtocentry{Nummer}{Überschrift}
\addsubparagraphtocentry{Nummer}{Überschrift}
```

v3.08

Auch die oben dokumentierte Anweisung `\addtocentrydefault` wird von den KOMA-Script-Klassen nur dann direkt aufgerufen, wenn für die angegebene *Ebene* keine direkte Anweisunge definiert oder diese `\relax` ist. In der Voreinstellung sind die angegebenen Anweisungen alle so definiert, dass sie ihre *Ebene* und die Argumente direkt an `\addtocentrydefault` weiter geben.

```
\@fontsizefilebase
```

Der auf Seite 307 für die Schriftgrößendateien angegebene Präfix `scrsize`, ist lediglich die Voreinstellung für das interne Makro `\@fontsizefilebase`, die verwendet wird, wenn das Makro beim Laden der Klasse oder des Pakets scrextend noch nicht definiert ist. Autoren von Wrapper-Klassen können dieses Makro abweichend definieren, um andere Schriftgrößendateien zu verwenden. Ebenso können Autoren von Wrapper-Klassen die *fallback*-Lösung dadurch ändern oder abschalten, dass sie das Macro `\changefontsizes`, das als Argument die gewünschte Größe erwartet, umdefinieren.

```
\newkomafont[Warnung]{Element}{Voreinstellung}
\aliaskomafont{Aliasname}{Element}
```

Experten können mit `\newkomafont` eine *Voreinstellung* für die Schrift eines *Element*s definieren. Anschließend kann diese Voreinstellung mit den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 56) verändert werden. Natürlich wird diese Schrift damit noch lange nicht verwendet. Der Experte muss selbst Sorge dafür tragen, dass er an den entsprechenden Stellen die Anweisung `\usekomafont` (siehe Seite 56) für dieses Element in seinen Definitionen einbaut.

Das optionale Argument *Warnung* definiert eine Warnmeldung. Diese wird bei den KOMA-Script-Klassen per `\ClassWarning` oder beim Paket scrextend per `\PackageWarning` immer dann ausgegeben, wenn die Voreinstellung für das Element verändert wird. Dabei wird als Urheber der Warnung die verwendete KOMA-Script-Klasse oder das Paket scrextend ausgegeben.

Mit `\aliaskomafont` kann für ein bereits existierendes *Element* ein *Aliasname* definiert werden. KOMA-Script informiert den Benutzer automatisch über den Namen des tatsächlichen Elements, wenn dieser den *Aliasname* verwendet. *Aliasname*n können beispielsweise

dann eingesetzt werden, wenn der Entwickler sich später einen besseren Namen für ein Element überlegt und der alte Name aus Kompatibilitätsgründen weiter verwendbar bleiben soll. Außerdem kann damit die Benutzerfreundlichkeit erhöht werden, indem einem Element all die Namen als Alias zugeordnet werden, die unterschiedliche Benutzer intuitiv wählen würden. KOMA-Script selbst macht von dieser Möglichkeit reichen Gebrauch.

`\setparsizes{`*Einzug*`}{`*Abstand*`}{`*Endzeilenleerraum*`}`

KOMA-Script bietet mit dieser Anweisung die Möglichkeit, sowohl den Absatzeinzug, als auch den Absatzabstand und den Freiraum am Ende der letzten Zeile des Absatzes einzustellen. Diese Anweisung ist immer dann zu verwenden, wenn die Änderungen auch bei Einstellung `parskip=relative` beachtet werden sollen. KOMA-Script selbst verwendet sie beispielsweise in der Form

```
\setparsizes{0pt}{0pt}{0pt plus 1fil}
```

um sowohl den Einzug als auch den Abstand abzuschalten und am Ende des Absatzes beliebigen Freiraum zu erlauben. Eine solche Maßnahme ist sinnvoll, wenn ein Absatz nur aus einer Box besteht, die ohne Abstand nach oben oder unten gesetzt werden soll und die gesamte Spaltenbreite einnimmt. Soll demgegenüber die Box nur die gesamte Breite einnehmen, jedoch mit der aktuellen Einstellung bezüglich des Absatzabstandes gesetzt werden, so ist

```
\setlength{\parfillskip}{0pt plus 1fil}
```

vorzuziehen.

`\raggedchapterentry`

Diese Anweisung ist nicht mit Anweisungen wie `\raggedsection` zu vergleichen. Während jene Anweisungen es ermöglichen, bestimmte Elemente im Blocksatz oder im rechts- oder linksbündigen oder beidseitigen Flattersatz zu setzen, besteht mit Hilfe von `\raggedchapterentry` lediglich die Möglichkeit, Kapiteleinträge im Inhaltsverzeichnis statt im voreingestellten Blocksatz im linksbündigen Flattersatz zu setzen. Dazu ist die Anweisung als `\raggedright` zu definieren. Siehe hierzu auch die einschränkenden Hinweise in Abschnitt 16.2, Seite 307.

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

```
\chapterheadstartvskip
\chapterheadendvskip
\partheadstartvskip
\partheadmidvskip
\partheadendvskip
\partheademptypage
```

Diese Anweisungen werden innerhalb der Definition der Überschriften `\chapter`,`\part`, `\addchap`, `\addpart` und deren Sternvarianten verwendet. Dabei ist

\chapterheadstartvskip eine Anweisung, die dafür vorgesehen ist, vor der Kapitelüberschrift einen vertikalen Abstand einzufügen. Entsprechend ist \chapterheadendvskip eine Anweisung, die dafür vorgesehen ist, nach der Kapitelüberschrift einen vertikalen Abstand einzufügen.

Für das Einfügen der vertikalen Abstände über und unter Teile-Überschriften sind die Anweisungen \partheadstartvskip und \partheadendvskip vorgesehen. Dabei wird ein Seitenumbruch als Teil des vertikaler Abstandes interpretiert und ist bei scrbook und scrreprt in der Voreinstellung von \partheadendvskip enthalten. Die Anweisung \partheadmidvskip ist für den Abstand zwischen der Teile-Nummer und dem Text der Teile-Überschrift vorgesehen. Die Anweisung \partheademptypage wird bei scrbook und scrreprt gegebenenfalls für die leere Seite nach der Überschrift verwendet.

scrbook, scrreprt

v3.02

Die Voreinstellungen für diese sechs Anweisungen von scrbook und scrreprt sind Tabelle 16.1 und Tabelle 16.2 zu entnehmen. Für scrartcl sind die Voreinstellung der dort definierten drei Anweisungen in Tabelle 16.3 zu finden. Auf [KDP] gibt es außerdem ein Beispiel, bei dem durch Umdefinierung von \chapterheadstartvskip und \chapterheadendvskip Linien über und unter der Kapitelüberschrift gesetzt werden.

scrbook, scrreprt scrartcl

\appendixmore

Bei den KOMA-Script-Klassen gibt es innerhalb der Anweisung \appendix eine Besonderheit. Ist nämlich \appendixmore definiert, so wird diese Anweisung von \appendix ebenfalls ausgeführt. Intern wird dies von den KOMA-Script-Klassen scrbook und scrreprt für die Realisierung der Layoutoption appendixprefix genutzt (siehe Abschnitt 3.16, Seite 89). Dies sollten Sie unbedingt beachten, falls Sie selbst das Makro \appendixmore definieren oder umdefinieren wollen. Ist eine dieser beiden Optionen gesetzt, so erhalten Sie bei \newcommand{\appendixmore}{...} eine Fehlermeldung. Dadurch soll verhindert werden, dass Sie die Optionen außer Kraft setzen, ohne es zu merken.

scrbook, scrreprt

Beispiel: Sie wollen nicht, dass bei Verwendung der Klasse scrbook oder scrreprt im Hauptteil die Kapitel mit einer Präfixzeile versehen werden (siehe Layoutoption chapterprefix und in Abschnitt 3.16, Seite 89). Damit die Konsistenz gewahrt bleibt, wollen Sie auch nicht, dass eine solche Zeile im Anhang verwendet wird. Stattdessen sollen in den Anhängen direkt vor dem Kapitelbuchstaben das Wort »Anhang« in der jeweiligen Sprache stehen. Dies soll auch für die Kolumnentitel gelten. Also verwenden Sie nicht die Layoutoption appendixprefix, sondern definieren in der Dokumentpräambel:

```
\newcommand*{\appendixmore}{%
  \renewcommand*{\chapterformat}{%
    \appendixname~\thechapter\autodot\enskip}%
  \renewcommand*{\chaptermarkformat}{%
    \appendixname~\thechapter\autodot\enskip}}
```

Tabelle 16.1.: Voreinstellungen der Anweisungen für die vertikalen Abstände bei Kapitel-Überschriften von scrbook und scrreprt abhängig von Option `headings`

Mit `headings=big`:

Anweisung	Voreinstellung
`\chapterheadstartvskip`	`\vspace*{2.3\baselineskip}`
`\chapterheadendvskip`	`\vspace*{1.725\baselineskip plus .115\baselineskip minus .192\baselineskip}`

Mit `headings=normal`:

Anweisung	Voreinstellung
`\chapterheadstartvskip`	`\vspace*{2\baselineskip}`
`\chapterheadendvskip`	`\vspace*{1.5\baselineskip plus .1\baselineskip minus .167\baselineskip}`

Mit `headings=small`:

Anweisung	Voreinstellung
`\chapterheadstartvskip`	`\vspace*{1.8\baselineskip}`
`\chapterheadendvskip`	`\vspace*{1.35\baselineskip plus .09\baselineskip minus .15\baselineskip}`

Tabelle 16.2.: Voreinstellungen der Anweisungen für die vertikalen Abstände bei Teile-Überschriften von scrbook und scrreprt

Anweisung	Voreinstellung
`\partheadstartvskip`	`\null\vfil`
`\partheadmidvskip`	`\par\nobreak\vskip 20pt`
`\partheadendvskip`	`\vfil\newpage`
`\partheademptypage`	`\if@twoside\if@openright \null\thispagestyle{empty}% \newpage \fi\fi`

Tabelle 16.3.: Voreinstellungen der Anweisungen für die vertikalen Abstände bei Teile-Überschriften von scrartcl

Anweisung	Voreinstellung
`\partheadstartvskip`	`\addvspace{4ex}`
`\partheadmidvskip`	`\par\nobreak`
`\partheadendvskip`	`\vskip 3ex`

Sollten Sie später dann doch noch entscheiden, dass Sie die Option `appendixprefix` verwenden wollen, so erhalten Sie aufgrund der dann bereits definierten Anweisung `\appendixmore` eine Fehlermeldung. Damit wird verhindert, dass obige Definition unbemerkt die Einstellungen überschreibt, die Sie per Option getroffen haben.

Wenn Sie ein vergleichbares Verhalten des Anhangs für die Klasse scrartcl erreichen wollen, so ist dies ebenfalls möglich. Dazu schreiben Sie in die Präambel Ihres Dokuments:

```
\newcommand*{\appendixmore}{%
  \renewcommand*{\othersectionlevelsformat}[3]{%
    \ifthenelse{\equal{##1}{section}}{%
      \appendixname~}{}%
    ##3\autodot\enskip}
  \renewcommand*{\sectionmarkformat}{%
    \appendixname~\thesection\autodot\enskip}}
```

Sie benötigen dafür außerdem das ifthen-Paket (siehe [Car99a]).

Eine alternative Implementierung wäre:

```
\newcommand*{\appendixmore}{%
  \renewcommand*{\othersectionlevelsformat}[3]{%
    \ifstr{##1}{section}{\appendixname~}{}%
    ##3\autodot\enskip}
  \renewcommand*{\sectionmarkformat}{%
    \appendixname~\thesection\autodot\enskip}}
```

Die dabei verwendete Anweisung `\ifstr` wird von KOMA-Script bereitgestellt. Siehe dazu Abschnitt 10.3, Seite 263.

Die Erklärungen zu den in diesem Beispiel umdefinierten Anweisungen finden Sie in Abschnitt 3.16, Seite 102 und Seite 104.

```
\newbibstyle[Elternstil]{Name}{Anweisungen}
\newblock
\@openbib@code
\bib@beginhook
\bib@endhook
```

Schon die Standardklassen kennen zur Unterteilung der Einträge in das Literaturverzeichnis die Anweisung `\newblock`. Was diese Anweisung genau macht, hängt dabei von den Klassenoptionen ab. Wird die Option `openbib` verwendet, so wird am Ende der Standardklasse die Anweisung `\@openbib@code` und `\newblock` selbst umdefiniert. Von den Standardklassen wird die Anweisung `\@openbib@code` beim Start der Liste für das Literaturverzeichnis – genauer: bei der Festlegung der Parameter für diese Liste – ausgeführt. Es darf davon ausgegangen werden, dass auch viele Pakete, die das Literaturverzeichnis umdefinieren, diese Anweisung entsprechend abarbeiten.

Bei den KOMA-Script-Klassen geschieht etwas ähnliches. Allerdings wird `\@openbib@code` nicht am Ende der Klasse umdefiniert. Stattdessen wird mit `\newbibstyle` der Stil `openstyle` für das Literaturverzeichnis definiert. Die *Anweisungen*, die dabei in der Implementierung angegeben wurden, beinhalten die gewünschten Umdefinierung von `\@openbib@code` und von `\newblock`. Wird nun mit Hilfe der Option `bibliography=openstyle` dieser Literaturverzeichnisstil gewählt, so werden die *Anweisungen* unmittelbar ausgeführt, also `\@openbib@code` und `\newblock` umdefiniert.

Neben `\@openbib@code` und `\newblock` können in *Anweisungen* auch noch `\bib@beginhook` und `\bib@endhook` umdefiniert werden. Die Anweisung `\bib@beginhook` wird unmittelbar nach der Überschrift und der Präambel des Literaturverzeichnisses, aber noch vor der Liste mit den Literatureinträgen ausgeführt. Die Anweisung `\bib@endhook` wird direkt nach dieser Liste am Ende des Literaturverzeichnisses ausgeführt. Im Falle eines mit `\BreakBibliography` (siehe Abschnitt 3.23, Seite 141) unterbrochenen Literaturverzeichnisses werden diese Anweisungen außerdem am Anfang und Ende jedes Teil, also unmittelbar vor und nach `\BreakBibliography` ausgeführt.

Die Anweisungen `\newblock`, `\@openbib@code`, `\bib@beginhook` und `\bib@endhook` werden bei der Verwendung eines neuen Literaturverzeichnisstils zunächst als leer definiert. Danach werden die *Anweisungen* des bei der Definition des Stils optional angegebenen Elternstils ausgeführt und dann erst die *Anweisungen*, die bei Definition des neuen Stils angegeben wurden. Daraus ergibt sich auch, dass jede der vier Anweisungen innerhalb von *Anweisung* bei Bedarf keinesfalls mit `\newcommand`, sondern mit `\renewcommand` definiert werden sollten.

Setzt der Anwender mit den Anweisungen `\AtEndBibliography` und `\AfterBibliographyPreamble` weitere *Anweisungen* für die Ausführung nach der Präambel und am Ende des Literaturverzeichnisses, so werden die mit `\AfterBibliographyPreamble` festgelegten *Anweisungen* einmalig am Anfang des Literaturverzeichnisses nach `\bib@beginhook` und die mit `\AtEndBibliography` festgelegten *Anweisungen* einmalig am Ende des Literaturverzeichnisses vor `\bib@endhook` ausgeführt.

Mit Hilfe des Pakets multicol (siehe [Mit00]) könnte man beispielsweise einen Literaturstil für ein zweispaltiges Literaturverzeichnis definieren:

```
\newbibstyle{twocolumstyle}{%
  \renewcommand*{\bib@beginhook}{\begin{multicols}{2}}%
  \renewcommand*{\bib@endhook}{\end{multicols}}}%
```

Soll es außerdem eine *open*-Variante davon geben, kann man hier die Möglichkeiten der Vererbung verwenden und bei der Definition einen Elternstil mit angeben:

```
\newbibstyle{twocolumstyle}[openstyle]{%
  \renewcommand*{\bib@beginhook}{\begin{multicols}{2}}%
  \renewcommand*{\bib@endhook}{\end{multicols}}}%
```

Die Auswahl eines dieser neuen Stile erfolgt dann einfach wieder über die Option `bibliography`.

16.4. Mehr oder weniger obsolete Optionen und Anweisungen

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

Zusätzliche Informationen zur Briefklasse scrlttr2

In diesem Kapitel finden Sie zusätzliche Informationen zu der KOMA-Script-Klasse scrlttr2. Einige Teile des Kapitels sind dabei dem KOMA-Script-Buch [KM08] vorbehalten. Dies sollte kein Problem sein, denn der Anwender, der die Klasse einfach nur verwenden will, wird diese Informationen normalerweise nicht benötigen. Ein Teil der Informationen richtet sich an Anwender, denen die vordefinierten Möglichkeiten nicht mehr genügen. So befasst sich beispielsweise der erste Abschnitt ausführlich mit den Pseudolängen, die den Briefbogen bestimmen und die für Anpassungen an eigene Briefbogenlayouts abweichend zu setzen sind. Darüber hinaus finden sich in diesem Kapitel auch Informationen über Möglichkeiten, die aus Gründen der Verbesserung der Kompatibilität zur obsoleten KOMA-Script-Klasse scrlettr geschaffen wurden. Es wird auch ausführlich erklärt, wie man einen Brief dieser veralteten Klasse auf die aktuelle Briefklasse übertragen kann.

17.1. Pseudolängen für fortgeschrittene Anwender

TEX arbeitet mit einem festen Satz an Registern. Es gibt Register für Token, für Boxen, für Zähler, für Abstände (englisch: *skip*) und für Größen (englisch: *dimension*). Von all diesen Registern gibt es jeweils 256 Stück. Für LATEX-Längen, die mit `\newlength` angefordert werden, werden Abstandsregister belegt. Sind alle diese Register verbraucht, kann man keine weiteren Längen definieren. Die Briefklasse scrlttr2 würde normalerweise allein für die erste Seite mehr als 20 solche Register verbrauchen. LATEX selbst belegt bereits 40 dieser Register. Das typearea-Paket benötigt ebenfalls einige, so dass ein Viertel der kostbaren Register verbraucht wäre. Aus diesem Grund werden briefspezifische Längen in scrlttr2 eben nicht in Längen, sondern in Makros abgelegt, den Pseudolängen. Der Nachteil dieses Vorgehens besteht darin, dass man mit diesen Makros nicht so einfach rechnen kann wie mit echten Längen.

Wer nun einwenden will, dass LATEX in der empfohlenen und für KOMA-Script benötigten Installation mit ε-TEX inzwischen das oben genannte Beschränkungsproblem nicht mehr besitzt, hat Recht. Allerdings kam diese Entscheidung für scrlttr2 ein wenig zu spät.

Eine Auflistung aller von scrlttr2 definierten und verwendeten Pseudolängen ist Tabelle 17.1 zu entnehmen. Dabei ist auch angegeben, wo in den nachfolgenden Unterabschnitten nähere Erklärungen zu der jeweiligen Pseudolänge zu finden sind.

Abbildung 17.1 auf Seite 321 zeigt eine schematische Darstellung der wichtigsten Abstände auf dem Briefbogen. Dabei sind neben den Pseudolängen für die veränderbaren Abständen zusätzlich in heller Schrift auch die Längen angegeben, die für einige, wenige fest programmierte Abstände verwendet werden. Aus Gründen der Übersichtlichkeit wurde in der Darstellung auf einige weniger häufig benötigten Pseudolängen jedoch auch verzichtet.

Tabelle 17.1.: Von der Klasse scrlttr2 verwendete Pseudolängen

`backaddrheight`
: Höhe der Rücksendeadresse am oberen Rand des Anschriftfeldes (Abschnitt 17.1.3, Seite 326)

`bfoldmarklength`
: Länge der unteren horizontalen Faltmarke (Abschnitt 17.1.1, Seite 322)

`bfoldmarkvpos`
: Abstand der unteren horizontalen Faltmarke von der oberen Kante des Papiers (Abschnitt 17.1.1, Seite 322)

`firstfoothpos`
: Abstand des Brieffußes von der linken Kante des Papiers; Werte größer der Breite oder kleiner der negativen Breite des Papiers werden gesondert behandelt (Abschnitt 17.1.8, Seite 331)

`firstfootvpos`
: Abstand des Brieffußes von der oberen Kante des Papiers (Abschnitt 17.1.8, Seite 330)

`firstfootwidth`
: Breite des Brieffußes (Abschnitt 17.1.8, Seite 331)

`firstheadhpos`
: Abstand des Briefkopfes von der linken Kante des Papiers; Werte größer der Breite oder kleiner der negativen Breite des Papiers werden gesondert behandelt (Abschnitt 17.1.2, Seite 324)

`firstheadvpos`
: Abstand des Briefkopfes von der oberen Kante des Papiers (Abschnitt 17.1.2, Seite 324)

`firstheadwidth`
: Breite des Briefkopfes (Abschnitt 17.1.2, Seite 324)

`foldmarkhpos`
: Abstand der horizontalen Faltmarken von der linken Kante des Papiers (Abschnitt 17.1.1, Seite 323)

...

Tabelle 17.1.: Von der Klasse scrlttr2 verwendete Pseudolängen (*Fortsetzung*)

`foldmarkvpos`
: Abstand der vertikalen Faltmarken von der oberen Kante des Papiers (Abschnitt 17.1.1, Seite 323)

`fromrulethickness`
: Dicke einer optionalen Linie im Briefkopf (Abschnitt 17.1.2, Seite 325)

`fromrulewidth`
: Länge einer optionalen Linie im Briefkopf (Abschnitt 17.1.2, Seite 325)

`lfoldmarkhpos`
: Abstand der vertikalen Faltmarke von der linken Kante des Papiers (Abschnitt 17.1.1, Seite 323)

`lfoldmarklength`
: Länge der vertikalen Faltmarke (Abschnitt 17.1.1, Seite 323)

`locheight`
: Höhe der Absenderergänzung, falls der Wert nicht 0 ist; bei 0 wird stattdessen `toaddrheight` verwendet (Abschnitt 17.1.4, Seite 327)

`lochpos`
: Abstand der Absenderergänzung von der rechten Papierkante, falls der Wert positiv ist, oder negativer Abstand der Absenderergänzung von der linken Papierkante, falls der Wert negativ ist; bei 0 wird stattdessen der negative Wert von `toaddrhpos` verwendet (Abschnitt 17.1.4, Seite 327)

`locvpos`
: Abstand der Absenderergänzung von der oberen Papierkante, falls der Wert nicht 0 ist; bei 0 wird stattdessen `toaddrvpos` verwendet (Abschnitt 17.1.4, Seite 327)

`locwidth`
: Breite des Feldes für die Absenderergänzung, wobei bei einem Wert von 0 die Breite automatisch aufgrund der in Abschnitt 4.10, Seite 184 beschriebenen Option `locfield` berechnet wird (Abschnitt 17.1.4, Seite 327)

`mfoldmarklength`
: Länge der mittleren horizontalen Faltmarke (Abschnitt 17.1.1, Seite 323)

...

Tabelle 17.1.: Von der Klasse scrlttr2 verwendete Pseudolängen (*Fortsetzung*)

`mfoldmarkvpos`
: Abstand der mittleren horizontalen Faltmarke von der oberen Kante des Papiers (Abschnitt 17.1.1, Seite 322)

`pfoldmarklength`
: Länge der Lochermarke (Abschnitt 17.1.1, Seite 323)

`refaftervskip`
: vertikaler Abstand nach der Geschäftszeile (Abschnitt 17.1.5, Seite 329)

`refhpos`
: Abstand der Geschäftszeile von der linken Papierkante, wobei bei einem Wert von 0 automatisch relativ zur Papierbreite zentriert wird (Abschnitt 17.1.5, Seite 328)

`refvpos`
: Abstand der Geschäftszeile von der oberen Kante des Papiers (Abschnitt 17.1.5, Seite 328)

`refwidth`
: Breite der Geschäftszeile (Abschnitt 17.1.5, Seite 328)

`sigbeforevskip`
: vertikaler Abstand zwischen Gruß und Signatur (Abschnitt 17.1.7, Seite 330)

`sigindent`
: Einzug der Signatur gegenüber dem Textkörper (Abschnitt 17.1.7, Seite 330)

`specialmailindent`
: linker Einzug der Versandart innerhalb des Anschriftfeldes (Abschnitt 17.1.3, Seite 327)

`specialmailrightindent`
: rechter Einzug der Versandart innerhalb des Anschriftfeldes (Abschnitt 17.1.3, Seite 327)

`subjectaftervskip`
: vertikaler Abstand nach dem Betreff (Abschnitt 17.1.6, Seite 329)

`subjectbeforevskip`
: zusätzlicher vertikaler Abstand vor dem Betreff (Abschnitt 17.1.6, Seite 329)

...

Tabelle 17.1.: Von der Klasse scrlttr2 verwendete Pseudolängen (*Fortsetzung*)

`subjectvpos`	Abstand des Betreffs von der oberen Kante des Papiers, wobei ein Wert von 0 stattdessen den Betreff gemäß Option `subject` setzt (Abschnitt 17.1.6, Seite 329)
`tfoldmarklength`	Länge der oberen horizontalen Faltmarke (Abschnitt 17.1.1, Seite 323)
`tfoldmarkvpos`	Abstand der oberen horizontalen Faltmarke von der oberen Kante des Papiers (Abschnitt 17.1.1, Seite 322)
`toaddrheight`	Höhe des Anschriftfeldes (Abschnitt 17.1.3, Seite 325)
`toaddrhpos`	Abstand des Anschriftfeldes von der linken Papierkante, falls der Wert positiv ist, oder negativer Abstand des Anschriftfeldes von der rechten Papierkante, falls der Wert negativ ist (Abschnitt 17.1.3, Seite 325)
`toaddrindent`	linker und rechter Einzug der Anschrift innerhalb des Anschriftfeldes (Abschnitt 17.1.3, Seite 326)
`toaddrvpos`	Abstand des Anschriftfeldes von der oberen Kante des Papiers (Abschnitt 17.1.3, Seite 325)
`toaddrwidth`	Breite des Anschriftfeldes (Abschnitt 17.1.3, Seite 325)

`\@newplength{`*`Name`*`}`

Mit Hilfe dieser Anweisung wird eine neue Pseudolänge definiert. Die neue Pseudolänge ist dann über ihren *Namen* eindeutig identifiziert. Wird versucht, eine bereits vorhandene Pseudolänge erneut zu definieren, so wird dies mit einer Fehlermeldung quittiert.

Da der Anwender selbst normalerweise keine eigenen Pseudolängen definieren muss, handelt es sich bei diesem Befehl um keine Benutzeranweisung. Sie kann innerhalb des Dokuments nicht, wohl aber beispielsweise innerhalb einer `lco`-Datei verwendet werden.

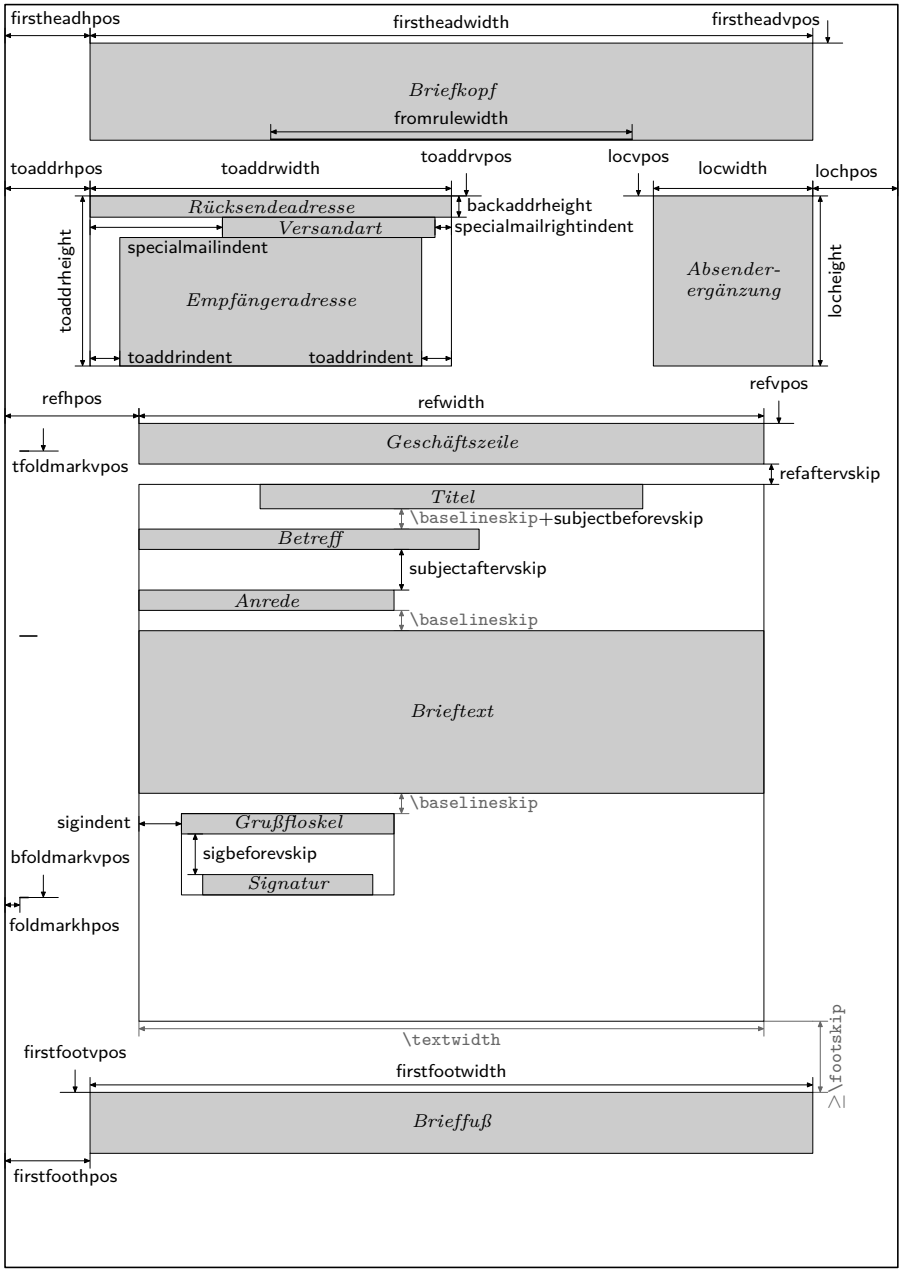


Abbildung 17.1.: Schematische Darstellung der wichtigsten Pseudolängen für den Briefbogen

\@setplength[*Faktor*]{*Pseudolänge*}{*Wert*}
\@addtoplength[*Faktor*]{*Pseudolänge*}{*Wert*}

Mit Hilfe von \@setplength kann einer *Pseudolänge* das Vielfache eines *Wertes* zugewiesen werden. Der *Faktor* wird dabei als optionales Argument übergeben (siehe auch \setlengthtoplength, Abschnitt 4.2, Seite 149).

Mit \@addtoplength kann man zu einer *Pseudolänge* das Vielfache einen *Wert* addieren. Auch dabei wird der *Faktor* als optionales Argument übergeben.

Um einer *Pseudolänge* das Vielfache einer anderen Pseudolänge zuzuweisen oder hinzuzuaddieren, verwendet man innerhalb von *Wert* die Anweisung \useplength (siehe Abschnitt 4.2, Seite 149). Um von einer *Pseudolänge* den Wert einer anderen *Pseudolänge* zu subtrahieren, verwendet man gleichzeitig als *Faktor* ein Minuszeichen, -1 oder einen anderen negativen Faktor.

Da der Anwender selbst normalerweise keine Pseudolängen ändern muss, handelt es sich bei diesen Befehlen um keine Benutzeranweisungen. Sie können innerhalb des Dokuments nicht, wohl aber beispielsweise innerhalb einer lco-Datei verwendet werden.

17.1.1. Faltmarken

Falt- oder Falzmarken sind kleine horizontale Striche am linken und kleine vertikale Striche am oberen Rand. KOMA-Script unterstützt für den Briefbogen derzeit drei konfigurierbare horizontale und eine konfigurierbare vertikale Faltmarke. Dazu wird noch eine horizontale Loch- oder Seitenmittenmarke unterstützt, die nicht in der Vertikalen verschoben werden kann.

tfoldmarkvpos
mfoldmarkvpos
bfoldmarkvpos

Die Briefklasse scrlttr2 kennt insgesamt drei in der vertikalen Platzierung konfigurierbaren Faltmarken. Die Position der oberen Faltmarke vom oberen Papierrand wird von der Pseudolänge tfoldmarkvpos bestimmt. Für die Position der mittleren Faltmarke ist Pseudolänge

v2.97e

mfoldmarkvpos, für die unteren Faltmarke bfoldmarkvpos zuständig. Mit der Locher- oder Seitenmittenmarke kommt noch eine weitere horizontale Marke dazu. Diese wird jedoch immer in der vertikalen Seitenmitte platziert.

Die obere und untere Faltmarke dienen nicht der exakten Drittelung des Papiers beim Falten. Stattdessen soll das Papier mit ihrer Hilfe so geknickt werden können, dass das Feld für die Anschrift in einem Fensterbriefumschlag zu sehen ist. Die Einstellungen sind daher in den vordefinierten lco-Dateienlco-Datei=lco-Datei unterschiedlich gewählt. Eine Besonderheit stellt DINmtext dar. Hier wird zwingend von einem Briefumschlag im Format C6/5 (auch »C6 lang« genannt) ausgegangen. Briefe, die mit dieser Option erstellt wurden, sind weder für Umschläge im Format C5 noch für Umschläge im Format C4 geeignet.

Die mittlere Faltmarke wird für abendländische Briefe normalerweise nicht benötigt. Beispielsweise in Japan gibt es jedoch so unterschiedliche Briefumschläge, dass eine weitere Faltmarke benötigt wurde (siehe die japanischen `lco`-Dateien). An dieser Stelle sei darauf hingewiesen, dass die Bezeichnungen »obere«, »mittlere« und »untere« Faltmarke lediglich eine Sprachkonvention darstellen. Tatsächlich ist nicht festgelegt, dass `tfoldmarkvpos` kleiner als `mfoldmarkvpos` und dieses kleiner als `bfoldmarkvpos` sein muss. Ist eine der Pseudolängen hingegen Null, so wird die entsprechende Faltmarke auch dann nicht gesetzt, wenn sie per Option `foldmarks` (siehe Abschnitt 4.10, Seite 165) explizit aktiviert wurde.

`tfoldmarklength`
`mfoldmarklength`
`bfoldmarklength`
`pfoldmarklength`

v2.97e Diese vier Pseudolängen bestimmen die Länge der vier horizontalen Marken. Dabei gilt eine Besonderheit. Ist die Länge nämlich mit Null angegeben, so werden bei den Pseudolängen `tfoldmarklength`, `mfoldmarklength` und `bfoldmarklength` für die drei in der vertikalen Position konfigurierbaren Faltmarken stattdessen 2 mm als Länge verwendet. Die Länge der Lochermarke, `pfoldmarklength`, wird hingegen auf 4 mm gesetzt.

`foldmarkhpos`

Diese Pseudolänge gibt den Abstand aller horizontalen Faltmarken vom linken Papierrand an. Normalerweise sind das 3,5 mm. Sie können den Wert aber auch in Ihrer eigenen `lco`-Datei ändern, falls Sie einen Drucker verwenden, der einen breiteren unbedruckbaren linken Rand hat. Ob die Faltmarken überhaupt gesetzt werden, hängt außerdem von der Option `foldmarks` ab (siehe Abschnitt 4.10, Seite 165).

`lfoldmarkhpos`

v2.97e Neben den horizontalen Faltmarken gibt es auch noch eine vertikale Faltmarke. Deren Abstand von der linken Papierkante wird über die Pseudolänge `lfoldmarkhpos` bestimmt. Diese Faltmarke wird beispielsweise bei Briefen für einige japanische Chou- oder You-Umschläge benötigt, wenn man diese für A4-Papier verwenden will. Auch für Umschläge im C6-Format kann sie nützlich sein.

`lfoldmarklength`

v2.97e Die Pseudolänge `lfoldmarklength` bestimmt die Länge der vertikalen Faltmarke. Auch hier gibt es die Besonderheit, dass bei einer angegebenen Länge von Null stattdessen 4 mm verwendet werden.

`foldmarkvpos`

v2.97e Diese Pseudolänge gibt den Abstand aller vertikalen Faltmarken vom oberen Papierrand an. Normalerweise sind das 3,5 mm. Sie können den Wert aber auch in Ihrer eigenen `lco`-Datei

ändern, falls Sie einen Drucker verwenden, der einen breiteren unbedruckbaren oberen Rand hat. Ob die Faltmarken überhaupt gesetzt werden, hängt außerdem von der Option `foldmarks` ab (siehe Abschnitt 4.10, Seite 165). Derzeit gibt es nur eine einzige vertikale Faltmarke, die als linke vertikale Faltmarke bezeichnet wird.

`foldmarkthickness`

v2.97c Diese Pseudolänge gibt die Dicke aller Faltmarken an. Voreingestellt sind 0,2 pt, also eine sehr dünne Haarlinie. Insbesondere, wenn die Farbe der Faltmarken geändert wird, kann dies zu wenig sein!

17.1.2. Briefkopf

Unter dem Briefkopf verstehen wir alle Angaben, die den Absender betreffen und die über der Anschrift stehen. Normalerweise würde man erwarten, dass diese über den Seitenstil gesetzt werden. Bei der alten Briefklasse scrlettr war dies auch so. Bei scrlttr2 wird der Briefkopf jedoch unabhängig vom Seitenstil von der Anweisung `\opening` ausgegeben. Dabei wird der Briefkopf absolut positioniert, ist also vom Satzspiegel unabhängig. Die erste Seite eines Briefes, also die Seite mit dem Briefkopf, wird tatsächlich mit dem Seitenstil `empty` gesetzt.

`firstheadvpos`

Die Pseudolänge `firstheadvpos` gibt den Abstand des Briefkopfes von der oberen Papierkante an. Der Wert wird in den vordefinierten `lco`-Dateien unterschiedlich gesetzt. Ein typischer Wert ist 8 mm.

`firstheadhpos`

v3.05 Die Pseudolänge `firstheadhpos` gibt bei einem positiven Wert den Abstand des Briefkopfes von der linken Papierkante an. Ist der Wert sogar größer oder gleich der Breite des Papiers, `\paperwidth`, so wird der Briefkopf horizontal zentriert auf dem Briefbogen platziert. Ein negativer Wert gibt den Abstand des Briefkopfes von der rechten Papierkante an. Ist der Wert jedoch kleiner oder gleich der negativen Breite des Papiers, so wird der Briefkopf bündig zum linken Rand des Satzspiegels platziert.

Voreingestellt ist typischerweise ein Wert von `\maxdimen`, also der größt mögliche Wert für eine Länge. Die Folge ist eine horizontale Zentrierung.

`firstheadwidth`

Die Pseudolänge `firstheadwidth` gibt die Breite des Briefkopfes an. Der Wert wird in den vordefinierten `lco`-Dateien unterschiedlich gesetzt. Während er normalerweise von der Papierbreite und dem horizontalen Abstand der Empfängeradresse vom linken Papierrand abhängt, entspricht er bei `KOMAold` der Breite des Satzspiegels und ist bei `NF` fest auf 170 mm eingestellt.

`fromrulethickness`
`fromrulewidth`

Wie bereits bei Option `fromrule` in Abschnitt 4.10, Seite 169 erwähnt wurde, kann in den vordefinierten Briefköpfen eine Linie im oder unter dem Absender gesetzt werden. Hat die Pseudolänge `fromrulewidth` die Länge 0, so wird dabei die Länge dieser Linie automatisch bestimmt. Dies ist die Voreinstellung bei den vordefinierten `lco`-Dateien. Der Wert kann mit `\@setplength` (siehe Seite 322) in eigenen `lco`-Dateien aber auch abweichend gesetzt werden.

v2.97c

Die voreingestellte Dicke, `fromrulethickness`, der Linie beträgt 0,4 pt.

17.1.3. Anschrift

Unter der Anschrift versteht man normalerweise nur den Namen und die Adresse des Empfängers. Als erste Erweiterung zur Anschrift kann die Versandart betrachtet werden, die etwa bei Einschreiben oder Infobriefen zur Anwendung kommt. Bei Fensterbriefumschlägen wird auch die so genannte Rücksendeadresse zur Anschrift gezählt, da sie im Anschriftfenster zu sehen sein wird. Die Anschrift folgt unmittelbar auf den Briefkopf.

`toaddrvpos`
`toaddrhpos`

Diese Pseudolängen geben den Abstand des Anschriftfensters eines Fensterbriefumschlags vom oberen und vom linken Rand des Papiers an. Sie werden in den vordefinierten `lco`-Dateien unterschiedlich eingestellt. Für `toaddrhpos` gilt außerdem eine Besonderheit. Ist der Wert negativ, so ist sein Betrag der Abstand des Anschriftfeldes vom rechten Rand des Papiers. Sie finden dies beispielsweise bei `SN` oder `NF`. Am kleinsten ist der Wert `toaddrvpos` bei `DINmtext`. Hier kann es schnell passieren, dass der Briefkopf in das Anschriftfenster ragt. Ob das Anschriftfenster überhaupt gesetzt wird, hängt von der Option `addrfield` ab (siehe Abschnitt 4.10, Seite 181).

`toaddrheight`

Diese Pseudolänge gibt die Höhe des Anschriftfeldes einschließlich der Versandart an. Ist keine Versandart angegeben, so wird die Anschrift vertikal innerhalb dieses Feldes zentriert. Ist eine Versandart angegeben, so wird die Anschrift unterhalb der Versandart vertikal im Rest des Anschriftfeldes zentriert.

`toaddrwidth`

Diese Pseudolänge gibt die Breite des Anschriftfensters an. Diese wird in den vordefinierten `lco`-Dateien entsprechend der unterschiedlichen Normen unterschiedlich eingestellt. Typische Werte liegen zwischen 70 mm und 100 mm.

Beispiel: Angenommen, Sie haben das Problem, dass Ihr Drucker einen sehr breiten unbedruckbaren rechten oder linken Rand von 15 mm besitzt. Dadurch kann bei Option `SN` der Briefkopf, die Absenderergänzung und die Anschrift nicht komplett gedruckt werden. Sie erstellen daher eine neue `lco`-Datei mit folgendem Inhalt:

```
\ProvidesFile{SNmmarg.lco}
              [2002/06/04 v0.1 my own lco]
\LoadLetterOption{SN}
\@addtoplength{toaddrwidth}{%
  -\useplength{toaddrhpos}}
\@setplength{toaddrhpos}{-15mm}
\@addtoplength{toaddrwidth}{%
  \useplength{toaddrhpos}}
\endinput
```

Bis Sie sich einen Drucker mit kleineren Rändern zugelegt haben, verwenden Sie dann `SNmmarg` an Stelle von `SN`.

`toaddrindent`

Manchmal will man, dass die Anschrift nicht am linken Rand des Anschriftfensters beginnt und bis zum rechten Rand des Fensters reicht, sondern ein wenig eingezogen wird. Der Wert dieses Einzugs kann über die Pseudolänge `toaddrindent` festgelegt werden. Typischerweise ist dieser Wert jedoch 0 pt.

v3.03 Bei jeder der Einstellungen `addrfield=PP`, `addrfield=image` und `addrfield=backgroundimage` (siehe Abschnitt 4.10, Seite 181) wird beim Wert 0 pt stattdessen ein Einzug von 8 mm verwendet. Soll hier tatsächlich kein Einzug verwendet werden, so kann mit 1 sp ein vernachlässigbar kleiner Einzug gesetzt werden. Desweiteren wird `toaddrindent` bei den genannten Einstellungen für `addrfield` auch für den Abstand zum rechten Rand des Anschriftfensters verwendet.

`backaddrheight`

Bei Fensterbriefumschlägen wird der Absender häufig in einer kleinen Schrift einzeilig über der Empfängeradresse ausgegeben. Diese Absenderangabe nennt man Rücksendeadresse, da sie im Anschriftfenster sichtbar ist und der Post bei unzustellbaren Briefen für die Rücksendung an den Absender dient. In dieser Adresse muss daher auch nur die Information enthalten sein, die zu diesem Zweck notwendig ist.

Die Höhe, die innerhalb des Anschriftfensters für die Rücksendeadresse zur Verfügung steht, ist in der Pseudolänge `backaddrheight` abgelegt. Der Wert wird in den vordefinierten `lco`-Dateien typischerweise auf 5 mm eingestellt. Ob die Rücksendeadresse überhaupt gesetzt wird, bestimmt der Anwender mit den Optionen `addrfield` (siehe Abschnitt 4.10, Seite 181) und `backaddress` (siehe Abschnitt 4.10, Seite 181).

`specialmailindent`
`specialmailrightindent`

Zwischen Rücksendeadresse und Empfängeradresse kann noch eine optionale Versandart gesetzt werden. Diese wird genau dann gesetzt, wenn die Variable `specialmail` einen Inhalt hat. Die Ausrichtung wird mit Hilfe der Pseudolängen `specialmailindent` und `specialmailrightindent` festgelegt. Diese geben den linken und rechten Einzug der Zeile an. In den vordefinierten `lco`-Dateien ist `specialmailindent` auf den dehnbaren Wert `\fill` gesetzt, während `specialmailrightindent` auf 1 em eingestellt ist. Damit wird die Versandart 1 em vom rechten Rand des Anschriftfensters gesetzt.

`PPheadheight`
`PPheadwidth`

v3.03 Die Pseudolänge `PPheadheight` gibt bei den beiden Einstellungen `addrfield=PP` und `addrfield=backgroundimage` die Höhe an, die am Anfang des Adressfeldes für den Port-Payé-Kopf reserviert wird. Die Pseudolänge `PPheadwidth` wird nur bei `addrfield=PP` (siehe Abschnitt 4.10, Seite 181) verwendet und gibt die Breite des linken Feldes des Port-Payé-Kopf mit dem P. P.-Logo, der Postleitzahl und dem Ort an. Die Breite des rechten Feldes mit dem Code für den Absender und der Priorität ist durch die Restbreite bestimmt.

Den normalerweise voreingestellten Wert von 0 mm für Pseudolänge `PPheadheight` ändert scrlttr2 selbständig in 20,74 pt. Den normalerweise voreingestellten Wert von 0 mm für `PPheadwidth` ändert scrlttr2 selbständig in 42 mm.

`PPdatamatrixvskip`

v3.03 Durch diese Pseudolänge wird der vertikale Abstand zwischen dem Port-Payé-Kopf und der Data-Matrix bei `addrfield=PP` (siehe Abschnitt 4.10, Seite 181) festgelegt. Den normalerweise voreingestellten Wert von 0 mm ändert scrlttr2 selbständig in 9 mm. Die Data-Matrix wird rechtsbündig zum Port-Payé-Kopf gesetzt.

17.1.4. Absenderergänzungen

Insbesondere bei Geschäftsbriefen reicht der Platz im Briefkopf und im Seitenfuß oftmals nicht aus, um alle Angaben des Absenders unter zu bringen. Für die zusätzlichen Informationen wird oft der Platz neben der Anschrift genutzt. In dieser Anleitung wird dieses Feld *Absenderergänzung* genannt.

`locheight`
`lochpos`
`locvpos`
`locwidth`

v2.97d Die Pseudolängen `locwidth` und `locheight` geben die Breite und Höhe der Absenderergänzung an. Die Pseudolängen `lochpos` und `locvpos` geben die Abstände von der rechten und

der oberen Papierkante an. Diese Werte werden in den vordefinierten `lco`-Dateien typischerweise auf 0 pt gesetzt. Dieser Wert nimmt eine Sonderstellung ein. Er bedeutet, dass die Werte erst bei `\opening` an Hand der Breite des Papiers, der Breite des Anschriftfensters und des Abstandes des Anschriftfensters von der linken und oberen Papierkante gesetzt werden. Dabei findet auch die Option `locfield` (siehe Abschnitt 4.10, Seite 184) Berücksichtigung. Wie auch bei `toaddrhpos` nehmen negative Werte für `lochpos` eine Sonderstellung ein. Es wird dann statt des Abstandes vom rechten Papierrand der Betrag von `lochpos` als Abstand vom linken Papierrand verwendet. Die Bedeutung ist also genau umgekehrt zu der bei `toaddrhpos` (siehe Abschnitt 17.1.3, Seite 325).

17.1.5. Geschäftszeile

Die Geschäftszeile kann auch länger als eine Zeile sein und wird nur gesetzt, wenn mindestens eine der Variablen für die Geschäftszeile nicht leer ist. Es werden nur die Felder gesetzt, die nicht leer sind. Um ein scheinbar leeres Feld zu setzen, muss man der entsprechenden Variablen einen scheinbar leeren Inhalt, beispielsweise ein festes Leerzeichen oder `\null`, geben. Wird auf die Geschäftszeile verzichtet, so werden an ihrer Stelle Bezeichnung und Inhalt der Variablen `date` ausgegeben. Informationen, wie Variablen zur Geschäftszeile hinzugefügt oder entfernt werden, sind in Abschnitt 17.2, Seite 331 zu finden.

`refvpos`

Diese Pseudolänge gibt den Abstand der Geschäftszeile von der Oberkante des Papiers an. Ihr Wert wird in den vordefinierten `lco`-Dateien unterschiedlich eingestellt. Typische Werte liegen zwischen 80,5 mm und 98,5 mm.

`refwidth`
`refhpos`

Die Pseudolänge `refwidth` gibt die Breite an, die für die Geschäftszeile zur Verfügung steht. Ihr Wert wird in den vordefinierten `lco`-Dateien typischerweise auf 0 pt gesetzt. Dieser Wert hat eine besondere Bedeutung. Es wird damit keineswegs festgelegt, dass für die Geschäftszeile keine Breite zur Verfügung steht. Vielmehr bedeutet der Wert, dass die verfügbare Breite erst innerhalb von `\opening` ermittelt wird. Die dort ermittelte Breite richtet sich dann nach der Einstellung der Option `refline` (siehe Abschnitt 4.10, Seite 186). Gleichzeitig wird dann auch `refhpos` entsprechend der Option gesetzt. Bei `refline=wide` wird die Geschäftszeile zentriert, wohingegen sie bei `refline=narrow` am Satzspiegel links ausgerichtet wird.

Ist `refwidth` von Null verschieden, wird die Breite der Geschäftszeile also nicht von der Option `refline` bestimmt, so gibt `refhpos` den Abstand der Geschäftszeile von der linken Papierkante an. Ist dieser Abstand Null, so wird die Geschäftszeile so ausgerichtet, dass das Verhältnis zwischen ihrem Abstand von der linken Papierkante zu ihrem Abstand von der rechten Papierkante dem Verhältnis zwischen dem Abstand des Satzspiegels von der linken

Papierkante zu seinem Abstand von der rechten Papierkante entspricht. Bei auf dem Papier horizontal zentriertem Satzspiegel wird also auch die Geschäftszeile zentriert.

In der Regel werden diese Sonderfälle für die häufigsten Anwendungen von geringem Interesse sein. Die einfachste Regel lautet hier: Entweder wird `refwidth` auf Null belassen und die Breite und Ausrichtung der Geschäftszeile über die Option `refline` bestimmt, oder sowohl `refwidth` als auch `refhpos` werden vom Anwender vorgegeben.

`refaftervskip`

Diese Pseudolänge gibt den vertikalen Abstand an, der nach der Geschäftszeile eingefügt werden soll. Der Wert wird in den vordefinierten `lco`-Dateien eingestellt. Er wirkt sich unmittelbar auf die Höhe des Textbereiches der ersten Seite aus. Der typische Wert liegt zwischen einer und zwei Zeilen.

17.1.6. Betreff

Der Betreff eine Briefes wird in unterschiedlichen Ländern unterschiedlich gesetzt. Die einen haben ihn gerne vor der Anrede, die anderen setzen ihn danach. Einige Berufsgruppen wollen ihn teilweise sogar vor der Geschäftszeile haben.

`subjectvpos`

v3.01 Ist der Wert dieser Pseudolänge 0 pt, so bestimmt die Option `subject` (siehe Abschnitt 4.10, Seite 190) die Position des Betreffs. Dabei spielen dann auch die nachfolgend erklärten Pseudolängen `subjectbeforevskip` und `subjectaftervskip` ihre Rolle. Bei allen anderen Werten wird der Betreff mit dem entsprechenden Abstand von der oberen Papierkante platziert. Es wird empfohlen in diesem Fall darauf zu achten, dass genügend Platz zur Verfügung steht, damit Überschneidungen mit anderen Elementen unwahrscheinlich sind.

Beispiel: Einige wenige Berufsgruppen ziehen es vor, wenn der Betreff noch vor der Geschäftszeile steht. Hierzu kann man die Position wie folgt wählen, wobei auch die Position der Geschäftszeile angepasst wird:

```
\ProvidesFile{lawsubj.lco}
              [2008/11/03 lawyers lco file]
\@setplength{subjectvpos}{\useplength{refvpos}}
\@addtoplength{refvpos}{3\baselineskip}
\endinput
```

Will man, dass zwischen Betreff und Geschäftszeile noch mindestens eine Zeile frei bleibt, hat man so Platz für maximal zwei Zeilen Betreff.

`subjectbeforevskip`
`subjectaftervskip`

v3.01

Wird der Betreff nicht absolut platziert, sondern vor oder nach der Anrede, so kann vor und nach dem Betreff ein zusätzlicher Abstand eingefügt werden. Der Abstand vor dem Betreff trifft dabei ggf. mit anderen Abständen, etwa dem automatischen Abstand von einer Zeile nach dem Titel, zusammen. In der Voreinstellung wird daher in der Regel kein weiterer Abstand an dieser Stelle eingefügt. Der Abstand nach dem Betreff beträgt in der Voreinstellung der Klasse hingegen zwei Zeilen.

17.1.7. Schlussgruß

Der Schlussgruß eines Briefes besteht aus mehreren Teilen. Neben der Grußformel selbst gibt es noch die Unterschrift und die Signatur, eine Art Erläuterung zur Unterschrift.

`sigindent`
`sigbeforevskip`

Grußfloskel und Erläuterung der Unterschrift werden innerhalb einer Box gesetzt. Die Breite dieser Box wird durch die längste Zeile innerhalb von Grußfloskel und Erläuterung bestimmt.

Die Box wird mit dem durch die Pseudolänge `sigindent` bestimmten Einzug gesetzt. In den vordefinierten `lco`-Dateien ist der Einzug auf 0 mm gesetzt.

Zwischen Grußfloskel und Erläuterung wird ein vertikaler Abstand eingefügt, der mit der Pseudolänge `sigbeforevskip` festgelegt ist. In den vordefinierten `lco`-Dateien ist der Wert auf zwei Zeilen eingestellt. In diese Lücke setzen Sie dann Ihre Unterschrift.

17.1.8. Briefbogenfuß

Die erste Seite eines Briefes, der Briefbogen, enthält nicht nur einen eigenen Kopf, den Briefkopf. Diese Seite enthält auch einen eigenen Fuß. Auch dieser wird nicht über den Seitenstil, sondern direkt von `\opening` ausgegeben.

`firstfootvpos`

Diese Pseudolänge gibt den Abstand des Fußes der ersten Briefseite von der Oberkante des Papiers an. Es wird außerdem dafür gesorgt, dass der Textbereich nicht in den Fuß hineinragt. Hierzu wird auf der ersten Seite gegebenenfalls die Höhe des Textbereiches mit Hilfe von `\enlargethispage` verkleinert. Mit Hilfe der Option `enlargefirstpage` (siehe Abschnitt 4.10, Seite 168) kann dafür gesorgt werden, dass die Höhe des Textbereiches umgekehrt gegebenenfalls auch vergrößert wird. Damit kann dann der Abstand zwischen Textbereich und Fuß der ersten Seite auf den Wert der Länge `\footskip` verringert werden.

v2.9t

Bei Kompatibilitätseinstellungen bis Version 2.9t (siehe `version` in Abschnitt 4.4, Seite 31) wird außer bei `KOMAold` und `NF` in allen vordefinierten `lco`-Dateien (siehe Abschnitt 4.21) der Fuß abhängig vom Satzspiegel gesetzt. Damit hat dann auch `enlargefirstpage` keine Wirkung. Ab Version 2.9u bekommt der Fuß eine Position am unteren Ende des Papiers.

Damit ist dann die Höhe des Satzspiegels des Briefbogens eventuell auch von der Option `enlargefirstpage` abhängig.

v2.97e Sollte der Briefbogenfuß mittels Option `firstfoot=false` (siehe Abschnitt 4.10, Seite 192) abgeschaltet sein, so wird die Einstellung von `firstfootvpos` ignoriert und stattdessen `\paperheight` angenommen. Es bleibt damit dann ein minimaler unterer Rand von `\footskip`.

`firstfoothpos`

v3.05 Die Pseudolänge `firstfoothpos` gibt bei einem positiven Wert den Abstand des Briefbogenfußes von der linken Papierkante an. Ist der Wert sogar größer oder gleich der Breite des Papiers, `\paperwidth`, so wird der Fuß horizontal zentriert auf dem Briefbogen platziert. Ein negativer Wert gibt den Abstand des Fußes von der rechten Papierkante an. Ist der Wert jedoch kleiner oder gleich der negativen Breite des Papiers, so wird der Fuß bündig zum linken Rand des Satzspiegels platziert.

Voreingestellt ist typischerweise ein Wert von `\maxdimen`, also der größt mögliche Wert für eine Länge. Die Folge ist eine horizontale Zentrierung.

`firstfootwidth`

Diese Pseudolänge gibt die Breite des Fußes der ersten Briefseite, also des Briefbogens, an. Der Wert stimmt in den vordefinierten `lco`-Dateien mit `firstheadwidth` überein.

17.2. Variablen für fortgeschrittene Anwender

Neben der Möglichkeit, vordefinierte Variablen zu verwenden, bietet scrlttr2 auch Anweisungen, um neue Variablen zu definieren oder deren automatische Verwendung innerhalb der Geschäftszeile zu beeinflussen.

```
\newkomavar[Bezeichnung]{Name}
\newkomavar*[Bezeichnung]{Name}
\removereffields
\defaultreffields
\addtoreffields{Name}
```

Mit `\newkomavar` wird eine neue Variable definiert. Diese Variable wird über *Name* angesprochen. Optional kann eine *Bezeichnung* für die Variable *Name* angegeben werden. Eine *Bezeichnung* wird dabei im Unterschied zum *Name* nicht verwendet, um auf eine Variable zuzugreifen. Vielmehr ist die *Bezeichnung* eine Ergänzung zum Inhalt einer Variable, die ähnlich ihrem Inhalt ausgegeben werden kann.

Mit der Anweisung `\addtoreffields` kann die Variable *Name* der Geschäftszeile (siehe Abschnitt 4.10, Seite 186) hinzugefügt werden. Dabei wird die *Bezeichnung* und der Inhalt der Variablen an das Ende der Geschäftszeile angehängt, falls ihr Inhalt nicht leer ist. Die

Sternvariante \newkomavar* entspricht der Variante ohne Stern mit anschließendem Aufruf der Anweisung \addtoreffields. Bei der Sternvariante wird die Variable also automatisch zur Geschäftszeile hinzugefügt.

Beispiel: Angenommen, Sie benötigen in der Geschäftszeile ein zusätzliches Feld für eine Durchwahl. Sie können das Feld dann wahlweise mit

```
\newkomavar[Durchwahl]{myphone}
\addtoreffields{myphone}
```

oder kürzer mit

```
\newkomavar*[Durchwahl]{myphone}
```

definieren.

Im Fall, dass eine Variable für die Geschäftszeile definiert wird, sollten Sie immer eine Bezeichnung dafür angeben.

Mit der Anweisung \removereffields können alle Variablen aus der Geschäftszeile entfernt werden. Dies betrifft auch die in der Klasse vordefinierten Variablen. Die Geschäftszeile ist dann leer. Sie können dies beispielsweise nutzen, wenn Sie die Reihenfolge der Variablen in der Geschäftszeile ändern wollen.

Zur Wiederherstellung der Reihenfolge der vordefinierten Variablen in der Geschäftszeile dient \defaultreffields. Dabei werden auch alle selbst definierten Variablen aus der Geschäftszeile entfernt.

Das Datum sollte der Geschäftszeile nicht über die Anweisung \addtoreffields hinzugefügt werden. Stattdessen stellt man mit Option date ein, ob das Datum links, rechts oder gar nicht in der Geschäftszeile erscheinen soll. Diese Optionen haben darüber hinaus auch einen Einfluss auf die Position des Datums, wenn gar keine Geschäftszeile verwendet wird.

```
\usekomavar[Anweisung]{Name}
\usekomavar*[Anweisung]{Name}
```

Die Anweisungen \usekomavar und \usekomavar* sind wie alle Anweisungen, von denen es eine Sternvariante gibt oder die ein optionales Argument besitzen, nicht voll expandierbar. Bei Verwendung innerhalb von \markboth, \markright oder ähnlichen Anweisungen muss dennoch kein \protect vorangestellt werden. Selbstverständlich gilt dies bei Verwendung von scrpage2 auch für \markleft. Allerdings können die Anweisungen nicht innerhalb von \MakeUppercase und ähnlichen Anweisungen verwendet werden, die direkten Einfluss auf ihr Argument haben. Diese Anweisungen können jedoch als optionales Argument angegeben werden. So erhält man beispielsweise den Inhalt einer Variable in Großbuchstaben mit:

```
\usekomavar[\MakeUppercase]{Name}
```

```
\ifkomavarempty{Name}{Wahr}{Falsch}
\ifkomavarempty*{Name}{Wahr}{Falsch}
```

Für die exakte Funktion ist wichtig, dass der Inhalt der Variablen soweit expandiert wird, wie dies mit `\edef` möglich ist. Bleiben dabei Leerzeichen oder unexpandierbare Makros wie `\relax` übrig, so gilt der Inhalt auch dann als nicht leer, wenn die Verwendung der Variablen zu keiner Ausgabe führen würde.

Auch diese Anweisung kann nicht innerhalb von `\MakeUppercase` oder ähnlichen Anweisungen verwendet werden. Sie ist jedoch robust genug, um beispielsweise als Argument von `\markboth` oder `\footnote` zu funktionieren.

17.3. `lco`-Dateien für fortgeschrittene Anwender

Obwohl jedes von typearea einstellbare Format verwendbar ist, kann es bei der Ausgabe der ersten Briefseite mit manchen Formaten zu unerwünschten Ergebnissen kommen. Das liegt aber nicht am Konzept der Klasse, sondern daran, dass derzeit hauptsächlich Parametersätze für ISO A4 existieren. Leider gibt es keine allgemein gültigen Regeln, um die Position von Anschriftfeldern und Ähnlichem für beliebige Papierformate zu berechnen. Es ist jedoch möglich, auch für andere Papierformate Parametersätze zu erstellen.

17.3.1. Überwachung des Papierformats

Derzeit existieren Parametersätze und `lco`-Dateien für A4-Papier und letter-Papier. Die Klasse scrlttr2 versteht aber theoretisch sehr viel mehr Papierformate. Daher ist es notwendig zu überwachen, ob die korrekte Papiergröße eingestellt ist.

```
\LetterOptionNeedsPapersize{Optionsname}{Papiergröße}
```

Damit man bei Verwendung einer in der `lco`-Datei nicht vorgesehenen *Papiergröße* zumindest gewarnt wird, sind in den mit KOMA-Script ausgelieferten `lco`-Dateien `\LetterOptionNeedsPapersize`-Anweisungen zu finden. Als erstes Argument wird dabei der Name der `lco`-Datei ohne die Endung »`.lco`« übergeben. Als zweites Argument wird die Papiergröße übergeben, für die diese `lco`-Datei gedacht ist.

Werden nacheinander mehrere `lco`-Dateien geladen, so kann jede dieser `lco`-Dateien eine Anweisung `\LetterOptionNeedsPapersize` enthalten. Innerhalb von `\opening` wird jedoch nur auf die jeweils letzte angegebene *Papiergröße* geprüft. Wie das nachfolgende Beispiel zeigt, ist es daher für den versierten Anwender leicht möglich, `lco`-Dateien mit Parametersätzen für andere Papierformate zu schreiben.

Beispiel: Nehmen wir einmal an, dass Sie A5-Papier in normaler Ausrichtung, also hochkant oder portrait, für Ihre Briefe verwenden. Nehmen wir weiter an, dass Sie diese in normale Fensterbriefumschläge im Format C6 stecken. Damit wäre prinzipiell die Position des Adressfeldes die gleiche wie bei einem normalen Brief in A4 nach DIN.

Der Unterschied besteht im Wesentlichen darin, dass das A5-Papier nur einmal gefaltet werden muss. Sie wollen deshalb verhindern, dass die obere und die untere Faltmarke gesetzt wird. Dies erreichen Sie beispielsweise, indem Sie die Marken außerhalb des Papiers platzieren.

```
\ProvidesFile{a5.lco}
              [2002/05/02 letter class option]
\LetterOptionNeedsPapersize{paper=a5}{a5}
\@setplength{tfoldmarkvpos}{\paperheight}
\@setplength{bfoldmarkvpos}{\paperheight}
\endinput
```

Eleganter wäre es natürlich, die Marken mit Hilfe der Option `foldmarks` abzuschalten. Außerdem muss auch noch die Position des Seitenfußes, also die Pseudolänge `firstfootvpos`, angepasst werden. Ich überlasse es dem Leser, dafür einen geeigneten Wert zu ermitteln. Mit einer solchen `lco`-Datei ist es lediglich wichtig, dass andere `lco`-Dateioptionen wie `SN` vor dem Laden von »`a5.lco`«, angegeben werden.

17.3.2. Positionen sichtbar machen

Wenn man selbst `lco`-Dateien entwickelt, beispielsweise um die Positionen von verschiedenen Feldern des Briefbogens an eigene Wünsche oder Notwendigkeiten anzupassen, ist es hilfreich, wenn zumindest einige Elemente direkt sichtbar gemacht werden können. Zu diesem Zweck existiert die `lco`-Datei `visualize.lco`. Diese kann wie jede `lco`-Datei geladen werden. Allerdings ist das Laden dieser *Letter Class Option* auf die Dokumentpräambel beschränkt und seine Auswirkungen können nicht wieder rückgängig gemacht werden. Die `lco`-Datei bedient sich der Pakete eso-pic und graphicx, die nicht zu KOMA-Script gehören.

v3.04

`\showfields{`*`Feldliste`*`}`

Mit dieser Anweisung kann die Visualisierung von Feldern des Briefbogens aktiviert werden. Das Argument *`Feldliste`* ist dabei eine durch Komma separierte Liste der Feldern, die visualisiert werden sollen. Folgende Felder werden derzeit unterstützt:

`test` – ist ein Testfeld der Größe 10 cm auf 15 cm, das jeweils 1 cm vom oberen und linken Papierrand entfernt ist. Dieses Testfeld existiert zu Debuggingzwecken. Es dient als Vergleichsmaß für den Fall, dass im Dokumenterstellungsprozess die Maße verfälscht werden.

`head` – ist der Kopfbereich des Briefbogens. Es handelt sich hier um ein nach unten offenes Feld.

`foot` – ist der Fußbereich des Briefbogens. Es handelt sich hier um ein nach unten offenes Feld.

`address` – ist das Anschriftfenster.

`location` – ist das Feld der Absenderergänzung.

`refline` – ist die Geschäftszeile. Es handelt sich hier um ein nach unten offenes Feld.

Mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 56) für das Element `field` kann die Farbe der Visualisierung geändert werden. Voreingestellt ist `\normalcolor`.

```
\setshowstyle{Visualisierungsstil}
\edgesize
```

In der Voreinstellung werden die einzelnen Felder durch Begrenzungslinien markiert. Dies entspricht dem *Visualisierungsstil* `frame`. Nach unten offene Felder werden dabei nicht komplett umrahmt, sondern unten offen mit kleinen Pfeilen nach unten dargestellt. Als Alternative hierzu steht auch der *Visualisierungsstil* `rule` zur Verfügung. Dabei wird das Feld farbig hinterlegt. Hierbei kann nicht zwischen geschlossenen und nach unten offenen Feldern unterschieden werden. Stattdessen werden nach unten offene Felder mit einer Mindesthöhe dargestellt. Der dritte verfügbare *Visualisierungsstil* ist `edge`. Dabei werden die Ecken der Felder markiert. Bei nach unten offenen Feldern entfallen dabei die unteren Eckmarkierungen. Die Größe der Eckmarkierungen ist im Makro `\edgesize` abgelegt und mit 1 ex voreingestellt.

```
\showenvelope(Breite,Höhe)(HOffset,VOffset)[Zusatz]
\showISOenvelope{Format}[Zusatz]
\showUScommercial{Format}[Zusatz]
\showUScheck[Zusatz]
\unitfactor
```

Diese Anweisungen dienen dazu eine Seite mit einer Zeichnung eines Umschlags auszugeben. Der Umschlag wird dabei immer um 90° gedreht auf einer eigenen Seite im Maßstab 1:1 ausgegeben. Das Anschriftfenster wird automatisch aus den aktuellen Daten für die Anschriftposition auf dem Briefbogen: `toaddrvpos`, `toaddrheight`, `toaddrwidth` und `toaddrhpos`, erzeugt. Hierfür ist es notwendig zu wissen, um welchen Wert der gefaltete Briefbogen auf jeder Seite kleiner als die *Breite* und *Höhe* des Briefbogens ist. Sind diese beiden Werte, *HOffset* und *VOffset*, bei `\showenvelope` nicht angegeben, so wird versucht, sie aus den Faltmarken und der Papiergröße selbst zu berechnen.

Die Anweisungen `\showISOenvelope`, `\showUScommercial` und `\showUScheck` basieren auf `\showenvelope`. Mit `\showISOenvelope` kann ein ISO-Umschlag im *Format* C4, C5, C5/6, DL (auch bekannt als C5/6) oder C6 erzeugt werden. Mit `\showUScommercial` wird hingegen ein US-Commercial-Umschlag im *Format* 9 oder 10 ausgegeben. `\showUScheck` schließlich ist für Umschläge im US-Check-Format zuständig.

Innerhalb des Umschlags wird die Lage des Briefbogens gestrichelt angedeutet. Die dabei verwendete Farbe kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont`

(siehe Abschnitt 4.9, Seite 56) für das Element `letter` verändert werden. Voreingestellt ist `\normalcolor`.

Die Umschlagzeichnung wird automatisch bemaßt. Die Farbe der Bemaßung und die Größe deren Beschriftung kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 56) für das Element `measure` verändert werden. Voreingestellt ist hier `\normalcolor`. Die Bemaßung erfolgt in Vielfachen von `\unitlength` mit einer maximalen Genauigkeit von 1/`\unitfactor`, wobei die Genauigkeit der TEX-Arithmetik die tatsächliche Grenze darstellt. Voreingestellt ist 1. Eine Umdefinierung ist mit `\renewcommand` möglich.

Beispiel: Es wird ein Beispielbrief im Format ISO A4 erzeugt. Die unterstützten Felder sollen zwecks Überprüfung ihrer Position mit gelben Rahmenlinien markiert werden. Desweiteren soll die Position des Fensters in einem Umschlag der Größe DL mit Hilfe einer Zeichnung überprüft werden. Die Maßlinien in dieser Zeichnung sollen rot und die Maßzahlen in kleinerer Schrift ausgegeben werden, wobei die Maßzahlen in cm mit einer Genauigkeit von 1 mm ausgegeben werden sollen. Der gestrichelte Briefbogen im Umschlag soll hingegen grün eingefärbt werden.

```
\documentclass[visualize]{scrlttr2}
\usepackage{xcolor}
\setkomafont{field}{\color{yellow}}
\setkomafont{measure}{\color{red}\small}
\setkomafont{letter}{\color{green}}
\showfields{head,address,location,refline,foot}
\usepackage[ngerman]{babel}
\usepackage{lipsum}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
                         54321 Musterheim}
\begin{letter}{%
    Petra Mustermann\\
    Vor dem Berg 1\\
    12345 Musterhausen%
  }
\opening{Hallo,}
\lipsum[1]
\closing{Bis dann}
\end{letter}
\setlength{\unitlength}{1cm}
\renewcommand*{\unitfactor}{10}
\showISOenvelope{DL}
\end{document}
```

Auf der ersten Seite findet sich nun der Briefbogen, auf der zweiten Seite wird die Zeichnung des Umschlags ausgegeben.

Bezüglich der Bemaßung ist zu beachten, dass ungünstige Kombinationen von `\unitlength` und `\unitfactor` sehr schnell einen TEX-Fehler der Art *arithmetic overflow* provozieren. Ebenso kann es geschehen, dass ausgegebene Maßzahlen geringfügig vom tatsächlichen Wert abweichen. Beides sind keine Fehler von `visualize`, sondern lediglich Implementierungsgrenzen.

17.4. Unterstützung verschiedener Sprachen

Die Klasse scrlttr2 unterstützt viele Sprachen. Dazu zählen Deutsch (`german` für alte deutsche Rechtschreibung, `ngerman` für neue deutsche Rechtschreibung, `austrian` für Österreichisch mit alter deutscher Rechtschreibung und `naustrian` für Österreichisch mit neuer deutscher Rechtschreibung), Englisch (`english` ohne Angabe, ob amerikanisches oder britisches Englisch, `american` und `USenglish` für Amerikanisch, `british` und `UKenglish` für Britisch), Französisch, Italienisch, Spanisch, Niederländisch, Kroatisch, Finnisch, Norwegisch und Schwedisch.

v3.09

v3.08

Zwischen den Sprachen wird bei Verwendung des babel-Paketes (siehe [Bra01]) mit der Anweisung `\selectlanguage{`*`Sprache`*`}` gewechselt. Andere Pakete wie german (siehe [Rai98a]) und ngerman (siehe [Rai98b]) besitzen diese Anweisung ebenfalls. In der Regel erfolgt eine Sprachumschaltung jedoch bereits aufgrund des Ladens eines solchen Paketes.

Erlauben Sie mir noch einen Hinweis zu den Sprachumschaltpaketen. Das Paket french (siehe [Gau03]) nimmt neben der Umdefinierung der Begriffe aus Tabelle 17.3 weitere Änderungen vor. So definiert es etwa die Anweisung `\opening` um. Dabei geht es einfach davon aus, dass `\opening` immer wie in der Standardbriefklasse letter definiert ist. Dies ist bei scrlttr2 jedoch nicht der Fall. Das Paket french zerstört deshalb die Definition aus scrlttr2 und arbeitet nicht korrekt mir KOMA-Script zusammen. Ich betrachte dies als Fehler des Paketes french.

Wird das Paket babel für die Umschaltung auf die Sprache `french` verwendet und ist gleichzeitig das Paket french installiert, so ergeben sich eventuell genau dieselben Probleme, weil in diesem Fall Teile des Paketes french verwendet werden.

Mit Babel ab Version 3.7j tritt dieses Problem jedoch nur noch auf, wenn per Option explizit angegeben wird, dass babel das french-Paket verwenden soll. Kann nicht sicher gestellt werden, dass nicht eine alte Version von babel verwendet wird, so empfehle ich, mit

```
\usepackage[...,frenchb,...]{babel}
```

französische Sprache auszuwählen. Gegebenenfalls ist dann aber trotzdem mit `\selectlanguage{french}` auf Französisch umzuschalten.

Es ist nicht auszuschließen, dass mit anderen Sprachen und Paketen ähnliche Probleme auftreten.

```
\captionsenglish
\captionsUSenglish
\captionsamerican
\captionsbritish
\captionsUKenglish
\captionsgerman
\captionsngerman
\captionsaustrian
\captionsnaustrian
\captionsfrench
\captionsitalian
\captionsspanish
\captionsdutch
\captionscroatian
\captionsfinnish
\captionsnorsk
\captionsswedish
```

Wird die Sprache eines Briefes gewechselt, so werden über diese Anweisungen die Begriffe aus Tabelle 17.3, Seite 340 umdefiniert. Sollte das verwendete Sprachumschaltpaket dies nicht unterstützen, so können obige Anweisungen notfalls auch direkt verwendet werden.

```
\dateenglish
\dateUSenglish
\dateamerican
\datebritish
\dateUKenglish
\dategerman
\datengerman
\dateaustrian
\datenaustrian
\datefrench
\dateitalian
\datespanish
\datedutch
\datecroatian
\datefinnish
\datenorsk
\dateswedish
```

Je nach verwendeter Sprache werden auch die Datumsangaben des nummerischen Datums (siehe Option `numericaldate` in Abschnitt 4.10, Seite 185) in unterschiedlicher Form umgesetzt. Die genauen Angaben können der Tabelle 17.2 entnommen werden.

Tabelle 17.2.: Sprachabhängige Ausgabeformate für das Datum

Anweisung	Ausgabebeispiel
\dateenglish	24/12/1993
\dateUSenglish	12/24/1993
\dateamerican	12/24/1993
\datebritish	24/12/1993
\dateUKenglish	24/12/1993
\dategerman	24. 12. 1993
\datengerman	24. 12. 1993
\dateaustrian	24. 12. 1993
\datenaustrian	24. 12. 1993
\datefrench	24. 12. 1993
\dateitalian	24. 12. 1993
\datespanish	24. 12. 1993
\datedutch	24. 12. 1993
\datecroatian	24. 12. 1993.
\datefinnish	24.12.1993.
\datenorsk	24.12.1993
\dateswedish	24/12 1993

```
\yourrefname
\yourmailname
\myrefname
\customername
\invoicename
\subjectname
\ccname
\enclname
\headtoname
\headfromname
\datename
\pagename
\phonename
\faxname
\emailname
\wwwname
\bankname
```

Die aufgeführten Anweisungen enthalten die jeweiligen sprachtypischen Begriffe. Diese können für die Realisierung einer weiteren Sprache oder aber auch zur eigenen freien Gestaltung angepasst werden. Wie dies geht, wird in Abschnitt 10.4 erklärt. Von scrlttr2 werden die Begriffe erst nach der Präambel, also bei `\begin{document}` gesetzt. Sie sind daher vorher nicht verfügbar und können vorher auch nicht geändert werden. In Tabelle 17.3 sind die Voreinstellungen für

Tabelle 17.3.: Voreinstellungen für die sprachabhängigen Begriffe bei Verwendung der Sprachen `english` und `ngerman` soweit nicht durch die Pakete zur Sprachumschaltung bereits definiert

Anweisung	`english`	`ngerman`
`\bankname`	Bank account	Bankverbindung
`\ccname`[1]	cc	Kopien an
`\customername`	Customer no.	Kundennummer
`\datename`	Date	Datum
`\emailname`	Email	E-Mail
`\enclname`[1]	encl	Anlagen
`\faxname`	Fax	Fax
`\headfromname`	From	Von
`\headtoname`[1]	To	An
`\invoicename`	Invoice no.	Rechnungsnummer
`\myrefname`	Our ref.	Unser Zeichen
`\pagename`[1]	Page	Seite
`\phonename`	Phone	Telefon
`\subjectname`	Subject	Betrifft
`\wwwname`	Url	URL
`\yourmailname`	Your letter of	Ihr Schreiben vom
`\yourrefname`	Your ref.	Ihr Zeichen

1 Diese Begriffe werden normalerweise bereits von Sprachpaketen wie babel definiert und dann von scrlttr2 nicht überschrieben. Abweichungen im Wortlaut sind daher möglich und der Anleitung des verwendeten Sprachpakets zu entnehmen.

`english` und `ngerman` zu finden.

17.5. Von der obsoleten scrlettr zur aktuellen scrlttr2

Die alte Briefklasse scrlettr wurde mit Einführung von scrlttr2 (siehe Kapitel 4) 2002 obsolet. scrlettr sollte für neue Briefe besser nicht mehr verwendet werden. Die Klasse wird nicht mehr weiterentwickelt und es findet daher auch kein Support mehr dafür statt. Wer dennoch unbedingt die Anleitung zur alten Briefklasse benötigt, kann diese in `scrlettr.dtx` finden. Am besten führt man dazu einige LaTeX-Läufe mit jener Datei durch, also beispielsweise:

```
latex scrlettr.dtx
mkindex scrlettr
latex scrlettr.dtx
mkindex scrlettr
latex scrlettr.dtx
```

Man erhält so die Datei `scrlettr.dvi` mit der Anleitung. Wer stattdessen `scrlettr.pdf` haben möchte, ersetzt `latex` durch `pdflatex`.

Um den Umstieg von der alten auf die neue Klasse zu erleichtern, existiert mit `KOMAold` eine Kompatibilitätseinstellung. Grundsätzlich ist in der neuen Klasse die gesamte alte Funktionalität enthalten. Ohne `KOMAold` ist jedoch die Benutzerschnittstelle eine andere und auch die Voreinstellungen stimmen nicht überein. Näheres dazu ist Abschnitt 4.21, Tabelle 4.18 zu entnehmen.

Im KOMA-Script-Buch [KM08] finden sich an dieser Stelle weitere Informationen.

Änderungliste

Sie finden im folgenden eine Auflistung aller wesentlichen Änderungen der Benutzerschnittstelle im KOMA-Script-Paket der neueren Zeit. Die Liste ist sowohl nach Versionen als auch nach Paket- und Klassennamen sortiert. Zu jeder Version, jedem Paket und jeder Klasse ist jeweils angegeben, auf welchen Seiten dieser Dokumentation die Änderungen zu finden sind. Auf den entsprechenden Seiten finden Sie dazu passende Randmarkierungen.

Literaturverzeichnis

Sie finden im Folgenden eine ganze Reihe von Literaturangaben. Auf all diese wird im Text verwiesen. In vielen Fällen handelt es sich um Dokumente oder ganze Verzeichnisse, die im Internet verfügbar sind. In diesen Fällen ist statt eines Verlages eine URL angegeben. Wird auf ein LaTeX-Paket verwiesen, so findet der Verweis in der Regel in der Form „CTAN:// *Verweis*“ statt. Der Präfix „CTAN://“ steht dabei für das TeX-Archiv eines jeden CTAN-Servers oder -Spiegels. Sie können den Präfix beispielsweise durch ftp://ftp.dante.de/tex-archive/ ersetzen. Bei LaTeX-Paketen ist außerdem zu beachten, dass versucht wurde, die Version anzugeben, auf die im Text Bezug genommen wurde. Bei einigen Paketen war es mehr ein Ratespiel, eine einheitliche Versionsnummer und ein Erscheinungsdatum zu finden. Auch muss die angegebene Version nicht immer die neueste verfügbare Version sein. Wenn Sie sich ein Paket neu besorgen und installieren, sollten Sie jedoch zunächst immer die aktuelle Version ausprobieren. Bevor Sie ein Dokument oder Paket von einem Server herunterladen, sollten Sie außerdem überprüfen, ob es sich nicht bereits auf Ihrem Rechner befindet.

[Ame02] American Mathematical Society:
User's guide for the amsmath package, Februar 2002.
CTAN://macros/latex/required/amslatex/math/.

[BCJ+05] Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley und Rainer Schöpf:
The LaTeX2e Source, Dezember 2005.

[Bra01] Johannes Braams:
Babel, a multilangual package for use with LaTeX's standard document classes, Februar 2001.
CTAN://macros/latex/required/babel/.

[Car99a] David Carlisle:
The ifthen package, September 1999.
CTAN://macros/latex/base/.

[Car99b] David Carlisle:
The keyval package, März 1999.
CTAN://macros/latex/required/graphics/.

[Car99c] David Carlisle:
The tabularx package, Januar 1999.
CTAN://macros/latex/tools.

[Car99d] David P. Carlisle:
Packages in the 'graphics' bundle, Februar 1999.
CTAN://macros/latex/required/graphics/.

[Car04] David Carlisle:
The longtable package, Februar 2004.
CTAN://macros/latex/required/tools/.

[Che11] Florent Chervet:
tabu and longtabu, Februar 2011.
CTAN://macros/latex/contrib/tabu/.

[Dal99] Patrick W. Daly:
Natural sciences citations and references, Mai 1999.
CTAN://macros/latex/contrib/natbib/.

[DUD96] DUDEN:
Die deutsche Rechtschreibung. DUDENVERLAG, Mannheim, 21. Auflage, 1996.

[Fai05] Robin Fairbairns:
footmisc — a portmanteau package for customising footnotes in LaTeX, März 2005.
CTAN://macros/latex/contrib/footmisc.

[Gau03] Bernard Gaulle:
Les distributions de fichiers de francisation pour latex, Dezember 2003.
CTAN://language/french/.

[KDP] *KOMA-Script Homepage*.
http://www.komascript.de.

[Keh97] Roger Kehr:
XINDY, A Flexible Indexing System, 1997.

[Ker07] Dr. Uwe Kern:
Extending LaTeX's color facilities: the xcolor package, Januar 2007.
CTAN://macros/latex/contrib/xcolor/.

[Kie99] Axel Kielhorn:
adrconv, November 1999.
CTAN://macros/latex/contrib/adrconv/.

[KM08] Markus Kohm und Jens Uwe Morawski:
KOMA-Script. Edition DANTE. Lehmanns Media, Berlin, 3. Auflage, 2008, ISBN 978-3-86541-291-1.

[Koh02] Markus Kohm:
Satzspiegelkonstruktionen im Vergleich. Die TeXnische Komödie, 4:28–48, 2002. DANTE e. V.

[Koh03] Markus Kohm:
Moderne Briefe mit KOMA-Script. Die TEXnische Komödie, 2:32–51, 2003. DANTE e. V.

[Koh06] Markus Kohm:
Creating more than one index using splitidx and splitindex, Juni 2006.
CTAN://macros/latex/contrib/splitindex.

[Lam87] Leslie Lamport:
MakeIndex: An Index Processor For LATEX, Februar 1987.
CTAN://indexing/makeindex/doc/makeindex.pdf.

[Lap06] Olga Lapko:
The floatrow package, Juli 2006.
CTAN://macros/latex/contrib/floatrow/.

[Leh11] Philipp Lehman:
The etoolbox package, Januar 2011.
CTAN://macros/latex/contrib/etoolbox/.

[Lem08] Werner Lemberg:
CJK, Dezember 2008.
CTAN://languages/chinese/CJK/.

[Lin01] Anselm Lingnau:
An improved environment for floats, Juli 2001.
CTAN://macros/latex/contrib/float/.

[Mit00] Frank Mittelbach:
An environment for multicolumn output, Juli 2000.
CTAN://macros/latex/required/tools/.

[Obe07] Heiko Oberdiek:
The selinput package, September 2007.
CTAN://macros/latex/contrib/oberdiek.

[Pac] Jean Marie Pacquet:
KomaLetter2; Example by Jean-Marie Pacquet (French style). Wiki.
http://wiki.lyx.org/Examples/KomaLetter2#toc6.

[Rai98a] Bernd Raichle:
german package, Juli 1998.
CTAN://language/german/.

[Rai98b] Bernd Raichle:
ngerman package, Juli 1998.
CTAN://language/german/.

[Sch09] Martin Schröder:
The ragged2e package, Juni 2009.
CTAN://macros/latex/contrib/ms/.

[Sch10] R Schlicht:
The microtype package: An interface to the micro-typographic extensions of pdfTEX, Januar 2010.
CTAN://macros/latex/contrib/microtype.

[SKPH99] Walter Schmidt, Jörg Knappen, Hubert Partl und Irene Hyna:
LaTeX 2ε-Kurzbeschreibung, April 1999.
CTAN://info/lshort/german/.

[Tea05a] LaTeX3 Project Team:
LaTeX 2ε font selection, November 2005.
CTAN://macros/latex/doc/fntguide.pdf.

[Tea05b] LaTeX3 Project Team:
LaTeX 2ε for authors, November 2005.
CTAN://macros/latex/doc/usrguide.pdf.

[Tea06] LaTeX3 Project Team:
LaTeX 2ε for class and package writers, Februar 2006.
CTAN://macros/latex/doc/clsguide.pdf.

[Tob00] Geoffrey Tobin:
setspace LaTeX package, Dezember 2000.
CTAN://macros/latex/contrib/setspace/.

[Tsc60] Jan Tschichold:
Erfreuliche Drucksachen durch gute Typographie. Ravensburger Buchverlag Otto Maier GmbH, 1960, ISBN 3-87512-413-8. Nachdruck bei: MaroVerlag, Augsburg, 2001.

[Tsc87] Jan Tschichold:
Ausgewählte Aufsätze über Fragen der Gestalt des Buches und der Typographie. Birkhäuser Verlag, Basel, 2. Auflage, 1987.

[Ume00] Hideo Umeki:
The geometry package, Juni 2000.
CTAN://macros/latex/contrib/geometry/.

[vO00] Piet van Oostrum:
Page layout in LaTeX, Oktober 2000.
CTAN://macros/latex/contrib/fancyhdr/.

[WF00] Hans Peter Willberg und Friedrich Forssman:
Erste Hilfe in Typografie. Verlag Hermann Schmidt, Mainz, 2000.

[Wik] Wiki:
Deutsche TEX-FAQ.
http://projekte.dante.de/DanteFAQ/WebHome.

Index

Fett hervorgehobene Zahlen geben die Seiten der Erklärung zu einem Stichwort wieder. Normal gedruckte Zahlen verweisen hingegen auf Seiten mit zusätzlichen Informationen zum jeweiligen Stichwort.

Allgemeiner Index

G

H

I

K

L

M

N

O

Befehle, Umgebungen und Variablen

B

C

D

E

K

L

M

N

S

T

Längen und Zähler

B

F

Elemente mit der Möglichkeit zur Schriftumschaltung

Dateien, Klassen und Pakete

Klassen- und Paketoptionen

I

K

L

M

N

O

P

R

S

T

U

V