# KOMA-Script

ein wandelbares LaTeX 2$_\varepsilon$-Paket

Die Anleitung

# KOMA-Script

Markus Kohm

2026-10-19

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

Rechtliche Hinweise:

Der Autor dieser Anleitung ist in dieser Eigenschaft nicht verantwortlich für die Funktion oder Fehler der in dieser Anleitung beschriebenen Software. Bei der Erstellung von Texten und Abbildungen wurde mit großer Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden.

Der Autor kann jedoch für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen. Für Verbesserungsvorschläge und Hinweise auf Fehler ist der Autor 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.

Freie Bildschirm-Version ohne Optimierung des Umbruchs

Diese Anleitung ist als Bestandteil von KOMA-Script frei im Sinne der LaTeX Project Public License Version 1.3c. Eine für KOMA-Script gültige deutsche Übersetzung liegt KOMA-Script in der Datei »lppl-de.txt« bei. Diese Anleitung – auch in gedruckter Form – darf nur zusammen mit den übrigen Bestandteilen von KOMA-Script weitergegeben und verteilt werden. Eine Verteilung der Anleitung unabhängig von den übrigen Bestandteilen von KOMA-Script bedarf der ausdrücklichen Genehmigung des Autors.

Eine umbruchoptimierte und erweiterte Ausgabe der KOMA-Script-Anleitung ist in der dante-Edition von Lehmanns Media erschienen (siehe [Koh20a]).

Für meine Frau ohne deren tagtägliche Arbeit ich mir meine ehrenamtlichen Taten nicht leisten könnte!

# Vorwort zu KOMA-Script 3.49.3

Der seit langem dokumentierte Umbau bei KOMA-Script ist mit dieser Version in eine entscheidende Phase eingetreten. Erstmals werden Optionen und Anweisungen in größerem Umfang aus Klassen und Paketen entfernt. Dies betrifft alle Optionen, vor deren Verwendung schon lange gewarnt wird.

Wer also beispielsweise trotz aller Warnungen noch immer eine seit zwanzig Jahren veraltete Option wie `noheadsepline` verwendet, wird damit in der aktuellen Version allenfalls noch eine Warnung erzielen. Die tatsächliche Funktion ist den aktuellen und dokumentierten Optionen vorbehalten. Wer trotz aller Warnungen noch immer eine veraltete Anweisung verwendet, erhält sogar eine Fehlermeldung.

Anwender sollten darauf gefasst sein, dass es ab KOMA-Script 3.50 dann auch bei veralteten Optionen keine Warnung mehr geben wird. Auch Rückschaltmöglichkeiten mit Option `version` soll es ab dann nicht mehr geben.

Für Anwender, die diese Übergangsphase verpasst haben, gibt es im Wiki die Auflistung von Fehlermeldungen und Warnungen. Darüber hinaus gibt es einen Beitrag mit Hinweisen wie veraltete Optionen zu ersetzen sind.

Ich bin mir bewusst, dass einigen Anwendern dieser scheinbare Paradigmenwechsel nicht gefallen wird. Aber absolute Kompatibilität sowohl im Ergebnis als auch im Code ist ohnehin längst eine Illusion. Mit der rasanten Entwicklung beim LaTeX-Kern und bei weit verbreiteten Paketen ist sie abhängig vom Dokument längst nicht mehr zu erreichen. Immerhin sind Anpassungen in der Regel aber leicht umsetzbar.

Für mich als Entwickler bringt die Codevereinfachung eine deutliche Verringerung des Aufwands. Angekündigt war der Schritt ohnehin schon lange.

Einen besonderen Dank möchte ich an dieser Stelle Marei Peischl aussprechen. Sie ist derzeit die treibende Kraft auf dem Weg zu PDF-Tagging mit KOMA-Script. Das hat mir den Anschub gegeben, die Umbauarbeiten weiter voran zu treiben. Wenn es nach ihr ginge, würde bereits diese Release `\DocumentMetadata` ermöglichen. Dass das kurzfristig und hoffentlich kurzzeitig verschoben wurde, liegt allein in meiner Verantwortung. Dasselbe gilt für alle Fehler, die sich in dieser Release befinden.

Markus Kohm, Neckarhausen im Oktober 2026

# Inhaltsverzeichnis

**Vorwort zu KOMA-Script 3.49.3** 7

**1. Einleitung** 19
1.1. Vorbemerkung . . . 19
1.2. Dokumentaufbau . . . 19
1.3. Die Geschichte von KOMA-Script . . . 21
1.4. Danksagung . . . 22
1.5. Rechtliches . . . 22
1.6. Installation . . . 22
1.7. Fehlermeldungen, Fragen, Probleme . . . 23
1.8. Weitere Informationen . . . 24

**Teil I: KOMA-Script für Autoren** 25

**2. Satzspiegelberechnung mit typearea.sty** 26
2.1. Grundlagen der Satzspiegelkonstruktion . . . 26
2.2. Satzspiegelkonstruktion durch Teilung . . . 29
2.3. Satzspiegelkonstruktion durch Kreisschlagen . . . 30
2.4. Frühe oder späte Optionenwahl . . . 30
2.5. Kompatibilität zu früheren Versionen von KOMA-Script . . . 32
2.6. Einstellung des Satzspiegels und der Seitenaufteilung . . . 33
2.7. Einstellung des Papierformats . . . 48
2.8. Tipps . . . 50

**3. Die Hauptklassen scrbook, scrreprt, scrartcl** 54
3.1. Frühe oder späte Optionenwahl . . . 55
3.2. Kompatibilität zu früheren Versionen von KOMA-Script . . . 56
3.3. Entwurfsmodus . . . 57
3.4. Seitenaufteilung . . . 58
3.5. Wahl der Schriftgröße für das Dokument . . . 59
3.6. Textauszeichnungen . . . 60
3.7. Dokumenttitel . . . 67
3.8. Zusammenfassung . . . 73
3.9. Inhaltsverzeichnis . . . 74
3.10. Absatzauszeichnung . . . 80
3.11. Erkennung von rechten und linken Seiten . . . 83

3.12. Kopf und Fuß bei vordefinierten Seitenstilen . . . 83
3.13. Vakatseiten . . . 90
3.14. Fußnoten . . . 93
3.15. Abgrenzung . . . 99
3.16. Gliederung . . . 100
3.17. Schlauer Spruch . . . 124
3.18. Listen . . . 126
3.19. Mathematik . . . 136
3.20. Gleitumgebungen für Tabellen und Abbildungen . . . 136
3.21. Randnotizen . . . 154
3.22. Anhang . . . 155
3.23. Literaturverzeichnis . . . 156
3.24. Stichwortverzeichnis . . . 159

**4. Briefe mit Klasse scrlttr2 oder Paket scrletter** 162
4.1. Frühe oder späte Optionenwahl . . . 162
4.2. Kompatibilität zu früheren Versionen von KOMA-Script . . . 163
4.3. Entwurfsmodus . . . 165
4.4. Seitenaufteilung . . . 165
4.5. Variablen . . . 166
4.6. Pseudolängen . . . 171
4.7. Genereller Aufbau eines Briefdokuments . . . 178
4.8. Wahl der Schriftgröße für das Dokument . . . 188
4.9. Textauszeichnungen . . . 190
4.10. Briefbogen . . . 196
4.10.1. Faltmarken . . . 196
4.10.2. Briefkopf . . . 201
4.10.3. Anschrift . . . 215
4.10.4. Absenderergänzung . . . 221
4.10.5. Geschäftszeilen . . . 223
4.10.6. Titel und Betreff . . . 228
4.10.7. Schlussgruß . . . 232
4.10.8. Briefbogenfuß . . . 234
4.11. Absatzauszeichnung . . . 238
4.12. Erkennung von rechten und linken Seiten . . . 239
4.13. Kopf und Fuß bei vordefinierten Seitenstilen . . . 240
4.14. Vakatseiten . . . 245
4.15. Fußnoten . . . 247
4.16. Listen . . . 250
4.17. Mathematik . . . 254

4.18. Gleitumgebungen für Tabellen und Abbildungen . . . 254
4.19. Randnotizen . . . 255
4.20. *Letter-Class-Option*-Dateien . . . 255
4.21. Adressdateien und Serienbriefe . . . 262

**5. Kopf- und Fußzeilen mit scrlayer-scrpage** 267
5.1. Frühe oder späte Optionenwahl . . . 267
5.2. Höhe von Kopf und Fuß . . . 269
5.3. Textauszeichnungen . . . 270
5.4. Verwendung vordefinierter Seitenstile . . . 273
5.5. Beeinflussung von Seitenstilen . . . 284

**6. Adressdateien mit scraddr erschließen** 296
6.1. Befehle . . . 296
6.2. Anwendung . . . 297
6.3. Paketoptionen für Warnungen . . . 299

**7. Grundlegende Fähigkeiten der KOMA-Script-Klassen mit Hilfe des Pakets scrextend anderen Klassen erschließen** 300
7.1. Frühe oder späte Optionenwahl . . . 300
7.2. Kompatibilität zu früheren Versionen von KOMA-Script . . . 302
7.3. Optionale, erweiterte Möglichkeiten . . . 302
7.4. Entwurfsmodus . . . 303
7.5. Wahl der Schriftgröße für das Dokument . . . 303
7.6. Textauszeichnungen . . . 304
7.7. Dokumenttitel . . . 306
7.8. Erkennung von rechten und linken Seiten . . . 311
7.9. Wahl eines vordefinierten Seitenstils . . . 311
7.10. Vakatseiten . . . 312
7.11. Fußnoten . . . 313
7.12. Schlauer Spruch . . . 316
7.13. Listen . . . 318
7.14. Randnotizen . . . 319

**8. Unterstützung für die Anwaltspraxis durch scrjura** 320

**9. Das KOMA-Script-Logo mit Paket scrlogo** 321

**Teil II:**
**KOMA-Script für fortgeschrittene Anwender und Experten** 322

**10. Grundlegende Funktionen im Paket scrbase** 323
10.1. Laden des Pakets . . . 323
10.2. Schlüssel als Eigenschaften von Familien und deren Mitgliedern . . . 323
10.3. Verzweigungen . . . 339
10.4. Definition sprachabhängiger Bezeichner . . . 343
10.5. Identifikation von KOMA-Script . . . 347
10.6. Erweiterungen des LaTeX-Kerns . . . 347
10.7. Erweiterungen der mathematischen Fähigkeiten von ε-TeX . . . 348
10.8. Mechanismus für mehrstufige Haken . . . 349
10.9. Obsolete Optionen und Anweisungen . . . 353

**11. Paketabhängigkeiten mit scrlfile beherrschen** 354
11.1. Aktionen vor und nach dem Laden . . . 355
11.2. Dateien beim Einlesen ersetzen . . . 360
11.3. Dateien gar nicht erst einlesen . . . 362

**12. Dateien mit scrwfile sparen und ersetzen** 365

**13. Verzeichnisse verwalten mit Hilfe von tocbasic** 366
13.1. Grundlegende Anweisungen . . . 366
13.2. Erzeugen eines Verzeichnisses . . . 370
13.3. Konfiguration von Verzeichniseinträgen . . . 378
13.4. Interne Anweisungen für Klassen- und Paketautoren . . . 394
13.5. Ein komplettes Beispiel . . . 397
13.6. Alles mit einer Anweisung . . . 399
13.7. Titel von (Gleit-)Umgebungen . . . 406
13.8. Nutzung von tocbasic mit Fremdklassen . . . 410
13.9. Obsolete Befehle . . . 417

**14. Definition von Ebenen und Seitenstilen mit scrlayer** 418
14.1. Frühe oder späte Optionenwahl . . . 418
14.2. Einige grundlegende Informationen . . . 419
14.3. Deklaration von Ebenen . . . 421
14.4. Deklaration und Verwaltung von Seitenstilen . . . 433
14.5. Höhe von Kopf und Fuß . . . 443
14.6. Beeinflussung von Seitenstilen . . . 444
14.7. Definition und Verwaltung von Schnittstellen für Endanwender . . . 450

**15. Zusätzliche Möglichkeiten von scrlayer-scrpage** 451
15.1. Beeinflussung von Seitenstilen . . . 451
15.2. Definition eigener Seitenstil-Paare . . . 454
15.3. Definition komplexer Seitenstile . . . 456
15.4. Definition einfacher Seitenstile mit dreigeteiltem Kopf und Fuß . . . 459
15.5. Das obsolete Erbe von scrpage2 . . . 459

**16. Notizspalten mit scrlayer-notecolumn** 461
16.1. Hinweise zum Entwicklungsstand . . . 461
16.2. Frühe oder späte Optionenwahl . . . 462
16.3. Textauszeichnungen . . . 463
16.4. Deklaration neuer Notizspalten . . . 465
16.5. Erstellen einer Notiz . . . 469
16.6. Erzwungene Ausgabe von Notizspalten . . . 474

**17. Zusätzliche Informationen zum Paket typearea.sty** 477
17.1. Experimentelle Möglichkeiten . . . 477
17.2. Anweisungen für Experten . . . 478
17.3. Lokale Einstellungen durch die Datei typearea.cfg . . . 480
17.4. Mehr oder weniger obsolete Optionen und Anweisungen . . . 480

**18. Zusätzliche Informationen zu den Hauptklassen und scrextend** 481
18.1. Ergänzungen zu Benutzeranweisungen . . . 481
18.2. Zusammenspiel von KOMA-Script und anderen Paketen . . . 481
18.3. Erkennung von KOMA-Script-Klassen . . . 481
18.4. Einträge ins Inhaltsverzeichnis . . . 482
18.5. Schrifteinstellungen . . . 484
18.6. Absatzmarkierung . . . 486
18.7. Zähler . . . 487
18.8. Gliederung . . . 487
18.9. Literaturverzeichnis . . . 510
18.10. Mehr oder weniger obsolete Optionen und Anweisungen . . . 512

**19. Zusätzliche Informationen zur Klasse scrlttr2 und Paket scrletter** 513
19.1. Variablen für fortgeschrittene Anwender . . . 513
19.2. Ergänzende Informationen zu den Seitenstilen . . . 515
19.3. lco-Dateien für fortgeschrittene Anwender . . . 515
19.4. Unterstützung verschiedener Sprachen . . . 519
19.5. Obsolete Anweisungen . . . 522

**Änderungsliste** 525

**Literaturverzeichnis** 538

**Index** 544

Allgemeiner Index . . . . . 544
Befehle, Umgebungen und Variablen . . . . . 547
Längen und Zähler . . . . . 559
Elemente mit der Möglichkeit zur Schriftumschaltung . . . . . 560
Dateien, Klassen und Pakete . . . . . 561
Klassen- und Paketoptionen . . . . . 563
Haken (*do-hooks*) . . . . . 567

# Abbildungsverzeichnis

2.1. Doppelseite mit der Rasterkonstruktion für die klassische Neunerteilung nach Abzug einer Bindekorrektur . . . 29

3.1. Parameter für die Darstellung der Fußnoten . . . 97
3.2. Beispiel: Bildbeschreibung daneben, unten . . . 143
3.3. Beispiel: Bildbeschreibung daneben, mittig . . . 144
3.4. Beispiel: Bildbeschreibung daneben, oben . . . 145
3.5. Beispiel: Bildunterschrift mit Voreinstellung . . . 147
3.6. Beispiel: Bildunterschrift mit teilweise hängendem Einzug . . . 147
3.7. Beispiel: Bildunterschrift mit hängendem Einzug und Umbruch . . . 147
3.8. Beispiel: Bildunterschrift mit Einzug in der zweiten Zeile . . . 147

4.1. Schematische Darstellung der wichtigsten Pseudolängen für den Briefbogen . . . 177
4.2. Genereller Aufbau eines Briefdokuments mit beliebig vielen einzelnen Briefen . . . 179
4.3. Genereller Aufbau eines einzelnen Briefes innerhalb eines Briefdokuments . . . 180
4.4. Beispiel: Brief mit Anschrift und Anrede . . . 183
4.5. Beispiel: Brief mit Anschrift, Anrede, Text und Grußfloskel . . . 185
4.6. Beispiel: Brief mit Anschrift, Anrede, Text, Grußfloskel und Postskriptum . . . 186
4.7. Beispiel: Brief mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum und Verteiler . . . 187
4.8. Beispiel: Brief mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen und Verteiler . . . 188
4.9. Beispiel: Brief mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen, Verteiler und ungesund großer Schrift . . . 191
4.10. Schematische Darstellung des Briefbogens mit den wichtigsten Anweisungen und Variablen für die skizzierten Elemente . . . 197
4.11. Beispiel: Brief mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen, Verteiler und Lochermarke . . . 199
4.12. Beispiel: Brief mit Absender, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen und Verteiler . . . 205
4.13. Beispiel: Brief mit Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke . . . 207
4.14. Beispiel: Brief mit erweitertem Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke; Standard- vs. erweiterter Briefkopf . . . 211
4.15. Beispiel: Brief mit erweitertem Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke; links- vs. rechtsbündiger Briefkopf . . . 212

4.16. Beispiel: Brief mit erweitertem Absender, Logo, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke; Absender links vs. rechts vs. zentriert . . . 214
4.17. Beispiel: Brief mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke . . . 223
4.18. Beispiel: Brief mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke . . . 227
4.19. Beispiel: Brief mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Betreff, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke . . . 231
4.20. Beispiel: Brief mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Betreff, Anrede, Text, Grußfloskel, geänderter Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke . . . 234
4.21. Beispiel: Brief mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Betreff, Anrede, Text, Grußfloskel, geänderter Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke mit `lco`-Datei . . . 259

5.1. Befehle zum Setzen des Seitenkopfes . . . 275
5.2. Befehle zum Setzen des Seitenfußes . . . 278

13.1. Illustration einiger Attribute des Verzeichniseintragsstils `dottedtocline` . . . 380
13.2. Illustration einiger Attribute des Verzeichniseintragsstils `largetocline` . . . 381
13.3. Illustration einiger Attribute des Verzeichniseintragsstils `tocline` . . . 381
13.4. Illustration einiger Attribute des Verzeichniseintragsstils `undottedtocline` am Beispiel einer Kapitelüberschrift . . . 382
13.6. Beispiel: Verwendung von `\captionaboveof` innerhalb einer fremden Gleitumgebung . . . 410
13.5. Beispiel: Ein Rechteck . . . 410
13.7. Beispiel für tocbasic ohne und mit Patch bei Verwendung von Klasse book . . . 415

16.1. Eine Ergebnisseite zu dem Beispiel aus dem Kapitel . . . 476

# Tabellenverzeichnis

2.1. Satzspiegelmaße in Abhängigkeit von DIV bei A4 . . . 35
2.2. DIV-Voreinstellungen für A4 . . . 36
2.3. Symbolische Werte für Option DIV und das *DIV*-Argument von \typearea . . . 38
2.4. Symbolische *BCOR*-Argumente für \typearea . . . 40
2.5. Standardwerte für einfache Schalter in KOMA-Script . . . 41
2.6. Ausgabetreiber für Option pagesize=*Ausgabetreiber* . . . 51

3.1. Klassengegenüberstellung . . . 54
3.2. Elemente, deren Schrift bei scrbook, scrreprt oder scrartcl mit \setkomafont und \addtokomafont verändert werden kann . . . 61
3.3. Schriftvoreinstellungen für die Elemente des Titels . . . 71
3.4. Der Haupttitel . . . 71
3.5. Mögliche Werte für Option toc . . . 76
3.6. Schriftvoreinstellungen für die Elemente des Inhaltsverzeichnisses . . . 79
3.7. Mögliche Werte für Option parskip . . . 81
3.8. Schriftvoreinstellungen für die Elemente des Seitenstils . . . 86
3.9. Makros zur Festlegung des Seitenstils besonderer Seiten . . . 87
3.10. Verfügbare Nummerierungsstile für Seitenzahlen . . . 89
3.11. Mögliche Werte für Option footnotes . . . 94
3.12. Mögliche Werte für Option open . . . 101
3.13. Mögliche Werte für Option headings . . . 103
3.14. Mögliche Werte für Option numbers . . . 106
3.15. Schriftvoreinstellungen für die Elemente der Gliederung bei scrbook und scrreprt . . . 111
3.16. Schriftvoreinstellungen für die Elemente des Spruchs . . . 124
3.17. Mögliche Werte für Option captions . . . 139
3.18. Schriftvoreinstellungen für die Elemente der Tabellen- oder Abbildungsunterschrift bzw. -überschrift . . . 141
3.19. Ausrichtungen für mehrzeilige Beschreibungen in Gleitumgebungen . . . 149
3.20. Mögliche Werte für Option listof . . . 152
3.21. Mögliche Werte für Option bibliography . . . 158
3.22. Mögliche Werte für Option index . . . 160

4.1. Von scrlttr2 und scrletter unterstützte Variablen . . . 166
4.2. Von scrlttr2 und scrletter verwendete Pseudolängen . . . 172
4.3. Elemente, deren Schrift bei Briefen mit \setkomafont und \addtokomafont verändert werden kann . . . 192

4.4. Kombinierbare Werte für die Konfiguration der Faltmarken mit der Option `foldmarks` . . . 198
4.5. Mögliche Werte für Option `fromalign` zur Platzierung des Absenders auf dem Briefbogen . . . 203
4.6. Mögliche Werte für Option `fromrule` . . . 204
4.7. Vordefinierte Bezeichnungen der Variablen für die Absenderangaben im Briefkopf . . . 209
4.8. Vordefinierte Bezeichnungen und Inhalte der Trennzeichen für die Absenderangaben im Briefkopf ohne Option `symbolicnumbers` . . . 210
4.9. Mögliche Werte für Option `addrfield` . . . 216
4.10. Schriftvoreinstellungen für die Elemente des Anschriftfensters . . . 217
4.11. Mögliche Werte für Option `priority` . . . 217
4.12. Mögliche Werte für Option `locfield` . . . 221
4.13. Mögliche Werte für Option `refline` . . . 224
4.14. Vordefinierte Bezeichnungen der Variablen der Geschäftszeile . . . 225
4.15. Schriftvoreinstellungen für die Elemente der Geschäftszeile . . . 226
4.16. Vordefinierte Bezeichnungen der Variablen für den Betreff . . . 229
4.17. Mögliche Werte für Option `subject` . . . 230
4.18. Mögliche Werte für Option `pagenumber` . . . 242
4.19. Vordefinierte `lco`-Dateien . . . 259

5.1. Elemente, deren Schrift bei scrlayer-scrpage mit `\setkomafont` und `\addtokomafont` verändert werden kann, einschließlich der jeweiligen Voreinstellung . . . 271
5.2. Mögliche Werte für Option `markcase` . . . 288
5.3. Symbolische Werte für Option `headwidth` und `footwidth` . . . 294

7.1. Optional verfügbare, erweiterte Möglichkeiten von scrextend . . . 303

10.1. Überblick über übliche sprachabhängige Begriffe . . . 345

13.1. Attribute für die vordefinierten Verzeichniseintragsstile von tocbasic . . . 383
13.2. Optionen für die Anweisung `\DeclareNewTOC` . . . 400
13.3. Gegenüberstellung von Beispielumgebung `remarkbox` und Umgebung `figure` . . . 405
13.4. Beispiel: Maße des Rechtecks aus Abbildung 13.5 . . . 410
13.5. für tocbasic verfügbare Patches . . . 412

14.1. Optionen für die Beschreibung von Seiten-Ebenen mit ihrer jeweiligen Bedeutung als Ebenen-Eigenschaft . . . 423
14.2. Optionen und gleichnamige Haken für Ebenen-Seitenstile (in der Reihenfolge ihrer Abarbeitung) . . . 437

15.1. Von scrlayer-scrpage zu einem Seitenstil definierte Ebenen . . . . . 458

16.1. Mögliche Einstellungen für die Deklaration von Notizspalten . . . . . 466

18.1. Stil unabhängige Eigenschaften bei der Konfiguration von Gliederungsbefehlen 489
18.2. Eigenschaften des Stils `part` bei der Konfiguration von Gliederungsbefehlen . . . 490
18.3. Eigenschaften des Stils `chapter` bei der Konfiguration von Gliederungsbefehlen 491
18.4. Eigenschaften des Stils `section` bei der Konfiguration von Gliederungsbefehlen 492
18.5. Voreinstellungen für die Kapitelüberschriften von scrbook und scrreprt in Abhängigkeit von Option `headings` . . . . . 495
18.6. Voreinstellungen für die Formatierung der Überschriften von scrbook und scrreprt 496

19.1. Voreinstellungen für die sprachabhängigen Begriffe in Briefen . . . . . 523
19.2. Sprachabhängige Ausgabeformate für das Datum . . . . . 524

# 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 auf Fehler stoßen.

## 1.1. Vorbemerkung

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 der Autor gegenüber sieht, besteht jedoch darin, dass eine Anleitung für alle möglichen Zielgruppen benötigt wird. Ich habe mich bemüht, eine Anleitung zu erstellen, die für den Informatiker gleichermaßen geeignet ist wie für die Sekretärin des Fischhändlers. Ich habe mich bemüht, obwohl es sich dabei eigentlich um ein unmögliches Unterfangen handelt. Ergebnis sind zahlreiche Kompromisse. Ich bitte jedoch, die Problematik bei eventuellen Beschwerden zu berücksichtigen und bei der Verbesserung der derzeitigen Lösung zu helfen.

Trotz des Umfangs der Anleitung bitte ich 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.md` 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 Informationen 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 [DGS+12] nahegelegt. Wiedereinsteigern aus der Zeit von LaTeX 2.09 sei zumindest [Tea26] und ebenfalls [Tea23] 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, sich einen groben Überblick darüber zu verschaffen und es bei Problemen ebenso wie diese Anleitung 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. Hierzu 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 einige grundlegende 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 ver-

wendet. 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{`*`Umgebung`*`}`, 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 *`Umgebung`* der Anweisung `\begin`.

## 1.3. Die Geschichte von KOMA-Script

Anfang der 1990er 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 $2_\varepsilon$. 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 $2_\varepsilon$ 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 Ergebnis. 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 ursprüngliche Dokumentation integriert wurden. Irgendwann wurde es damit auch Zeit für eine komplett überarbeitete Anleitung.

## 1.4. Danksagung

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. Dieses Mal danke ich Elke, Jana, Ben und Edoardo stellvertretend für Beta-Test und Kritik. Ich hoffe, ihr macht damit noch weiter.

Ganz besonderen Dank bin ich den Gründern und den Mitgliedern 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])m in der Usenet-Gruppe `de.comp.text.tex` und den vielen LATEX-Foren im Internet, die mir so manche Antwort auf Fragen zu KOMA-Script abnehmen.

Mein Dank gilt aber 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 besonders bedanke ich mich noch einmal bei dem äußerst großzügigen Spender, der mich mit dem bisher und vermutlich für alle Zeiten größten Einzelbetrag für die bisher geleistete Arbeit an KOMA-Script bedacht hat.

## 1.5. Rechtliches

KOMA-Script steht unter der LATEX Project Public Licence. Eine nicht offizielle deutsche Übersetzung ist Bestandteil des KOMA-Script-Pakets. In allen Zweifelsfällen gilt im deutschsprachigen Raum der Text `lppl-de.txt`, während in allen anderen Ländern der Text `lppl.txt` anzuwenden ist.

Für die Korrektheit der Anleitung, Teile der Anleitung oder einer anderen in diesem Paket enthaltenen Dokumentation wird keine Gewähr übernommen.

## 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.

Daneben gibt es auf [KDP] seit einiger Zeit Installationspakete von Zwischenversionen von KOMA-Script für die wichtigsten Distributionen. Für deren Installation ist die dortige Anleitung zu beachten.

## 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 folgende Checkliste abarbeiten:

- 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.

- Falls Sie nicht die aktuelle KOMA-Script-Version verwenden, sollten Sie unbedingt die Installation einer aktuellen Version ins Auge gefasst werden. Tritt das Problem mit der aktuellen Version von KOMA-Script dann nicht mehr auf, haben Sie die Lösung bereits gefunden.

- 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 oder durch Beispieldateien aus dem Paket mwe [Sch18] ersetzt werden. Das Paket können Sie auch verwenden, um Fülltext zu erzeugen. 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 und deren Version. 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 erzeugen Sie für das Problem einen neuen Eintrag im Ticketsystem unter https://sf.net/p/koma-script/tickets. Falls dies für Sie nicht möglich ist, können Sie ihre Meldung ersatzweise an komascript@gmx.info verschicken.

Häufig werden Sie eine Frage zu KOMA-Script oder im Zusammenhang mit KOMA-Script lieber öffentlich, beispielsweise in de.comp.text.tex oder einem Forum 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.

Bitte beachten Sie, dass typografisch nicht optimale Voreinstellungen keine Fehler darstellen. Aus Gründen der Kompatibilität werden Voreinstellungen nach Möglichkeit auch in neuen KOMA-Script-Versionen beibehalten. Darüber hinaus ist Typografie auch eine Frage der Sprache und Kultur. Die Voreinstellungen von KOMA-Script stellen daher zwangsläufig einen Kompromiss dar.

## 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 322 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 [Ume10]), 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.

## 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 ein Verfahren, das mit verschiedenen Diagonalen und deren Schnittpunkten arbeitet. Das gewünschte Ergebnis 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:

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

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 [TF11]) 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 Wörter 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 darstellt. 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.

Soweit die Grundlagen. In den folgenden beiden Abschnitten werden die von KOMA-Script angebotenen Konstruktionen im Detail vorgestellt.

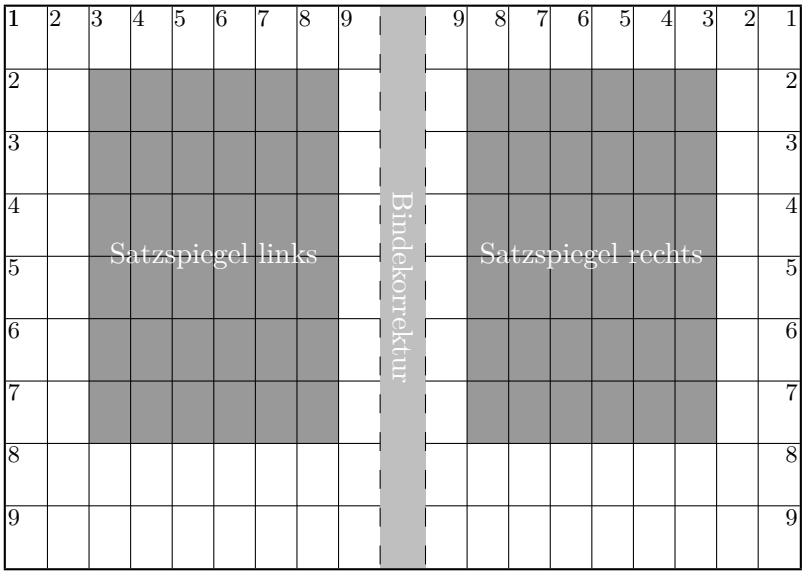



Abbildung 2.1.: Doppelseite mit der Rasterkonstruktion für die klassische Neunerteilung nach Abzug einer Bindekorrektur

## 2.2. Satzspiegelkonstruktion durch Teilung

Der einfachste Weg, um zu erreichen, dass der Textbereich dasselbe Seitenverhältnis 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 36 zu entnehmen sind. Bei Verzicht auf Bindekorrektur, wenn also $BCOR = 0\,\text{pt}$ gilt, ergeben sich in etwa die Satzspiegelmaße aus Tabelle 2.1, Seite 35.

Neben den voreingestellten Werten kann man *BCOR* und *DIV* direkt beim Laden des Pakets als Option angeben (siehe Abschnitt 2.6 ab Seite 33). Zusätzlich existiert ein Befehl, mit dem man einen Satzspiegel explizit berechnen kann und dem man die beiden Werte als Parameter übergibt (siehe ebenfalls Abschnitt 2.6, Seite 40).

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 (siehe ebenfalls Abschnitt 2.6, Seite 36).

## 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 einer speziellen, symbolischen Angabe 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. Im Sinne der Abgeschlossenheit der Kapitel ist dieser Abschnitt nahezu identisch in mehreren Kapiteln zu finden.

\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 [Tea23] oder jeder LaTeX-Einführung, beispielsweise [DGS+12], beschrieben.

v3.00

Bei Verwendung einer KOMA-Script-Klasse sollten beim dann ohnehin überflüssigen Laden des Pakets typearea oder scrbase keine Optionen angegeben werden. Diese Klassen laden die beiden Pakete nämlich bereits ohne Optionen und LaTeX meldet einen Fehler, wenn eines der Pakete anschließend mit Optionen geladen wird.

Das Setzen der Optionen mit `\documentclass` hat übrigens einen entscheidenden Nachteil: Anweisungen, Längen, Zähler und ähnliches können darin leicht zerbrechen. So führt die Verwendung einer LaTeX-Länge im Wert einer Option bei dieser Anweisung bei vielen Nicht-KOMA-Script-Klassen zu einer Fehlermeldung, noch bevor der Wert an ein KOMA-Script-Paket übergeben wird, es also die Kontrolle darüber übernehmen könnte. Wertzuweisungen mit LaTeX-Längen oder LaTeX-Zählern sollten daher nie per `\documentclass`, sondern mit den nachfolgend dokumentierten Anweisungen `\KOMAoptions` oder `\KOMAoption` vorgenommen werden.

\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. Mit der Anweisung `\KOMAoptions` kann man wie bei `\documentclass` oder `\usepackage` 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 `\KOMAoption` der einen *Option* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *Werteliste* durch Komma voneinander getrennt.

Soll ein *Wert* ein Gleichheitszeichen oder ein Komma enthalten, 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 finden fortgeschrittene Anwender in Abschnitt 10.2 ab Seite 329.

Mit \KOMAoptions oder \KOMAoption gesetzte Optionen erreichen sowohl die KOMA-Script-Klasse als auch alle bereits geladenen KOMA-Script-Pakete, die diese Optionen kennen. Ist eine Option oder ein Wert insgesamt unbekannt, so wird die Option einschließlich des Wertes von scrbase als fehlerhaft gemeldet.

## 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 führen aber Verbesserungen und Korrekturen am Paket zu Änderungen im Verhalten, inbesondere beim Umbruch. Dies ist jedoch manchmal eher unerwünscht.

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

v3.01b Seit Version 3.01b besteht bei typearea die Wahl, ob eine Quelldatei, soweit irgend möglich, auch zukünftig bei einem LaTeX-Lauf zu exakt demselben Ergebnis führen soll oder ob sie 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 rückwirkende Kompatibilität verzichtet. Wird die Option ohne Wertangabe verwendet,
v3.01a 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 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 in Abschnitt 2.4 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}
```

mit `BCOR=8.25mm` als Option für typearea oder

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

zur Angabe von `BCOR=8.25mm` als globale Option.

Bei Verwendung einer KOMA-Script-Klasse sollte 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, theoretischen 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 wird in LaTeX mit der Anweisung \usepackage{bookman} aktiviert und 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 eines Werts von 12 sind Sie nach gründlichem Studium dieses Kapitels einschließlich der weiterführenden Abschnitte

Tabelle 2.1.: Satzspiegelmaße in Abhängigkeit von DIV bei A4 ohne Berücksichtigung von \topskip oder BCOR

| | 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)

überzeugt, dass der 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:

```
\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. Einschließlich Biegefalz werden dafür 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 voreingestellte Option a4paper konnte dabei entfallen.

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

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

\KOMAoptions oder \KOMAoption nach dem Laden des Pakets geändert, so werden Satzspiegel und Ränder 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. Solche festen Werte haben allerdings den Nachteil, dass sie die Laufweite der verwendeten Schrift nicht berücksichtigen. Das kann bei A4 und recht schmalen Schriften sehr rasch zu unangenehm hoher Zeichenzahl je Zeile führen. Siehe hierzu die Überlegungen in Abschnitt 2.1. 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 DIV=calc anstelle 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 17.3). 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 anstelle 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 Ränder automatisch neu berechnet.

```
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 unter 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 der Anweisung \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. |

Falls die letzte Satzspiegelberechnung vor der Verwendung von DIV=last oder DIV=current mit Hilfe von \areaset stattgefunden hat, wird übrigens auch die Neuberechnung wieder mit \areaset durchgeführt. Sie entspricht dann \areaset[current]{\textwidth}{\textheight}.

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 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 [TF11]) wie folgt aus:

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

An dieser Stelle erscheint es mir angebracht, darauf hinzuweisen, dass der Zeilenabstand für die Titelseite wieder auf den normalen Wert zurückgesetzt werden sollte und außerdem auch die Verzeichnisse mit dem normalen Zeilenabstand gesetzt werden sollten.

**Beispiel:** Ein vollständiges Beispiel wäre also:

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

Siehe hierzu auch die Anmerkungen in Abschnitt 2.8. Die Anweisung `\AfterTOCHead` wird in Teil II, Kapitel 13 auf Seite 374 vorgestellt.

Außerdem sei darauf hingewiesen, dass Änderungen am Zeilenabstand auch Auswirkungen auf Kopf und Fuß der Seite haben können. Dies kann sich beispielsweise bei Verwendung von scrlayer-scrpage auswirken und man muss dann selbst entscheiden, ob man dort lieber den normalen Durchschuss oder den veränderten haben will. Siehe dazu auch Option `singlespacing` in Kapitel 14 auf Seite 440.

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

| |
|---|
| `current`<br>Satzspiegelberechnung mit dem aktuell gültigen `BCOR`-Wert erneut durchführen. |

Bitte beachten Sie, dass diese Optionen bei Verwendung mit `\KOMAoptions` oder `\KOMAoption` eine automatische Neuberechnung von Satzspiegel und Rändern auslösen.

`\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 (siehe Tabelle 2.4). Beim Setzen der Option `BCOR` wird für *DIV* hingegen der symbolische Wert `last` verwendet. Wollen Sie, dass Satzspiegel und Ränder stattdessen mit dem symbolischen Wert `current` für *DIV* neu berechnet werden, so können Sie direkt `\typearea[`*BCOR*`]{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. Bei `\KOMAoptions{DIV=`*Faktor*`,BCOR=`*Korrektur*`}` werden hingegen Ränder und Satzspiegel zunächst in Folge von Option `DIV` und dann zusätzlich durch Option `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. Auch wenn am LaTeX-Kern inzwischen wieder mehr Änderungen vorgenommen werden, ist die Wahrscheinlichkeit hoch, dass dies in zukünftigen Versionen von LaTeX $2_\varepsilon$ 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}` oder `\KOMAoptions{DIV=last}` beziehungsweise `\typearea[current]{last}` für die Neuberechnung des Satzspiegels und der Ränder recht häufig benötigt werden, gibt es dafür die abkürzende Anweisung `\recalctypearea`.

v3.00

`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.

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

| Wert | Bedeutung |
|---|---|
| `true` | aktiviert die Option |
| `on` | aktiviert die Option |
| `yes` | aktiviert die Option |
| `false` | deaktiviert die Option |
| `off` | deaktiviert die Option |
| `no` | deaktiviert die Option |

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 auch noch der Wert `semi` angegeben werden. Dieser Wert `semi` führt zu doppelseitigem Satz mit einseitigen Rändern und einseitigen, also nicht

v3.12 alternierenden Marginalien. Eine eventuelle Bindekorrektur (siehe Option BCOR, Seite 33) wird jedoch ab KOMA-Script Version 3.12 wie beim doppelseitigen Satz auf Seiten mit ungerader Nummer dem linken Rand und auf Seiten mit gerader Nummer dem rechten Rand zugeschlagen. Wird auf Kompatibilität zu einer früheren Version zurückgeschaltet (siehe Abschnitt 2.5, Seite 32), so ist die Bindekorrektur dagegen auch bei `twoside=semi` immer Teil des linken Randes.

Die Option kann als Klassenoption bei `\documentclass`, als Paketoption bei `\usepackage` oder nach dem Laden von typearea per `\KOMAoptions` oder `\KOMAoption` gesetzt werden. Eine Verwendung dieser Option nach dem Laden von typearea führt automatisch zur Neuberechnung des Satzspiegels mit `\recalctypearea` (siehe Seite 40). 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 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 als Klassenoption bei `\documentclass`, als Paketoption bei `\usepackage` oder nach dem Laden von typearea per `\KOMAoptions` oder `\KOMAoption` gesetzt werden.

Eine Verwendung nach dem Laden von typearea führt automatisch zur Neuberechnung des Satzspiegels mittels \recalctypearea (siehe Seite 40).

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

Bisher wurde zwar erklärt, wie die Satzspiegelkonstruktion funktioniert und in welchem Verhältnis einerseits die Ränder zueinander stehen, andererseits der Textkörper zur Seite steht, aber eine entscheidende Frage blieb ausgeklammert: Was genau ist *der Rand*?

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. 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 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 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 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 aber 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, Seite 41 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 37) oder die Anweisung `\recalctypearea` (siehe Seite 40).

`mpinclude=`*Ein-Aus-Wert*

v2.8q

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 (siehe beispielsweise Befehl `\marginpar` in [DGS+12] oder Abschnitt 3.21) 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, Seite 41 angegeben sind. Man kann die Option auch ohne Wertzuweisung verwenden. In diesem Fall wird `true` als *Ein-Aus-Wert* verwendet.

v3.00

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: Die Anpassung von \textwidth und \marginparwidth nach der Berechnung des Satzspiegels. Siehe dazu \AfterCalculatingTypearea in Abschnitt 17.2, Seite 480.

Bitte beachten Sie unbedingt, dass diese Option bei Verwendung einer der KOMA-Script-Klassen als Klassenoption oder per \KOMAoptions beziehungsweise oder \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 37) oder die Anweisung \recalctypearea (siehe Seite 40).

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

v3.00

Optionen **headlines** und **headheight**.

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 3.12) 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 Ränder entscheidend ist. Außerdem sollten LaTeX-Längen wie `\baselineskip` keinesfalls im optionalen Argument von `\documentclass` oder `\usepackage` verwendet werden.

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 37) oder die Anweisung `\recalctypearea` (siehe Seite 40).

```
footlines=Zeilenanzahl
footheight=Höhe
\footheight
```

v3.12

Wie schon für den Kopf fehlt aber noch die Angabe, wie hoch der Fuß sein soll. Hierzu dienen die Optionen `footlines` und `footheight`. Allerdings ist die Höhe des Fußes im Gegensatz zur Höhe des Kopfes keine Länge des LaTeX-Kerns selbst. Daher definiert typearea zur Einführung eine neue Länge `\footheight`, falls diese noch nicht existiert. Ob diese dann auch beispielsweise von Klassen und Paketen für die Gestaltung von Kopf und Fuß verwendet wird, hängt von den verwendeten Klassen und Paketen ab. Das KOMA-Script-Paket scrlayer-scrpage berücksichtigt `\footheight` und arbeitet somit aktiv mit typearea zusammen. Die KOMA-Script-Klassen berücksichtigen `\footheight` hingegen nicht, da sie ohne Paketunterstützung nur Seitenstile mit einzeiligen Seitenfüßen anbieten.

Die Option `footlines` setzt man vergleichbar zu `headlines` auf die Anzahl der Fußzeilen. Normalerweise arbeitet das typearea-Paket mit 1,25 Fußzeilen. Dieser Wert stellt einen Kompromiss dar. Zum einen ist er groß genug, um auch für eine über- und unterstrichene Fußzeile (siehe Abschnitt 3.12) Platz zu bieten, zum anderen ist er klein genug, um das Randgewicht nicht zu stark zu verändern, wenn mit einer einfachen Fußzeile ohne Trennlinien gearbeitet wird. Damit ist der voreingestellte Wert in den meisten Standardfällen ein guter Wert. In einigen Fällen will oder muss man aber die Fußhöhe genauer den tatsächlichen Erfordernissen anpassen.

**Beispiel:** Angenommen, im Fuß soll eine zweizeilige Copyright-Angabe gesetzt werden. Zwar gibt es in LaTeX selbst keinen Test, ob der für den Fuß vorgesehene Platz dafür genügend Raum bietet, die Überschreitung der vorgesehenen Höhe resultiert aber wahrscheinlich in einer unausgeglichenen Verteilung von Satzspiegeln und Rändern. Außerdem führt beispielsweise das Paket scrlayer-scrpage, mit dem ein solcher Fußinhalt gesetzt werden könnte, durchaus eine entsprechende Überprüfung durch und meldet gegebenenfalls auch Überschreitungen. Daher ist es sinnvoll, die benötigte größere Fußhöhe bereits bei der Berechnung des Satzspiegels anzugeben:

```
\documentclass[a4paper]{article}
\usepackage[footlines=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[footlines=2.1]{scrartcl}
```

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

In einigen Fällen ist es nützlich, wenn man die Fußhöhe nicht in Zeilen, sondern direkt als Längenwert angeben kann. Dies ist mit Hilfe der alternativ verwendbaren Option `footheight` 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 Ränder entscheidend ist. Außerdem sollten LaTeX-Längen wie `\baselineskip` keinesfalls im optionalen Argument von `\documentclass` oder `\usepackage` verwendet werden.

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 37) oder die Anweisung `\recalctypearea` (siehe Seite 40).

`\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`, `headheight`, `footlines` und `footheight` bleiben in diesem Fall jedoch unberücksichtigt! Siehe dazu die weiterführenden Informationen zu `\areaset` auf Seite 478 in Abschnitt 17.1.

Die Voreinstellung für *BCOR* ist 0 pt. Soll hingegen die aktuelle, beispielsweise per Option `BCOR` eingestellte Bindekorrektur erhalten bleiben, sollte man den symbolischen Wert `current` als optionales Argument verwenden.

**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 mit quadratischem Textbereich der Seitenlänge 15 cm und einem Binderand von 1 cm gesetzt werden, so ist Folgendes möglich:

```
\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, wenn `DIV=last` verwendet wird und der Satzspiegel zuletzt per `\areaset` eingestellt wurde.

## 2.7. Einstellung des Papierformats

Das Papierformat ist ein Grundmerkmal eines Dokuments. Wie bereits bei der Vorstellung der Satzspiegelkonstruktionen (siehe Abschnitt 2.1 bis Abschnitt 2.3 ab Seite 26) 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`*
`paper=`*`Ausrichtung`*

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, darf auch die nachfolgend beschriebene Option `pagesize` nicht deaktiviert sein.

v3.01a Zusätzlich kann das *`Format`* auch in der Form *`Breite`*`:`*`Höhe`* beziehungsweise *`Höhe`*`:`*`Breite`* angegeben werden. Welcher Wert die *`Höhe`* und welcher die *`Breite`* ist, richtet sich nach der Ausrichtung des Papiers. Mit `paper=landscape` oder `paper=seascape` ist der kleinere Wert die *`Höhe`* und der größere Wert die *`Breite`*. Mit `paper=portrait` ist dagegen der kleinere Wert die *`Breite`* und der größere Wert die *`Höhe`*.

Es wird darauf hingewiesen, dass bis Version 3.01a der erste Wert immer die *`Höhe`* und der zweite Wert die *`Breite`* war. Dagegen war von Version 3.01b bis Version 3.21a der erste Wert immer die *`Breite`* und der zweite Wert immer die *`Höhe`*. Dies ist insbesondere dann zu beachten, wenn mit einer entsprechenden Kompatibilitätseinstellung (siehe Option `version`, Abschnitt 2.5, Seite 32) 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=landscape,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 37) oder die Anweisung `\recalctypearea` (siehe Seite 40).

`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 Paket 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[1] oder ESC/P2[2] beziehungsweise ESC/P-R[3] 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

[1] PCL ist eine Familie von Druckersprachen, die HP für seine Tinten- und Laserdrucker verwendet.
[2] ESC/P2 ist die Druckersprache, die EPSON für seine 24-Nadel- und ältere Tinten- oder Laserdrucker benutzt.
[3] ESC/P-R ist die Druckersprache, die EPSON aktuell für Tinten- und Laserdrucker benutzt.

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 Verwendung von pdfTEX, luaTEX, XƎTEX oder VTEX werden deren Papierformat-Längen entsprechend gesetzt.

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

v3.17

**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.

## 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

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

`auto`
: Falls die pdfTEX-spezifischen Register `\pdfpagewidth` und `\pdfpageheight` oder die luaTEX-spezifischen Register `\pagewidth` und `\pageheight` vorhanden sind, wird der Ausgabetreiber `pdftex` aktiviert. Zusätzlich wird auch der Ausgabetreiber `dvips` verwendet. Diese Einstellung ist grundsätzlich auch für XETEX geeignet.

`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`, `luatex`
: v3.20 Die Papiergröße wird über die pdfTEX-spezifischen Register `\pdfpagewidth` und `\pdfpageheight` oder die luaTEX-spezifischen Register `\pagewidth` und `\pageheight` gesetzt. Dies ist auch jederzeit innerhalb des Dokuments problemlos möglich.

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. Häufig wird 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 [Ume10]) oder verändern Sie die Satzspiegelparameter von LaTeX selbst. Die von typearea ermittelten Werte finden Sie in der log-Datei Ihres Dokuments. Mit Hilfe von Option usegeometry, die Sie in Teil II finden, kann außerdem die Zusammenarbeit von typearea und geometry verbessert werden. 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 $2_\varepsilon$ im Jahre 1994 obsolet. Verwenden Sie schlimmstenfalls den Befehl \linespread. Ich empfehle das Paket setspace, das nicht zu KOMA-Script gehört (siehe

[TF11]). 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. Näheres dazu finden Sie bei der Erklärung zu `DIV=current`.

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, anstelle 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{article}` gegen `\documentclass{scrartcl}`. 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.

Es sei an dieser Stelle jedoch nicht verschwiegen, dass einige Paketautoren ihre Pakete auf Basis der Implementierung und sogar von internem Code der Standardklassen entwickeln und dabei keine Rücksicht auf komplett unabhängige Entwicklungen wie die KOMA-Script-Klassen nehmen. In solchen Fällen kann es beim ersten LaTeX-Lauf nach der Umstellung durchaus zu Fehlermeldungen oder zusäzlichen Warnungen kommen. Meist lassen sich diese auf einfache Weise beheben. Oftmals können dazu die erweiterten Möglichkeiten von KOMA-Script genutzt werden, wodurch das problematische Paket dann vollständig entfällt. Manchmal kann auch das in [Koh23a] dokumentierte Paket scrhack Abhilfe schaffen. Auch der Ersatz von veralteten Paketen durch aktuelle Nachfolger kann zur Beseitigung derartiger Probleme beitragen. Teilweise geben sogar die KOMA-Script-Klassen durch entsprechende Warnungen Hilfestellung bei der Lösung von Inkompatibilitäten.

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

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

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

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. Falls Sie also Abschnitt 2.4 bereits gelesen und verstanden haben, können Sie auf Seite 56 mit Abschnitt 3.2 fortfahren.

```
\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 [Tea23] oder jeder LaTeX-Einführung, beispielsweise [DGS+12], beschrieben.

v3.00

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

Gegenüber der nachfolgend vorgestellten Schnittstelle zu Einstellungen von Optionen hat `\documentclass` einen Nachteil, der unbedingt zu beachten ist: Anweisungen, Längen, Zähler und ähnliches können darin leicht zerbrechen. So führt die Verwendung einer LaTeX-Länge im Wert einer Option bei dieser Anweisung bei vielen Nicht-KOMA-Script-Klassen zu einer Fehlermeldung. Wertzuweisungen mit LaTeX-Längen oder LaTeX-Zählern sollten daher nie per `\documentclass`, sondern mit den nachfolgend dokumentierten Anweisungen `\KOMAoptions` oder `\KOMAoption` vorgenommen werden.

```
\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. Mit der Anweisung `\KOMAoptions` kann man wie bei `\documentclass` oder `\usepackage` 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 `\KOMAoption` der einen *Option* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *Werteliste* durch Komma voneinander getrennt.

Soll ein *Wert* ein Gleichheitszeichen oder ein Komma enthalten, 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`«. Siehe dazu Teil II, Abschnitt 10.2, ab Seite 329.

Mit `\KOMAoptions` oder `\KOMAoption` gesetzte Optionen erreichen sowohl die KOMA-Script-Klasse als auch alle bereits geladenen KOMA-Script-Pakete, die diese Optionen kennen. Ist eine Option oder ein Wert insgesamt unbekannt, so wird die Option einschließlich des Wertes von scrbase als fehlerhaft gemeldet.

## 3.2. Kompatibilität zu früheren Versionen von KOMA-Script

Es gilt sinngemäß, was in Abschnitt 2.5 geschrieben wurde. Falls Sie also Abschnitt 2.5 bereits gelesen und verstanden haben, können Sie in Abschnitt 3.3 auf Seite 57 fortfahren.

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 führen aber Verbesserungen und Korrekturen an der Klasse zu Änderungen im Verhalten, inbesondere beim Umbruch. Dies ist jedoch manchmal eher unerwünscht.

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

v3.00

Seit Version 2.96a besteht bei KOMA-Script die Wahl, ob eine Quelldatei, soweit irgend möglich, auch zukünftig bei einem LaTeX-Lauf zu exakt demselben Ergebnis führen soll oder ob sie jeweils entsprechend der Anpassungen der neusten Version der Klasse 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 rückwirkende Kompatibilität verzichtet. Wird die Option ohne Wertangabe verwendet, so wird ebenfalls `last` angenommen. Dies entspricht auch der Voreinstellung, solange keine obsolete Option verwendet wird.

v3.01a

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 der Klasse nicht mehr verändert werden kann. Das Setzen mit `\KOMAoptions` oder `\KOMAoption` ist daher nicht vorgesehen.

## 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`*
`overfullrule=`*`Ein-Aus-Wert`*

v3.00

Mit Option `draft` 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 41 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 [Sch13].

v3.25

Da Option `draft` bei verschiedenen Paketen zu allerlei unerwünschten Effekten führen kann, bietet KOMA-Script die Möglichkeit, die Markierung für überlange Zeilen auch über Option

overfullrule zu steuern. Auch hier gilt, dass bei aktivierter Option die Markierung angezeigt wird.

## 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 selbstständig. 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, ab Seite 55.

Einige Einstellungen der KOMA-Script-Klassen haben 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 26 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, wenn bei der Konstruktion des Satzspiegels einige Grundregeln befolgt wurden. 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 die Möglichkeit, diesen *vertikalen Ausgleich* automatisch durchzuführen.

Wird über die Option twoside (siehe Abschnitt 2.6, Seite 40) doppelseitiger oder über Option twocolumn (siehe Seite 41) zweispaltiger Satz angefordert, so wird der vertikale Ausgleich

v3.17

ebenfalls eingeschaltet. Das gilt jedoch bei einer Kompatibilitätseinstellung zu einer KOMA-Script-Version vor 3.17 (siehe Abschnitt 3.2, Seite 56, Option version) nicht, wenn die Option über \KOMAoption oder \KOMAoptions geändert wird.

Man kann den vertikalen Ausgleich 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.

`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 `\normalsize` und die davon abhängigen Schriftgrößen der Befehle `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\large`, `\Large`, `\LARGE`, `\huge` und `\Huge` 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 2.6, Seite 40). Wird diese Neuberechnung jedoch vorgenommen, so erfolgt sie auf Basis der jeweils gültigen Grundschriftgröße. Die Auswirkungen des Wechsels der Grundschriftgröße auf zusätzlich geladene Pakete oder die verwendete Klasse sind von diesen Paketen und der Klasse abhängig. Es können also Fehler auftreten, die nicht als Fehler von KOMA-Script angesehen werden. Auch die KOMA-Script-Klassen passen nicht alle Längen an eine nachträglich vorgenommene Anderung der Grundschriftgröße an.

Diese Option sollte keinesfalls als Ersatz für `\fontsize` (siehe [Tea05]) missverstanden werden. Sie sollte auch nicht anstelle einer der von der Grundschrift abhängigen Schriftgrößenanweisungen, `\tiny` bis `\Huge`, verwendet werden! Die Verwendung innerhalb eines Absatzes ist aus diesem Grund auch explizit verboten!

Voreingestellt ist bei scrbook, scrreprt und scrartcl `fontsize=11pt`. Demgegenüber ist 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 [DGS+12], [Tea23] und [Tea05] zu entnehmen.

```
sfdefaults=Ein-Aus-Wert
\maybesffamily
\textmaybesf{Text}
```

v3.39

Die Anweisungen `\maybesffamily` und `\textmaybesf` verhalten sich je nach Einstellung von Option `sfdefaults` unterschiedlich. Als *Ein-Aus-Wert* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5 verwendet werden. Nur bei aktivierter Option führt `\maybesffamily` zu `\sffamily` und `\textmaybesf` zu `\textsf`. Dies ist auch die Voreinstellung. KOMA-Script selbst verwendet `\maybesffamily` in der Voreinstellung der Elemente `descriptionlabel`, `disposition` und `dictum`. `\maybesffamily` darf also auch als Teil der *Befehle* der nachfolgend erklärten Anweisungen `\setkomafont` und `\addtokomafont` verwendet werden.

```
\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 Schriftattribute umgeschaltet werden. In der Regel werden dies Befehle wie `\rmfamily`, `\sffamily`, `\ttfamily`, `\upshape`, `\itshape`, `\slshape`, `\scshape`, `\mdseries`, `\bfseries`, `\normalfont` oder einer der Befehle `\Huge`, `\huge`, `\LARGE`, `\Large`, `\large`, `\normalsize`, `\small`, `\footnotesize`, `\scriptsize` und `\tiny` sein. Die Erklärung zu diesen Befehlen entnehmen Sie bitte [DGS+12], [Tea23] oder [Tea05]. Auch Farbumschaltungen wie `\normalcolor` sind möglich (siehe [Car17] 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 Ele-

mente 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` verwendet werden. 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

`author`

v3.12 Autorangaben im Haupttitel des Dokuments mit `\maketitle`, also das Argument von `\author` (siehe Abschnitt 3.7, Seite 70)

`caption`

Text einer Abbildungs- oder Tabellenunter- oder -überschrift (siehe Abschnitt 3.20, Seite 141)

`captionlabel`

Label einer Abbildungs- oder Tabellenunter- oder -überschrift; Anwendung erfolgt nach dem Element `caption` (siehe Abschnitt 3.20, Seite 141)

`chapter`

Überschrift der Ebene `\chapter` (siehe Abschnitt 3.16, Seite 107)

`chapterentry`

Inhaltsverzeichniseintrag der Ebene `\chapter` (siehe Abschnitt 3.9, Seite 78)

`chapterentrydots`

v3.15 optionale Verbindungspunkte in Inhaltsverzeichniseinträgen der Ebene `\chapter` abweichend von Element `chapterentry`, `\normalfont` und `\normalsize` (siehe Abschnitt 3.9, Seite 78)

`chapterentrypagenumber`

Seitenzahl des Inhaltsverzeichniseintrags der Ebene `\chapter` abweichend vom Element `chapterentry` (siehe Abschnitt 3.9, Seite 78)

...

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

`chapterprefix`
: Kapitelnummernzeile sowohl bei Einstellung `chapterprefix=true` als auch `appendixprefix=true` (siehe Abschnitt 3.16, Seite 101)

`date`
v3.12
: Datum im Haupttitel des Dokuments mit `\maketitle`, also das Argument von `\date` (siehe Abschnitt 3.7, Seite 70)

`dedication`
v3.12
: Widmung nach dem Haupttitel des Dokuments mit `\maketitle`, also das Argument von `\dedication` (siehe Abschnitt 3.7, Seite 73)

`descriptionlabel`
: Label, also das optionale Argument der `\item`-Anweisung, in einer `description`-Umgebung (siehe Abschnitt 3.18, Seite 129)

`dictum`
: mit `\dictum` gesetzter schlauer Spruch (siehe Abschnitt 3.17, Seite 124)

`dictumauthor`
: Urheber eines schlauen Spruchs; Anwendung erfolgt nach dem Element `dictum` (siehe Abschnitt 3.17, Seite 124)

`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 100)

`enumeratelabel`
v3.44
: Grundeinstellung für die voreingestellten Aufzählungsnummern der Umgebung `enumerate` (siehe Abschnitt 3.18, Seite 128)

`footnote`
: Marke und Text einer Fußnote (siehe Abschnitt 3.14, Seite 94)

`footnotelabel`
: Marke einer Fußnote; Anwendung erfolgt nach dem Element `footnote` (siehe Abschnitt 3.14, Seite 94)

...

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

`footnotereference`
: Referenzierung der Fußnotenmarke im Text (siehe Abschnitt 3.14, Seite 94)

`footnoterule`
v3.07
: Linie über dem Fußnotenapparat (siehe Abschnitt 3.14, Seite 98)

`itemizelabel`
v3.33
: Grundeinstellung für die voreingestellten Aufzählungszeichen der Umgebung `itemize` (siehe Abschnitt 3.18, Seite 126)

`labelenumi`
v3.44
: Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumi` (siehe Abschnitt 3.18, Seite 128)

`labelenumii`
v3.44
: Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumii` (siehe Abschnitt 3.18, Seite 128)

`labelenumiii`
v3.44
: Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumiii` (siehe Abschnitt 3.18, Seite 128)

`labelenumiv`
v3.44
: Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumiv` (siehe Abschnitt 3.18, Seite 128)

`labelinglabel`
: Label, also das optionale Argument der `\item`-Anweisung, und Trennzeichen, also das optionale Argument der `labeling`-Umgebung, in einer `labeling`-Umgebung (siehe Abschnitt 3.18, Seite 130)

`labelingseparator`
: Trennzeichen, also das optionale Argument der `labeling`-Umgebung, in einer `labeling`-Umgebung; Anwendung erfolgt nach dem Element `labelinglabel` (siehe Abschnitt 3.18, Seite 130)

`labelitemi`
v3.33
: Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemi` (siehe Abschnitt 3.18, Seite 126)

...

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

`labelitemii`

v3.33 Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemii` (siehe Abschnitt 3.18, Seite 126)

`labelitemiii`

v3.33 Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemiii` (siehe Abschnitt 3.18, Seite 126)

`labelitemiv`

v3.33 Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemiv` (siehe Abschnitt 3.18, Seite 126)

`minisec`

mit `\minisec` gesetzte Überschrift (siehe Abschnitt 3.16 ab Seite 114)

`pagefoot`

wird nur verwendet, wenn das Paket scrlayer-scrpage geladen ist (siehe Kapitel 5, Seite 278)

`pagehead`

alternative Bezeichnung für `pageheadfoot`, solange scrlayer-scrpage nicht geladen ist (siehe auch Abschnitt 5.4, Seite 274)

`pageheadfoot`

Seitenkopf und Seitenfuß bei allen von KOMA-Script definierten Seitenstilen (siehe Abschnitt 3.12 ab Seite 83)

`pagenumber`

Seitenzahl im Kopf oder Fuß der Seite (siehe Abschnitt 3.12, Seite 85)

`pagination`

alternative Bezeichnung für `pagenumber`

`paragraph`

Überschrift der Ebene `\paragraph` (siehe Abschnitt 3.16, Seite 107)

`part`

Überschrift der Ebene `\part`, jedoch ohne die Zeile mit der Nummer des Teils (siehe Abschnitt 3.16, Seite 107)

`partentry`

Inhaltsverzeichniseintrag der Ebene `\part` (siehe Abschnitt 3.9, Seite 78)

...

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

`partentrypagenumber`
: Seitenzahl des Inhaltsverzeichniseintrags der Ebene `\part` abweichend vom Element `partentry` (siehe Abschnitt 3.9, Seite 78)

`partnumber`
: Zeile mit der Nummer des Teils in Überschrift der Ebene `\part` (siehe Abschnitt 3.16, Seite 107)

`publishers`
v3.12
: Verlagsangabe im Haupttitel des Dokuments mit `\maketitle`, also das Argument von `\publishers` (siehe Abschnitt 3.7, Seite 70)

`section`
: Überschrift der Ebene `\section` (siehe Abschnitt 3.16, Seite 107)

`sectionentry`
: Inhaltsverzeichniseintrag der Ebene `\section` (nur bei scrartcl verfügbar, siehe Abschnitt 3.9, Seite 78)

`sectionentrydots`
v3.15
: optionale Verbindungspunkte in Inhaltsverzeichniseinträgen der Ebene `\section` abweichend vom Element `sectionentry`, `\normalfont` und `\normalsize` (nur bei scrartcl verfügbar, siehe Abschnitt 3.9, Seite 78)

`sectionentrypagenumber`
: Seitenzahl des Inhaltsverzeichniseintrags der Ebene `\section` abweichend vom Element `sectionentry` (nur bei scrartcl verfügbar, siehe Abschnitt 3.9, Seite 78)

`sectioning`
: alternative Bezeichnung für `disposition`

`subject`
: Typisierung des Dokuments, also das Argument von `\subject` auf der Haupttitelseite mit `\maketitle` (siehe Abschnitt 3.7, Seite 70)

`subparagraph`
: Überschrift der Ebene `\subparagraph` (siehe Abschnitt 3.16, Seite 107)

`subsection`
: Überschrift der Ebene `\subsection` (siehe Abschnitt 3.16, Seite 107)

...

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

`subsubsection`
: Überschrift der Ebene `\subsubsection` (siehe Abschnitt 3.16, Seite 107)

`subtitle`
: Untertitel des Dokuments, also das Argument von `\subtitle` auf der Haupttitelseite mit `\maketitle` (siehe Abschnitt 3.7, Seite 70)

`title`
: Haupttitel des Dokuments, also das Argument von `\title` bei Verwendung von `\maketitle` (bezüglich der Größe des Haupttitels siehe die ergänzenden Bemerkungen im Text von Abschnitt 3.7 ab Seite 70)

`titlehead`
v3.12
: Kopf über dem Haupttitel des Dokuments, also das Argument von `\titlehead` mit `\maketitle` (siehe Abschnitt 3.7, Seite 70)

```
\usefontofkomafont{Element}
\useencodingofkomafont{Element}
\usesizeofkomafont{Element}
\usefamilyofkomafont{Element}
\useseriesofkomafont{Element}
\useshapeofkomafont{Element}
```

v3.12 Manchmal werden in der Schrifteinstellung eines Elements auch Dinge vorgenommen, die mit der Schrift eigentlich gar nichts zu tun haben, obwohl dies ausdrücklich nicht empfohlen wird. Soll dann nur die Schrifteinstellung, aber keine dieser zusätzlichen Einstellungen ausgeführt werden, so kann statt `\usekomafont` die Anweisung `\usefontofkomafont` verwendet werden. Diese Anweisung übernimmt nur die Schriftgröße und den Grundlinienabstand, die Codierung (engl. *encoding*), die Familie (engl. *family*), die Strichstärke oder Ausprägung (engl. *font series*) und die Form oder Ausrichtung (engl. *font shape*).

Mit den übrigen Anweisungen können auch einzelne Schriftattribute übernommen werden. Dabei übernimmt `\usesizeofkomafont` sowohl die Schriftgröße als auch den Grundlinienabstand.

Diese Befehle sollten jedoch nicht als Legitimation dafür verstanden werden, in die Schrifteinstellungen der Elemente beliebige Anweisungen einzufügen. Das kann nämlich sehr schnell zu Fehlern führen (siehe Abschnitt 18.5, Seite 485).

## 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 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
titlepage=firstiscover
\coverpagetopmargin
\coverpageleftmargin
\coverpagerightmargin
\coverpagebottommargin
```

v3.00 Mit dieser Option wird ausgewählt, ob für die mit `\maketitle` (siehe Seite 68) 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 41 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. Die zusätzlichen Elemente finden sie auf den nachfolgenden Seiten.

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.

v3.12 Mit der dritten Möglichkeit, `titlepage=firstiscover`, werden nicht nur Titelseiten aktiviert. Es wird auch dafür gesorgt, dass die erste von `\maketitle` ausgegebene Titelseite, also entweder der Schmutztitel oder der Haupttitel, als Umschlagseite ausgegeben wird. Jede andere Einstellung für die Option `titlepage` hebt diese Einstellung wieder auf. Die Ränder dieser Umschlagseite werden über `\coverpagetopmargin` (oberer Rand), `\coverpageleftmargin` (linker Rand), `\coverpagerightmargin` (rechter Rand) und natürlich `\coverpagebottommargin` (unterer Rand) bestimmt. Die Voreinstellungen sind von den Längen `\topmargin` und `\evensidemargin` abhängig und können mit `\renewcommand` geändert 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}
```

\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. Außerdem kennt \maketitle bei KOMA-Script noch ein optionales numerisches 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 87) 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 [BB13]) auch nicht empfehlenswert, diese Anweisungen in der Dokumentpräambel vor `\begin{document}` zu verwenden. Beispieldokumente finden Sie bei den weiteren Befehlen dieses Abschnitts.

`\extratitle{`*Schmutztitel*`}`
`\frontispiece{`*Frontispiz*`}`

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.

v3.25

Die Rückseite des Schmutztitels ist das *Frontispiz*. 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}
```

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. Für die Ausgabe wird die Schrift des gleichnamigen Elements verwendet (siehe Tabelle 3.4, Seite 71).

Die *Typisierung* wird unmittelbar über dem *Titel* in der Schrift des gleichnamigen Elements ausgegeben.

v2.8p Der *Titel* wird in einer sehr großen Schrift gesetzt. Dabei finden abgesehen von der Größe Schriftumschaltungen für das Element title Anwendung (siehe Tabelle 3.4, Seite 71).

v2.97c Der *Untertitel* steht knapp unter dem Titel in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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. Die Ausgabe erfolgt in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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. Die Ausgabe erfolgt in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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. Die Ausgabe erfolgt in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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

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

| Element-Name | Voreinstellung |
|---|---|
| author | \Large |
| date | \Large |
| dedication | \Large |
| publishers | \Large |
| subject | \normalfont\normalcolor\bfseries\Large |
| subtitle | \usekomafont{title}\large |
| title | \usekomafont{disposition} |
| titlehead | |

innerhalb des Arguments einer der übrigen Anweisungen, beispielsweise im Argument *Autor* der Anweisung \author, zu verwenden ist.

v3.12 Für die Ausgabe der Titelelemente kann die Schrift mit Hilfe der Befehle \setkomafont und \addtokomafont (siehe Abschnitt 3.6, Seite 60) eingestellt werden. Die Voreinstellungen sind Tabelle 3.3 zu entnehmen.

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\\
```

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

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

```
    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 sogenannten 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, das 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.

Seit KOMA-Script 3.12 kann man die erste von `\maketitle` ausgegebene Titelseite alternativ aber auch als Umschlagseite formatieren lassen. Dabei ändern sich nur die für diese Seite verwendeten Ränder (siehe Option `titlepage=firstiscover` auf Seite 67).

```
\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*`}`

v3.12

KOMA-Script bietet eine eigene Widmungsseite. Diese Widmung wird zentriert und in der Voreinstellung mit etwas größerer Schrift gesetzt. Die genaue Schrifteinstellung für das Element `dedication`, die Tabelle 3.3, Seite 71 zu entnehmen ist, kann über die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 60) verändert werden.

Die Rückseite ist 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 Partnerin oder Ihrem Partner 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}
```

## 3.8. Zusammenfassung

Insbesondere bei Artikeln findet man zwischen Titel und Inhaltsverzeichnis oft eine Zusammenfassung. Bei Verwendung eines Titelkopfes ist die Zusammenfassung in der Regel rechts und links eingezogen. Bei Verwendung von Titelseiten wird die Zusammenfassung eher als Kapitel oder Abschnitt gesetzt.

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

scrreprt, 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 41 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 das 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. Siehe dazu die in Abschnitt 3.16, ab Seite 112 dokumentierten Befehle `\chapter*`. `\addchap` und `\addchap*`.

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

scrartcl, scrreprt

Einige LaTeX-Klassen bieten eine spezielle Umgebung für die 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 ab Seite 112 die Anweisungen `\chapter*` und `\addchap` oder `\addchap*`.

Wird ein Titelkopf (siehe Option `titlepage`, Abschnitt 3.7, Seite 67) verwendet, so wird die Zusammenfassung intern mit Hilfe einer `quotation`-Umgebung (siehe Abschnitt 3.18, Seite 133) 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).

Neben den in diesem Abschnitt dokumentierten Möglichkeiten hat auch der mit \DeclareTOCStyleEntry gewählte und konfigurierte Eintragsstil des Pakets tocbasic (siehe Seite 379) maßgeblichen Einfluss auf die Darstellung des Inhaltsverzeichnisses. Entsprechend können sich auch die in Abschnitt 18.8 ab Seite 487 dokumentierten Befehle \DeclareSectionCommand, \ProvideSectionCommand, \DeclareNewSectionCommand und \RedeclareSectionCommand auf das Inhaltsverzeichnis auswirken.

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

v3.00 ins Inhaltsverzeichnis aufzunehmen, werden mit der *Einstellung* listof beide Verzeichnisse gemeinsam ins Inhaltsverzeichnis aufgenommen. Dabei werden auch Verzeichnisse berücksichtigt, die mit Hilfe des float-Pakets ab Version 1.2e (siehe [Lin01]) oder floatrow (siehe [Lap08]) 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

v3.18 auch dieser Eintrag grundsätzlich nicht nummeriert. Eine Abweichung von diesem Grundsatz wird von KOMA-Script trotz Bedenken des Autors mit toc=indexnumbered ebenfalls 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.

`chapterentrywithdots`, `chapterentrydotfill`
v3.15
: Bei den Kapiteleinträgen der Klassen scrbook und scrreprt werden Text und Seitenzahl ebenfalls durch eine punktierte Linie miteinander verbunden.

`chapterentrywithoutdots`, `chapterentryfill`
v3.15
: Bei den Kapiteleinträgen der Klassen scrbook und scrreprt werden Text und Seitenzahl nicht durch eine punktierte Linie miteinander verbunden. Dies entspricht der Voreinstellung.

`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. Dies entspricht der Voreinstellung.

...

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

`indenttextentries`, `indentunnumbered`, `numberline`

v3.12

Die Eigenschaft `numberline` (siehe Abschnitt 13.2, Seite 375) wird für das Inhaltsverzeichnis gesetzt. Dadurch werden nicht nummerierte Einträge linksbündig mit dem Text von nummerierten Einträgen gleicher Ebene gesetzt.

`index`, `idx`

Das Stichwortverzeichnis erhält einen Eintrag im Inhaltsverzeichnis, ohne dass es nummeriert wird.

`indexnumbered`, `idxnumbered`, `numberedindex`, `numberedidx`

v3.18

Das Stichwortverzeichnis erhält einen Eintrag im Inhaltsverzeichnis und wird nummeriert.

`leftaligntextentries`, `leftalignunnumbered`, `nonumberline`

v3.12

Die Eigenschaft `numberline` (siehe Abschnitt 13.2, Seite 375) wird für das Inhaltsverzeichnis gelöscht. Dadurch werden nicht nummerierte Einträge linksbündig mit der Nummer von nummerierten Einträgen gleicher Ebene gesetzt. Dies entspricht der Voreinstellung.

`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. Dies entspricht der Voreinstellung.

`noindex`, `noidx`

Das Stichwortverzeichnis erhält keinen Eintrag im Inhaltsverzeichnis. Dies entspricht der Voreinstellung.

`nolistof`

Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten keinen Eintrag im Inhaltsverzeichnis. Dies entspricht der Voreinstellung.

...

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

| | |
|---|---|
| `sectionentrywithdots`, `sectionentrydotfill` | |
| v3.15 | Bei den Abschnittseinträgen der Klasse scrartcl werden Text und Seitenzahl ebenfalls durch eine punktierte Linie miteinander verbunden. |
| `sectionentrywithoutdots`, `sectionentryfill` | |
| v3.15 | Bei den Abschnittseinträgen der Klasse scrartcl werden Text und Seitenzahl nicht durch eine punktierte Linie miteinander verbunden. Dies entspricht der Voreinstellung. |

`chapterentrydots=`*`Ein-Aus-Wert`*
`sectionentrydots=`*`Ein-Aus-Wert`*

scrbook, scrreprt, scrartcl

Diese Optionen legen fest, ob im Inhaltsverzeichnis bei den Klassen scrbook und scrreprt für die Kapitelebene beziehungsweise bei der Klasse scrartcl für die Abschnittsebene wie bei den tieferen Ebenen auch eine punktierte Verbindungslinie zwischen dem Text des Eintrags und der zugehörigen Seitenzahl verwendet werden soll. Als *`Ein-Aus-Wert`* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 41 verwendet werden. Voreingestellt ist für beide Optionen `false`, wodurch statt einer punktierten Linie lediglich ein Abstand verwendet wird.

Wird die punktierte Linie verwendet, so kann deren Schrift über das Element `chapterentrydots` oder `sectionentrydots` verändert werden (siehe `\setkomafont` und `\addtokomafont`, Abschnitt 3.6, Seite 60, sowie Tabelle 3.2, Seite 61). Die Voreinstellungen der Elemente sind Tabelle 3.6 auf Seite 79 zu entnehmen. Es ist zu beachten, dass die Pünktchen nur sauber untereinander stehen, wenn die Schrift aller Pünktchen identisch ist. Aus diesen Grund ist die Ausgangschrift immer `\normalfont\normalsize`. Von `chapterentry` oder `sectionentry` bleibt für die Pünktchen daher nur die Farbe erhalten.

`\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 `\chapter` bei scrbook und scrreprt beziehungsweise `\section` bei scrartcl sowie der Gliederungsebene `\part` wird nicht eingerückt. Gleichzeitig befinden sich zwischen dem Text der Gliederungsebene und der Seitenzahl in der Voreinstellung keine Pünktchen. Die typografischen Gründe dafür liegen in der normalerweise anderen Schriftart sowie der erv3.15 wünschten Hervorhebung. Dies kann jedoch mit den zuvor dokumentierten Optionen geändert

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

| Element | Voreinstellung |
|---|---|
| partentry | \usekomafont{disposition}\large |
| partentrypagenumber | |
| chapterentry | \usekomafont{disposition} |
| chapterentrydots | |
| chapterentrypagenumber | |
| sectionentry | \usekomafont{disposition} |
| sectionentrydots | |
| sectionentrypagenumber | |

werden. Das Inhaltsverzeichnis dieser Anleitung ist mit den Voreinstellungen gesetzt und dient als Beispiel.

v2.97c Die Schrift der oberen Inhaltsverzeichniseinträge ist 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` (scrbook und scrreprt) beziehungsweise `sectionentrypagenumber` (scrartcl) eingestellt werden (siehe auch `\setkomafont` und v3.15 `\addtokomafont` in Abschnitt 3.6, Seite 60, sowie Tabelle 3.2, Seite 61). Werden je nach Klasse für die Kapitel- beziehungsweise Abschnittseinträge ebenfalls punktierte Verbindungslinien zu den Seitenzahlen über Option `toc`, `chapterentrydots` oder `sectionentrydots` verwendet, so kann deren Schrift über die Elemente `chapterentrydots` und `sectionentrydots` verändert werden. Die Voreinstellungen der Elemente sind Tabelle 3.6 zu entnehmen.

```
tocdepth
\parttocdepth
\sectiontocdepth
\subsectiontocdepth
\subsubsectiontocdepth
\paragraphtocdepth
\subparagraphtocdepth
```

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 wird 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 Inhaltsverv3.15 zeichnis erfolgen sollen. Dies ist übrigens bei den Standardklassen ganz genauso. Im Unterschied zu den Standardklassen muss sich bei KOMA-Script aber niemand diese Zuordnung merken. KOMA-Script definiert für jede Gliederungsebene eine Anweisung `\`*`Ebene`*`tocdepth` mit dem entsprechenden Wert, die für Zuweisungen an `tocdepth` verwendet werden kann.

scrartcl

Bitten beachten Sie, dass die Werte für die Zähler `tocdepth` und `secnumdepth` (siehe Abschnitt 3.16, Seite 121) bei scrartcl für `\part` nicht übereinstimmen. Dies wurde aus Gründen der Kompatibilität von der Standardklasse article übernommen. Daher sollte beispielsweise die Anweisung `\partnumdepth` auch nicht zum Setzen von `tocdepth` verwendet werden.

**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}{\subsectiontocdepth}
```

Sie setzen den Zähler `tocdepth` daher auf den in Anweisung `\subsectiontocdepth` gespeicherten Wert. Dass dies normalerweise der Wert 2 ist, müssen Sie sich also gar nicht merken.

Wollen Sie stattdessen nur, dass eine Ebene weniger in das Inhaltsverzeichnis eingetragen wird als normalerweise, können Sie auch einfach vom voreingestellten Wert des Zählers `tocdepth` eins abziehen:

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

Den Wert, den Sie zu `tocdepth` addieren oder davon subtrahieren müssen, können Sie nach mindestens zwei LaTeX-Läufen einfach 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 anstelle des voreingestellten Absatzeinzugs gefordert. Die KOMA-Script-Klassen bieten 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, freigelassen. Mit dem Pluszeichen wird am Zeilenende mindestens ein Drittel und mit dem Stern mindestens ein Viertel einer normalen Zeile freigelassen. 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 am Ende des Absatzes sichtbar.

Neben den sich so ergebenden acht Kombinationen ist es noch möglich, als *`Methode`* die Werte für einfache Schalter aus Tabelle 2.5, Seite 41 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. Absatzenden werden nicht gekennzeichnet. |
| `full`, `true`, `on`, `yes` | Absätze werden durch einen vertikalen Abstand von einer Zeile gekennzeichnet, Absatzenden durch einen Leerraum von mind. einem 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. einem Geviert (1 em) der normalen Schrift am Ende der letzten Zeile.

`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 ungerader oder gerader Nummer. In dieser Anleitung ist vereinfachend auch 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`*`}`

v3.28

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 aktuell 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 ungerader 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 18.1, Seite 481 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 41 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 Paket typearea ebenfalls auf `headsepline` reagiert und automatisch die Paketoption `headinclude` mit gleichem Wert ausführt. Entsprechendes gilt auch für die Fußlinie. Im Gegensatz zu `headsepline` wirkt sich die Option `footsepline` auch beim Seitenstil `plain` aus, da `plain` eine Seitenzahl im Fuß ausgibt.

Die Optionen führen selbst keine automatische Neuberechnung des Satzspiegels aus. Zur Neuberechnung des Satzspiegels siehe Option `DIV` mit den Werten `last` oder `current` (siehe Seite 37) oder die Anweisung `\recalctypearea` (siehe Seite 40) in Kapitel 2.

Das Paket scrlayer-scrpage (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

scrbook, scrreprt

scrartcl

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.

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 84) 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 87).

**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 mit Einfügen einer Vakatseite herbeiführen können, und soll die Änderung erst ab der neuen Seite gelten, so ist ein `\cleardoublepage` davor nützlich.

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, dass die Umschaltung zwischen automatischen und manuellen Kolumnentiteln bei Verwendung von scrlayer-scrpage nicht mehr über den Seitenstil, sondern mit speziellen Anweisungen erfolgt. Die beiden Seitenstile **headings** und **myheadings** sollten nicht zusammen mit diesem Paket verwendet werden.

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 60). 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

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

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

dem Paket scrlayer-scrpage (siehe Kapitel 5, Seite 278) einen Seitenstil definiert, bei dem auch der Fuß Text enthält.

Die Voreinstellungen sind in Tabelle 3.8 zu finden.

**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 61).

Es ist an dieser Stelle nicht möglich, Versalien für die automatischen Kolumnentitel zu erzwingen. Wenn Sie dies wünschen, müssen Sie beispielsweise `\MakeMarkcase` entsprechend umdefinieren. Es wird jedoch empfohlen, in solchen Fällen das Paket scrlayer-scrpage zu verwenden (siehe Kapitel 5, Seite 287).

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 scrlayer-scrpage (siehe Kapitel 5), sondern beispielsweise das Paket fancyhdr (siehe [vO04]) 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 scrlayer-scrpage sorgt selbst für maximale Kompatibilität, solange beispielsweise für die Seitenzahl nicht direkt `\thepage`, sondern das dafür vorgesehene `\pagemark` verwendet wird.

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) |

`\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 Pakets scrlayer-scrpage steht außerdem die Anweisung `\markleft` zur Verfügung.

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

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

scrbook, scrreprt

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.

**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 84 erfahren haben, ist der Seitenstil `empty` genau das, was in diesem Beispiel verlangt wird.

Angenommen, Sie haben mit einem der Pakete scrlayer (siehe Abschnitt 14.4) oder Paket scrlayer-scrpage (siehe Abschnitt 15.2) einen eigenen Seitenstil mit Namen `chapter` für Kapitelanfangsseiten erstellt. Um ihn 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.

Mit der Anweisung `\AfterTOCHead` aus Abschnitt 13.2, Seite 374 wird die Lösung noch einfacher:

```
\AfterTOCHead[toc]{%
  \thispagestyle{empty}%
  \pagestyle{empty}}
```

Hierbei wird die Tatsache ausgenutzt. dass bei mehreren `\thispagestyle`-Anweisungen auf derselben Seite immer die letzte gewinnt.

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

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 |

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

verwendet, sollte in Abschnitt 18.1 auf Seite 481 Näheres über die Hintergründe zu `\rightmark` nachlesen. Auch die Ausführungen zu `\rightfirstmark` ab Seite Seite 451 in Kapitel 15, Teil II können hierzu wichtige Informationen liefern.

`\pagenumbering{`*`Nummerierungsstil`*`}`

Diese Anweisung funktioniert bei KOMA-Script in der gleichen Weise wie bei den Standardklassen. Genau genommen handelt es sich dabei 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 besser `\cleardoubleoddpage` diese Seite erst beendet werden. Mögliche Angaben für den *`Nummerierungsstil`* sind Tabelle 3.10 zu entnehmen.

Der Aufruf von `\pagenumbering` setzt immer die Seitenzahl zurück. Die aktuelle Seite bekommt also die Nummer 1 im gewählten *`Nummerierungsstil`*. Damit bei doppelseitigen Dokumenten diese neue Seite auch wirklich auf eine gerade Seite folgt, die linke Seite also nicht etwa fehlt, sollte man daher vor `\pagenumbering` immer `\cleardoubleoddpage` einfügen. Näheres zu einer dabei gegebenenfalls eingefügten Vakatseite erfahren Sie im nächsten Abschnitt.

Lassen Sie mich noch ein Wort zu einem häufigen Fehler verlieren, den man in diversen Vorlagen findet, die im Internet kursieren. Wenn Sie – natürlich ohne den Eingangskommentar – auf Zeilen wie:

```
% Achtung dieses Beispiel enthält Fehler!
% Beachten Sie bitte die Erklärungen im Text!
\tableofcontents
\pagenumbering{arabic}
\setcounter{page}{1}
```

stoßen, so ist das ein untrügliches Zeichen dafür, dass der Ersteller dieser Vorlage obige Ausführungen nicht gelesen oder nicht verstanden hat. Da `\tableofcontents` das Inhaltsverzeichnis zwar ausgibt, aber die letzte Seite nicht automatisch beendet, wird bereits für diese letzte Seite des Inhaltsverzeichnisses die Seitennummerierung umgeschaltet. Sie erhält damit die arabische Seitenzahl 1. Es fehlt also `\cleardoubleoddpage` vor `\pagenumbering`. Ebenso ist die letzte Zeile mit dem

Setzen der Seitennummerierung auf 1 überflüssig, da dies bereits von \pagenumbering erledigt wird.

Teilweise findet man – natürlich ohne den Eingangskommentar – auch:

```
% Achtung dieses Beispiel enthält Fehler!
% Beachten Sie bitte die Erklärungen im Text!
\tableofcontents
\pagebreak
\pagenumbering{arabic}
\setcounter{page}{1}
```

Hier hat der Ersteller versucht, das Problem mit der letzten Seite des Inhaltsverzeichnisses mit Hilfe von \pagebreak zu lösen. Diese Lösung ist aber leider auch nicht viel besser. Hier wird die letzte Seite des Inhaltsverzeichnisses auf die nächste Seite umbrochen. Damit werden bei einem doppelseitigen Dokument unter Umständen die Einträge auf der letzten Seite mit einem erhöhten vertikalen Abstand gesetzt (siehe \flushbottom auf Seite 58). \pagebreak ist hier eindeutig die falsche Anweisung.

Aber auch \newpage oder \clearpage würden bei einem doppelseitigen Dokument nicht genügen. Hätte die letzte Seite des Inhaltsverzeichnisses beispielsweise die römische Nummer vii, so würde auf die römisch nummerierte rechte Seite nun unmittelbar die arabisch nummerierte rechte Seite 1 folgen. Eine linke Seite zwischen diesen beiden Seiten würde im Dokument fehlen, was beim späteren Druck erhebliche Probleme bereiten könnte.

Mein Rat: Vermeiden Sie die Verwendung von Vorlagen, die bereits in solch einfachen Dingen Fehler enthalten. Korrekt wäre übrigens:

```
\tableofcontents
\cleardoubleoddpage
\pagenumbering{arabic}
```

scrartcl Das gilt auch, wenn mit scrartcl eine Klasse verwendet wird, bei der üblicherweise nach dem Inhaltsverzeichnis keine neue Seite begonnen wird. Schaltet man die Seitennummerierung um, so muss eine neue rechte Seite begonnen werden. Wollen Sie eine solche Umschaltung nicht, so sollten Sie den Nummerierungsstil der Seiten über das gesamte Dokument konsequent durchhalten, ohne ihn zwischendurch zu ändern. Bei Artikeln oder anderen kurzen Dokumenten ist das generell zu empfehlen.

scrbook Einfacher wird die Änderung des Nummerierungsstils bei Verwendung von scrbook. Dort werden Sie durch die beiden Anweisungen \frontmatter und \mainmatter bei der am häufigsten verwendeten Umschaltung unterstützt. Nähere Informationen entnehmen Sie bitte Abschnitt 3.15, ab Seite 99.

## 3.13. Vakatseiten

Vakatseiten sind Seiten, die beim Satz eines Dokuments absichtlich leer bleiben. 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 vorherige Kapitel ebenfalls auf einer rechten Seite endet, eine leere linke Seite eingefügt werden.

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

v3.00 Mit Hilfe dieser Option kann man den *`Seitenstil`* der Vakatseite bestimmen, die bei Bedarf von den Anweisungen `\cleardoublepage`, `\cleardoubleoddpage` oder `\cleardoubleevenpage` eingefügt wird, um bis zur gewünschten Seite zu umbrechen. Als *`Seitenstil`* sind dabei alle bereits definierten Seitenstile (siehe Abschnitt 3.12 ab Seite 83 und Kapitel 5 ab Seite 267) 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.

v3.00 Ab Version 3.00 werden in der Voreinstellung entsprechend der typografischen Gepflogenheiten Vakatseiten mit dem Seitenstil `empty` erzeugt.

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

```
\KOMAoptions{cleardoublepage=plain}
```

Näheres zum Seitenstil `plain` ist in Abschnitt 3.12, Seite 85 zu finden.

```
\clearpage
\cleardoublepage
\cleardoublepageusingstyle{Seitenstil}
\cleardoubleemptypage
\cleardoubleplainpage
\cleardoublestandardpage
\cleardoubleoddpage
\cleardoubleoddpageusingstyle{Seitenstil}
\cleardoubleoddemptypage
\cleardoubleoddplainpage
\cleardoubleoddstandardpage
\cleardoubleevenpage
\cleardoubleevenpageusingstyle{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 2.6, 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 `\cleardoubleoddpageusingstyle` 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 für die ein Argument erwartende Anweisung `\cleardoubleevenpageusingstyle`.

Die Arbeitsweise der Anweisungen `\cleardoublestandardpage`, `\cleardoubleemptypage`,

\cleardoubleplainpage und der ein Argument erwartenden Anweisung \cleardoublepageusingstyle ist ebenso wie die Standard-Anweisung \cleardoublepage von der in Abschnitt 3.16, Seite 100 erklärten Option open 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.

An der gewünschten Stelle schreiben Sie daher:

```
\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 [Car17]). Die letzte Zeile beginnt auf der nächsten – dann rechten – Seite ein neues Kapitel.

Im doppelseitigen Satz führt \cleardoubleoddpage immer zur nächsten ungeraden Seite, \cleardoubleevenpage zur nächsten geraden Seite. Eine gegebenenfalls einzufügende Vakatseite wird mit dem über Option cleardoublepage festgelegten Seitenstil ausgegeben.

## 3.14. Fußnoten

Die Anweisungen zum Setzen von Fußnoten sind in jeder LaTeX-Einführung, beispielsweise [DGS+12], zu finden. KOMA-Script bietet darüber hinaus aber auch noch die Möglichkeit, die Form der Fußnoten zu verändern.

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. |

`footnotes=`*`Einstellung`*
`\multfootsep`

v3.00

Fußnoten werden im Text in der Voreinstellung mit kleinen, hochgestellten Ziffern markiert. Werden in der Voreinstellung `footnotes=nomultiple` 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 handele.

Mit `footnotes=multiple` werden Fußnoten, die unmittelbar aufeinander folgen, stattdessen mit einem Trennzeichen aneinander gereiht. Das in `\multfootsep` definierte Trennzeichen ist als

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

definiert. Es ist also mit einem Komma vorbelegt. Dieses kann umdefiniert werden.

Der gesamte Mechanismus ist kompatibel zu footmisc, Version 5.3d bis 5.5b (siehe [Fai11]) 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.

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

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,

v3.00 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`.

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 zuvor erklärten Anweisung `\multfootsep` gespeichert.

**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, Sie wollen außerdem, 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. Mit `\nobreakspace` wurde ein Leerzeichen mit Verhinderung eines Absatzumbruchs innerhalb der Reihung der Fußnotenzeichen verwendet.

`\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:

```
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.

Es sei darauf hingewiesen, dass die Anweisung genau wie `\ref` oder `\pageref` zerbrechlich ist und deshalb in beweglichen Argumenten wie Überschriften `\protect` davor gestellt werden sollte. Ab LaTeX 2021-05-01 wird die Anweisung übrigens von LaTeX selbst bereitgestellt.

v3.33

`\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.

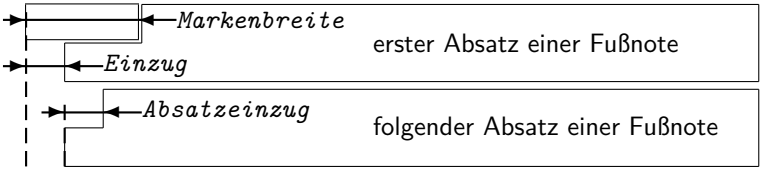



Abbildung 3.1.: Parameter für die Darstellung der Fußnoten

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 Schriftart der Markierung kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 60) für das Element `footnotelabel` davon abweichend eingestellt werden. Siehe hierzu auch Tabelle 3.2, Seite 61. Voreingestellt ist jeweils keine Umschaltung der Schrift. Bitte missbrauchen Sie das Element nicht für andere Zwecke, beispielsweise zur Verwendung von Flattersatz in den Fußnoten (siehe `\raggedfootnote`, Seite 98).

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 61). 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 60) 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 60) zu lösen:

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

### \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 60) 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.

### \raggedfootnote

v3.23 In der Voreinstellung werden die Fußnoten bei KOMA-Script genau wie bei den Standardklassen im Blocksatz gesetzt. Es ist aber auch möglich, die Formatierung abweichend vom restlichen Dokument zu ändern. Dazu ist \raggedfootnote umzudefinieren. Gültige Definitionen wären \raggedright, \raggedleft, \centering, \relax oder entsprechend der Voreinstellung eine leere Definition. Auch die Ausrichtungsbefehle des Pakets ragged2e sind zulässig (siehe [Sch09]).

**Beispiel:** Angenommen Sie verwenden Fußnoten ausschließlich, um Hinweise auf sehr lange Links anzugeben, deren Umbruch im Blocksatz zu schlechten Ergebnissen führen. Dann könnten Sie in der Dokumentpräambel mit

```
\let\raggedfootnote\raggedright
```

für die Fußnoten einfach auf linksbündigen Flattersatz umschalten.

```
\ExecuteDoHook{footnote/text/begin}
\ExecuteDoHook{footnote/text/end}
```

v3.36 Für Experten gibt es außerdem zwei Haken vom Typ *do-hook* (siehe Abschnitt 10.8 ab Seite 349). Der erste davon wird ganz zu Beginn von `\@makefntext` noch vor `\raggedfootnote` ausgeführt. Der zweite am Ende noch bevor der Absatz beendet ist. Derzeit werden beide Haken von KOMA-Script selbst nicht verwendet.

scrbook

## 3.15. Abgrenzung

Bei Büchern gibt es teilweise die Grobaufteilung in *Vorspann*, *Hauptteil* und *Nachspann*, die auch von scrbook unterstützt wird.

```
\frontmatter[Nummerierungsstil]
\frontmatter*
\mainmatter[Nummerierungsstil]
\mainmatter*
\backmatter[Nummerierungsstil]
\backmatter*
```

v3.49 Mit `\frontmatter` wird der Vorspann eingeleitet. Dadurch werden die nummerierten Seiten mit römischen Seitenzahlen versehen. Soll ein anderer *Nummerierungsstil* verwendet werden, so kann dieser per optionalem Argument eingestellt werden. Erlaubt sind hier alle Angaben, die auch für `\pagenumbering` aus Abschnitt 3.12, Seite 89 möglich sind. Soll die Seitennummerierung durchgehend erfolgen, sich also durch `\frontmatter` nicht ändern, so kann wahlweise ein leeres optionales Argument oder die Sternform `\frontmatter*` verwendet werden.

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 56). 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 112 und Seite 113) 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 56).

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. Soll ein anderer *Nummerierungsstil* verwendet werden, so kann dieser per optionalem Argument eingestellt werden. Erlaubt sind hier alle Angaben, die auch für \pagenumbering aus Abschnitt 3.12, Seite 89 möglich sind. Soll die Seitennummerierung durchgehend erfolgen, sich also durch \mainmatter nicht ändern, so kann wahlweise ein leeres optionales Argument oder die Sternform \mainmatter* verwendet werden.

v3.49

Mit \backmatter wird der Nachspann eingeleitet. Was zum Nachspann gehört, ist unterschiedlich. Manchmal wird im Nachspann nur das Literaturverzeichnis, manchmal nur das Stichwortverzeichnis gesetzt. Manchmal erscheint der gesamte Anhang im Nachspann. Der Nachspann gleicht bezüglich der Gliederungsüberschriften dem Vorspann. Eine getrennte Seitennummerierung ist jedoch nicht voreingestellt. Soll für die Seiten dennoch ein anderer *Nummerierungsstil* verwendet werden, so kann dieser per optionalem Argument eingestellt werden. Erlaubt sind hier alle Angaben, die auch für \pagenumbering aus Abschnitt 3.12, Seite 89 möglich sind. Die Sternform \backmatter* führt zu demselben Ergebnis wie die Normalform \backmatter ohne oder mit leerem optionalem Argument.

v3.49

## 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 in Tabelle 3.12. Dabei sind auch die Auswirkungen auf \cleardoublepage sowie \cleardoublepageusingstyle, \cleardoublestandardpage, \cleardoubleplainpage und die letzte der betroffenen Anweisungen, \cleardoubleemptypage, (siehe Abschnitt 3.13, Seite 92) angegeben.

v3.00

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.

Tabelle 3.12.: Mögliche Werte für Option open zur Auswahl von Umbrüchen mit Vakatseiten bei scrbook und scrreprt

| | |
|---|---|
| any | Teile, Kapitel, Index und Nachspann verwenden \clearpage, aber nicht \cleardoublepage; die Anweisungen \cleardoublepageusingstyle, \cleardoublestandardpage, \cleardoubleplainpage, \cleardoubleemptypage und \cleardoublepage verhalten sich wie bei open=right. |
| left | Teile, Kapitel, Index und Nachspann verwenden \cleardoublepage; die Anweisungen \cleardoublepageusingstyle, \cleardoublestandardpage, \cleardoubleplainpage, \cleardoubleemptypage und \cleardoublepage erzeugen einen Seitenumbruch und fügen gegebenenfalls eine Vakatseite ein, um im doppelseitigen Satz auf die nächste linke Seite zu gelangen. |
| right | Teile, Kapitel, Index und Nachspann verwenden \cleardoublepage; die Anweisungen \cleardoublepageusingstyle, \cleardoublestandardpage, \cleardoubleplainpage, \cleardoubleemptypage und \cleardoublepage erzeugen einen Seitenumbruch und fügen gegebenenfalls eine Vakatseite ein, um im doppelseitigen Satz auf die nächste rechte Seite zu gelangen. |

chapterprefix=*Ein-Aus-Wert*
appendixprefix=*Ein-Aus-Wert*
\IfChapterUsesPrefixLine{*Dann-Teil*}{*Sonst-Teil*}

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 41 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 84).

Zuweilen kommt es vor, dass man die Kapitelüberschriften im Hauptteil 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

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

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. Letztlich führt die Option dazu, dass `chapterprefix` zu Beginn des Anhangs automatisch geändert wird.

v3.18 Mit der Anweisung `\IfChapterUsesPrefixLine` kann man Code in Abhängigkeit der aktuellen Einstellung für die Präfixzeile ausführen. Ist `chapterprefix` aktiv, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*.

Die Schriftart der Kapitelnummernzeile bei `chapterprefix=true` oder `appendixprefix=true` kann mit den beiden Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 60) für das Element `chapterprefix` geändert werden. Voreingestellt ist die Verwendung des Elements `chapter` (siehe Seite 107, sowie Tabelle 3.15, Seite 111).

v2.96a

Weitere Einstellmöglichkeiten für Kapitelüberschriften sind den Erklärungen zu `\RedeclareSectionCommand` sowie den Anweisungen `\chapterlineswithprefixformat` und `\chapterlinesformat` in Abschnitt 18.8, Teil II zu entnehmen.

`headings=`*Einstellung*

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 111 zu entnehmen. Konkret setzen alle drei Einstellungen die Schrift für die Elemente `chapter`, `section`, `subsection`, `subsubsection`, `paragraph` und `subparagraph` auf entsprechende Voreinstellungen zurück. Bei scrartcl werden generell etwas kleinere Überschriften verwendet. Die Abstände vor und nach Kapitelüberschriften werden von diesen Optionen ebenfalls neu definiert.

v3.00

scrbook, scrreprt

scrbook, scrreprt

Auf Kapitelüberschriften wirken sich außerdem die beiden Optionen `headings=twolinechapter` und `headings=onelinechapter`, die den oben erklärten `chapterprefix=true` und `chapterprefix=false` entsprechen, aus. Für den Anhang stehen als Alternativen zu `appendixprefix=true` und `appendixprefix=false` die Optionen `headings=twolineappendix` und `headings=onelineappendix` zur Verfügung. Diese existieren natürlich nicht bei scrartcl.

v3.12 Option `headings=standardclasses` passt zum einen die Schriftgrößen der Überschriften an die der Standardklassen an. Des Weiteren wird die Schrift für Element `disposition` auf `\bfseries` gesetzt. Es wird also für Überschriften nicht mehr auf einen serifenlosen Font umgeschaltet. Bei Verwendung von scrbook oder scrreprt wird außerdem `headings=twolinechapter` gesetzt und die Abstände bei den Kapitelüberschriften werden denen der Standardklassen angepasst.

scrbook, scrreprt

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 zur 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 | setzt die Schrifteinstellung für die einzelnen Standard-Gliederungsebenen zurück und verwendet große Überschriften mit großen Abständen darüber und darunter. |
| normal | setzt die Schrifteinstellung für die einzelnen Standard-Gliederungsebenen zurück und verwendet mittelgroße Überschriften mit mittelgroßen Abständen darüber und darunter. |
| onelineappendix, noappendixprefix, appendixwithoutprefix, appendixwithoutprefixline | Kapitelüberschriften im Anhang werden ohne Präfixzeile gesetzt. |
| onelinechapter, nochapterprefix, chapterwithoutprefix, chapterwithoutprefixline | Kapitelüberschriften werden ohne Präfixzeile gesetzt. |
| openany | Die Anweisungen \cleardoublepageusingstyle, \cleardoublestandardpage, \cleardoubleplainpage, \cleardoubleemptypage und \cleardoublepage erzeugen einen Seitenumbruch und fügen gegebenenfalls eine Vakatseite ein, um im doppelseitigen Satz wie bei headings=openright auf die nächste rechte Seite zu gelangen. 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 gegebenenfalls 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 gegebenenfalls 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 Arguments 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 Arguments 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 Arguments der Gliederungsbefehle wird aktiviert. In der Voreinstellung wird das optionale Argument ausschließlich für den Eintrag ins Inhaltsverzeichnis verwendet.

`small`
: setzt die Schrifteinstellung für die einzelnen Standard-Gliederungsebenen zurück und verwendet kleine Überschriften mit kleinen Abständen darüber und darunter.

`standardclasses`
v3.12
: setzt die Schrifteinstellung für die einzelnen Standard-Gliederungsebenen zurück und verwendet Überschriften in den Größen der Standardklassen. Für Kapitelüberschriften wird bei scrbook und scrreprt `headings=twolinechapter` gesetzt.

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

...

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

| | |
|---|---|
| `twolinechapter`, `chapterprefix`, `chapterwithprefix`, `chapterwithprefixline` | Kapitelüberschriften werden mit einer Präfixzeile 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.

Bei Verwendung der Spracheinstellung `russian` des Pakets polyglossia wird die Einstellung der Option durch Umdefinierung von `\autodot` übrigens hart überschrieben. Auch ein nachträgliches Setzen der Option hat dann keine Wirkung mehr. Seit Version 1.50a bietet das Paket jedoch eine Option, um dieses sehr restriktive Verhalten abzuschalten. Näheres ist der Anleitung [CRSR22] zu entnehmen.

Eine 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. Spätere Änderungen der Option führen zu einer Fehlermeldung.

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. |

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

scrbook, scrreprt

Wie auch bei der Option `listof` in Abschnitt 3.20, Seite 151 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 56).

Außerdem kann mit der Option `chapteratlists` der Abstand verändert werden. Dazu gibt man als *`Wert`* den gewünschten Abstand an. Bei der Voreinstellung `listof=chaptergapsmall` (siehe Seite 152) 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. Eine Lösung, bei der nur Kapitel mit Gleitumgebungen im jeweiligen Verzeichnis angezeigt werden, finden Sie unter [Koh15].

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

```
\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 79) 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 fünf Optionen `head=`*Kolumnentitel*, `nohead=`*Ein-Aus-Wert*, `nonumber=`*Ein-Aus-Wert*, `reference=`*Querverweistitel* und `tocentry=`*Inhaltsverzeichniseintrag*, akzeptiert. Um ein Gleichheitszeichen oder ein Komma in einem der Werte von `head`, `reference` oder `tocentry` unterzubringen, muss dieses in zusätzliche geschweifte Klammern gesetzt werden.

v3.12 v3.22 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 kann dies mit einem unsichtbaren Eintrag wie `\mbox{}` erreicht werden.

**Beispiel:** Angenommen, Sie haben ein Dokument mit teilweise sehr langen Kapitelüberschriften. Diese langen 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`.

v3.22 So wie im Beispiel Option `head` den Titel für den lebenden Kolumnentitel und Option `tocentry` den Titel für den Verzeichniseintrag bestimmt, kann mit `reference` der Titel für einen Querverweise durch die Pakete nameref, titleref oder das titleref-Modul von zref explizit vorgegeben werden. Die Unterstützung für titleref ist dabei eher rudimentär, da das Paket mit den anderen beiden nur schlecht mithalten kann und auch nicht mit hyperref kompatibel ist.

v3.27 Darüber hinaus kann mit Hilfe der Option `nonumber=true` im erweiterten optionalen Argument die Nummerierung der Überschrift deaktiviert werden. Im Gegensatz zu den nachfolgend erklärten Sternvarianten der Gliederungsbefehle wird jedoch trotzdem ein Inhaltsverzeichniseintrag und gegebenenfalls ein Kolumnentitel erzeugt. Für `\part`, `\chapter` und `\section` entspricht dies weitgehend den auf Seite 113 erklärten Anweisungen `\addpart`, `\addchap` und `\addsec`.

v3.49.3 Durch Verwendung von Option `nohead=true` kann man die Erzeugung eines automatische Kolumnentitel für die eine Überschrift deaktivieren, falls diese verwendet werden. Näheres zu automatischen Kolumnentiteln finden Sie bei der Beschreibung zu `\chaptermark` bis `\subsectionmark`, Seite 120, bei der Beschreibung des Seitenstils `headings` in Abschnitt 3.12, Seite 84 oder in Kapitel 5, Seite 284.

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 `\setpartpreamble`, Seite 122) alleine auf einer Seite.

scrbook, scrreprt

scrbook, `\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 101).

scrbook, scrreprt

Bitte beachten Sie, dass \part und \chapter bei scrbook und scrreprt den Seitenstil für eine Seite umschalten. Der jeweilige Seitenstil ist bei KOMA-Script in den Makros \partpagestyle und \chapterpagestyle abgelegt (siehe Abschnitt 3.12, Seite 87).

v2.8p

Die Schriftart aller sieben Gliederungsebenen kann mit den Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 3.6, Seite 60) bestimmt werden. Dabei wird zunächst generell das Element disposition und anschließend zusätzlich je Gliederungsebene ein spezifisches Element verwendet (siehe Tabelle 3.2, Seite 61). Für den Nummernteil der Teile-Überschrift wird zusätzlich das Element partnumber verwendet, für die optionale Präfixzeile der Kapitelüberschriften das Element chapterprefix. Die Schriftart für das Element disposition ist

v3.39

als \normalcolor\maybesffamily\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 102). 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 (\part oder \addpart) zu fett wirken. Nun könnten Sie natürlich wie folgt verfahren:

```
\setkomafont{disposition}{\normalcolor\maybesffamily}
\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\maybesffamily}
\addtokomafont{partnumber}{\normalfont\maybesffamily}
```

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 |
| | chapterprefix | \usekomafont{chapter} |
| | section | \Large |
| | subsection | \large |
| | subsubsection | \normalsize |
| | paragraph | \normalsize |
| | subparagraph | \normalsize |
| headings=normal | part | \huge |
| | partnumber | \huge |
| | chapter | \LARGE |
| | chapterprefix | \usekomafont{chapter} |
| | section | \Large |
| | subsection | \large |
| | subsubsection | \normalsize |
| | paragraph | \normalsize |
| | subparagraph | \normalsize |
| headings=small | part | \LARGE |
| | partnumber | \LARGE |
| | chapter | \Large |
| | chapterprefix | \usekomafont{chapter} |
| | 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 53 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}
```

v3.49.3

Bei den Sternvarianten der Gliederungsbefehle erfolgt keine Nummerierung, wird kein Kolumnentitel gesetzt und kein Eintrag im Inhaltsverzeichnis vorgenommen. Die Verwendung der Sternvariante entspricht damit der sternlosen Hauptform mit dem optionalen Argument `[tocentry={}, nohead=true, nonumber=true]`, soweit die erweiterte Interpretation des optionalen Arguments aktiv ist. Siehe dazu die vorausgehende Erklärung zu `\part` bis `\subparagraph`.

scrbook, scrreprt

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.

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 87).

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.

```
\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 sowohl einen automatischen Kolumnentitel als auch einen Eintrag im Inhaltsverzeichnis, wobei auch die Einstellungen von Option headings einschließlich der Erweiterung für das optionale Argument beachtet werden. Das Aktivieren oder Deaktivieren des in der Erklärung zu \part, \chapter und \section dokumentierten Schalters nonumber bleibt dabei jedoch wirkungslos.

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 der Buch- und Berichtklasse, 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 87).

v2.8p Bezüglich der Möglichkeiten der Schriftumschaltung gilt das Gleiche, wie zuvor in der Erklärung zu \section*, \chapter* und \part* 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 61). 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}

\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 ab der Überschrift »Bauteile« sieht anschließend wie folgt aus:

**Bauteile**
1 Brett ($100 \times 50 \times 12$)
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.

```
\raggedsection
\raggedchapter
\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.

v3.15 Da manche Anwender für die Kapitelebene eine andere Ausrichtung wünschen als für alle anderen Ebenen, kann diese über `\raggedchapter` auch getrennt verändert werden. In der Voreinstellung verwendet die Anweisung jedoch einfach `\raggedsection`, so dass eine Änderung von `\raggedsection` sich indirekt auch auf `\raggedchapter` auswirkt.

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
\sectionformat
\subsectionformat
\subsubsectionformat
\paragraphformat
\subparagraphformat
\othersectionlevelsformat{Gliederungsname}{}{Zählerausgabe}
\IfUsePrefixLine{Dann-Teil}{Sonst-Teil}
\autodot
```

KOMA-Script fügt der Ausgabe der Gliederungsnummern oberhalb von `\the`*`Gliederungsname`* (siehe `\the`*`Zähler`*, Seite 481) 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` bis `\subparagraphformat` formatiert. Die Anweisung `\chapterformat` existiert, wie bereits `\thechapter`, selbstverständlich nicht in der Klasse scrartcl, sondern nur in den Klassen scrbook und scrreprt.

scrbook, scrreprt

Wie bereits bei Option `numbers` am Anfang dieses Abschnitts (siehe Seite 105) 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.

v3.17

Die Anweisung `\othersectionlevelsformat` wird seit KOMA-Script 3.17 nur noch in Ausnahmefällen verwendet, wenn für eine Gliederungsebene keine Format-Anweisung definiert oder diese `\relax` ist, was jedoch in der Voreinstellung für die Gliederungsebenen von KOMA-Script nicht zutrifft. Daher wird sie auch nicht mehr offiziell dokumentiert. Bei einer Kompatibilitätseinstellung zu Versionen vor 3.17 (siehe Option `version`, Abschnitt 3.2, Seite 56) haben hingegen die Anweisungen `\sectionformat` bis `\subparagraphformat` keine interne Funktion. Stattdessen wird für diese Ebenen dann weiterhin `\othersectionlevelsformat` verwendet.

Mit Hilfe von `\renewcommand` kann jede der Formatierungsanweisungen umdefiniert werden, um sie eigenen Anforderungen anzupassen. Nachfolgend finden Sie einige Definitionen, die denen aus den KOMA-Script-Klassen entsprechen:

```
\newcommand*{\partformat}{\partname~\thepart\autodot}
\newcommand*{\chapterformat}{%
  \mbox{\chapappifchapterprefix{\nobreakspace}\thechapter
    \autodot\IfUsePrefixLine{}{\enskip}}}
\newcommand*{\sectionformat}{\thesection\autodot\enskip}
\newcommand*{\subsectionformat}{%
```

```
  \thesubsection\autodot\enskip}
\newcommand*{\subsubsectionformat}{%
  \thesubsubsection\autodot\enskip}
\newcommand*{\paragraphformat}{\theparagraph\autodot\enskip}
\newcommand*{\subparagraphformat}{%
  \thesubparagraph\autodot\enskip}
\newcommand*{\othersectionlevelsformat}[3]{%
  #3\autodot\enskip}
```

v3.17 Wegen der Verwendung von `\IfUsePrefixLine` sollte die Anweisung `\chapterformat` nicht außerhalb von `\chapter` verwendet werden. `\IfUsePrefixLine` ist nur innerhalb der Gliederungsbefehle von KOMA-Script wohldefiniert. Dort wird dann im Falle der Verwendung einer Präfixzeile für die Nummer der Überschrift der *Dann-Teil* ausgeführt, während im anderen Fall der *Sonst-Teil* zur Ausführung kommt.

Bitte beachten Sie außerdem, dass bei der Umdefinierung selbstverständlich `\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 105.

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*{\sectionformat}{%
  \makebox[0pt][r]{\thesection\autodot\enskip}}
\renewcommand*{\subsectionformat}{%
  \makebox[0pt][r]{\thesubsection\autodot\enskip}}
\renewcommand*{\subsubsectionformat}{%
  \makebox[0pt][r]{%
    \thesubsubsection\autodot\enskip}}
\renewcommand*{\paragraphformat}{%
  \makebox[0pt][r]{\theparagraph\autodot\enskip}}
\renewcommand*{\paragraphformat}{%
  \makebox[0pt][r]{%
    \thesubparagraph\autodot\enskip}}
```

Die optionalen Argumente der \makebox-Anweisungen fordern von LaTeX dabei eine Box der Breite Null, deren Inhalt rechtsbündig angeordnet ist. Im Ergebnis wird der Inhalt der Box links von der aktuellen Position ausgegeben. Näheres zu den optionalen Argumenten von \makebox ist [Tea23] zu entnehmen.

Für Formatierungsänderungen bei den Überschriften, die über die reine Ausgabe der Nummer hinaus gehen, sei ausdrücklich auf \partlineswithprefixformat, \chapterlineswithprefixformat und \chapterlinesformat sowie \sectionlinesformat und das zugehörige \sectioncatchphraseformat in Abschnitt 18.8, ab Seite 502 verwiesen.

Es wird ausdrücklich darauf hingewiesen, dass das Paket polyglossia einige der hier dokumentierten Anweisungen abhängig von der Sprache hart umdefiniert. Dies betrifft beispielsweise die Spracheinstellungen russian und hungarian. Näheres dazu ist dessen Anleitung [CRSR22] zu entnehmen.

\chapappifchapterprefix{*Zusatztext*}
\chapapp

scrbook, scrreprt

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 101) 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}
\addchapmark{Kolumnentitel}
\sectionmark{Kolumnentitel}
\addsecmark{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 dabei übergebene Parameter beinhaltet den Text der Gliederungsüberschrift. Die zugehörige Gliederungsnummer wird automatisch in der `\...mark`-Anweisung hinzugefügt. Die Formatierung erfolgt je nach Gliederungsebene mit einer der drei Anweisungen `\chaptermarkformat`, `\sectionmarkformat` und `\subsectionmarkformat`.

Es ist zu beachten, dass die Kolumnentitel von `\addchap` und `\addsec` ebenfalls auf `\chaptermark` und `\sectionmark` basieren. Dabei wird aber lokal der Zähler `secnumdepth` auf einen Wert gesetzt, mit dem die Nummerierung von Kapiteln beziehungsweise Abschnitten abgeschaltet wird. Dies sollte man beispielsweise bei der Umdefinierung von `\chaptermark` und `\sectionmark` berücksichtigen (siehe `\Ifnumbered` auf Seite 122). Für die Sternformen `\addchap*` und `\addsec*` existieren außerdem die Anweisungen `\addchapmark` und `\addsecmark`, die in der Voreinstellung ebenfalls in der genannten Weise definiert sind.

scrbook, scrreprt

Während bei scrartcl weder `\chaptermark` noch `\addchapmark` oder `\chaptermarkformat` existieren, gibt es die beiden Anweisungen `\subsectionmark` und `\subsectionmarkformat` nicht nur bei scrartcl. Bei scrbook und scrreprt ist `\subsectionmark` aber als Leeranweisung definiert, hat also keine Auswirkung. Bei Verwendung des Pakets scrlayer-scrpage kann sich dies ändern (siehe Kapitel 5).

So wie mit `\partformat` bis `\subparagraphformat` die Nummern der Gliederungsüberschriften formatiert ausgegeben werden, werden entsprechend mit `\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 die beiden Anweisungen \chapapp und \chapappifchapterprefix Verwendung, die weiter oben erklärt wurden.

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

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.

v3.12 Damit sich der Anwender keine abstrakten Nummern merken muss, um einstellen zu können, bis zu welcher Gliederungsebene die Überschriften nummeriert werden sollen, gibt es die Anweisungen \partnumdepth bis \subparagraphnumdepth. Diese liefern die entsprechende Nummer zur jeweiligen Gliederungsebene.

**Beispiel:** Sie wollen, dass in einem Buchprojekt nur die Gliederungsebenen vom Teil (engl. *part*) über das Kapitel (engl. *chapter*) bis zum Abschnitt (engl. *section*) nummeriert werden. Dazu müssen Sie in der Dokumentpräambel den Zähler secnumdepth auf den Wert setzen, den die Anweisung \sectionnumdepth liefert:

```
\setcounter{secnumdepth}{\sectionnumdepth}
```

Eine Umdefinierung der Anweisungen ist nicht vorgesehen und wird ausdrücklich nicht empfohlen, da dies zu unvorhergesehenen Ergebnissen sowohl mit KOMA-Script als auch bei Verwendung von Drittpaketen führen kann.

Der Zähler secnumdepth ist nicht zu verwechseln mit dem Zähler tocdepth (siehe Abschnitt 3.9, Seite 79). Auch sind die Werte je nach Klasse nicht eins zu eins übertragbar.

\Ifnumbered{*Gliederungsebene*}{*Dann-Teil*}{*Else-Teil*}
\Ifunnumbered{*Gliederungsebene*}{*Dann-Teil*}{*Else-Teil*}

v3.28

Nachdem zuvor erklärt wurde, wie man bestimmen kann, welche Gliederungsebenen nummeriert werden sollen, kann mit diesen Anweisungen nun Code abhängig davon, ob eine *Gliederungsebene* nummeriert wird oder nicht, ausgeführt werden. Wird mit den aktuellen Einstellungen eine *Gliederungsebene* nummeriert, so führt \Ifnumbered den *Dann-Teil* aus, während \Ifunnumbered den *Else-Teil* ausführt. Wird mit den aktuellen Einstellungen eine *Gliederungsebene* nicht nummeriert, dann führt \Ifnumbered den *Else-Teil* aus, wohingegen \Ifunnumbered den *Dann-Teil* ausführt. Als *Gliederungsebene* wird der englische Name der Ebene angegeben, also part, chapter, section, subsection, subsubsection, paragraph oder subparagraph.

KOMA-Script selbst verwendet diesen Test beispielsweise in der Definition von \chaptermark beim Seitenstil headings. Dadurch wird indirekt auch sichergestellt, dass Überschriften mit \addchap den Kolumnentitel nicht mit einer Nummer versehen (siehe auch \addchapmark, Seite 120).

\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 102) und der Optionen für die Form der abstract-Umgebung (siehe Abschnitt 3.8, Seite 74), sieht das Ergebnis ungefähr wie folgt aus:

# Teil III.

# Abteilung Grünschnitt

**Zusammenfassung**

Dies ist ein Blindtext zur Demonstration.

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 74).

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 so jeweils eine Präambel unter und über der Überschrift gesetzt werden, mit drei Möglichkeiten für die horizontale Position:

- `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 Einstellung `beforeskip` für `\RedeclareSectionCommand` in Abschnitt 18.8, Tabelle 18.3, Seite 491.

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

v3.39

| Element | Voreinstellung |
|---|---|
| dictum | \normalfont\normalcolor\maybesffamily\small |
| dictumauthor | \itshape |

## 3.17. Schlauer Spruch

Ein vor allem bei Büchern manchmal anzutreffendes Element sind Redewendungen oder Zitate, die rechtsbündig unter oder über einer Überschrift gesetzt werden. 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 122) zu verwenden.

Der Spruch wird zusammen mit einem optional anzugebenden *Urheber* in einer `\parbox` (siehe [Tea23]) 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 `dictum` eingestellten Schriftart, die mit den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 60) 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 den *Urheber* als Argument. In der Voreinstellung ist `\dictumauthorformat` mit

```
\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 Elements `dictum` 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 60) möglich.

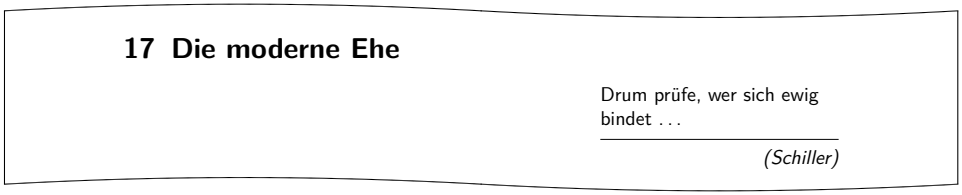
Wird `\dictum` innerhalb der Anweisung `\setchapterpreamble` oder `\setpartpreamble` (siehe Abschnitt 3.16, Seite 122) verwendet, so ist Folgendes zu beachten: `\textwidth` ist nicht die Breite des gesamten Textkörpers, sondern wie bei `minipage` die aktuelle Textbreite. Ist also die Breite `\dictumwidth` 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. Die horizontale Anordnung erfolgt immer mit `\raggeddictum`. Das optionale Argument zur horizontalen Anordnung, das die beiden Anweisungen vorsehen, ändert also nicht die Ausrichtung des Spruchs innerhalb der Box. 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 ragged2e hingewiesen, mit dem man Flattersatz mit Trennung erreichen kann (siehe [Sch09]).

## 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 Schriftart der Zeichen für die einzelnen Ebenen kann außerdem bei den KOMA-Script-Klassen mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 60) für die Elemente `labelitemi`, `labelitemii`, `labelitemiii` und `labelitemiv` geändert werden. In der Voreinstellung verwenden diese alle die Schrifteinstellung für Element `itemizelabel`. Lediglich Element `labelitemii` verwendet zusätzlich `\bfseries`. Die Voreinstellung von `itemizelabel` selbst ist `\normalfont`. Die einzelnen Stichpunkte der Umgebung werden mit `\item` eingeleitet.

v3.33

**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
  \item Automobile
    \begin{itemize}
    \item Rennwagen
    \item Personenwagen
    \item Lastwagen
    \end{itemize}
  \item Fahrräder
  \end{itemize}
```

Anschließend erhalten Sie:

**Die Fahrzeuge im Spiel**

- Flugzeuge
  - Doppeldecker
  - Jets
  - Transportmaschinen
    - einmotorig
      - düsengetrieben
      - propellergetrieben
    - zweimotorig
      - düsengetrieben
      - propellergetrieben
  - Drehflügler
- Motorräder
- 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 Schriftart der Nummern beziehungsweise Label für die einzelnen Ebenen kann außerdem bei den KOMA-Script-Klassen mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont`

v3.44

(siehe Abschnitt 3.6, Seite 60) für die Elemente `labelenumi`, `labelenumii`, `labelenumiii` und `labelenumiv` geändert werden. In der Voreinstellung verwenden diese alle die Schrifteinstellung für Element `enumeratelabel`. Die Voreinstellung für `enumeratelabel` selbst ist leer, also keine Änderung der Schrift. 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. Automobile
   a) Rennwagen
   b) Personenwagen

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 60) für das Element `descriptionlabel` (siehe Tabelle 3.2, Seite 61) geändert werden.

v3.39

In der Voreinstellung wird `\maybesffamily\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 Schlüsselwort}
  \item[Stichwort] ...
  ⋮
\end{labeling}
```

Eine andere Form der Stichwortliste ist nur bei den KOMA-Script-Klassenvorhanden: die `labeling`-Umgebung. Im Unterschied zu `description` kann bei `labeling` ein Muster angegeben werden, dessen Länge die Einrücktiefe aller Stichpunkte ergibt. Zwischen Stichpunkt und Beschreibungstext kann 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 60) für das Element `labelinglabel` (siehe Tabelle 3.2, Seite 61) geändert werden. Für die davon abweichende Schriftart der Trennzeichen ist das Element `labelingseparator` (siehe ebenfalls Tabelle 3.2, Seite 61) zuständig. Beide sind in der Voreinstellung leer, es findet also keine Schriftumschaltung statt.

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 `\\*` anstelle 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, kann man zusätzlich `\interlinepenalty` auf 10000 setzen:

```
\begin{verse}\interlinepenalty 10000
  Der Philosoph wie der Hausbesitzer
  hat immer Reparaturen.\\
  \bigskip
  Wer dir sagt, er hätte noch nie
  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.

`\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 unter Verwendung der Umgebung `quotation`:

```
\documentclass[paper=a5,pagesize]{scrartcl}
\usepackage[ngerman]{babel}
\begin{document}
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}
\end{document}
```

Das Ergebnis könnte dann so aussehen:

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 anstelle 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 soll. 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 83) und `\ifthispagewasodd` (Abschnitt 18.1, Seite 481).

Im Zusammenspiel von Listen mit Absätzen ergeben sich für Laien häufiger Fragen. Daher widmet sich die weiterführende Erklärung zu Option `parskip` in Abschnitt 18.1, Seite 481 auch diesem Thema. Ebenfalls im Expertenteil in Abschnitt 18.1, Seite 481 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 [DGS+12] 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.

```
\begin{figure}...\end{figure}
\begin{figure*}...\end{figure*}
\begin{figure-}...\end{figure-}
\begin{table}...\end{table}
\begin{table*}...\end{table*}
\begin{table-}...\end{table-}
```

Die KOMA-Script-Klassen scrbook, scrreprt und scrartcl verwenden die in Abschnitt 13.6, Seite 400 dokumentierte Anweisung `\DeclareNewTOC` zu Definition der von den Standardklassen gewohnten Umgebungen `figure` und `table`. Selbstverständlich werden dabei auch die für den zweispaltigen Modus vorgesehenen Umgebungen `figure*` und `table*` definiert, die für spaltenübergreifende Abbildungen und Tabellen am Seitenanfang gedacht sind. Da bei der Definition neben der Einstellung `float` auch `nonfloat` verwendet wird, werden außerdem auch die nicht gleitenden Umgebungen `figure-` und `table-` definiert. Diese können wie die anderen Umgebungen verwendet werden und spannen letztlich eine `minipage` mit passendem Abstand davor und danach auf. Bei gemischter Verwendung von gleitenden und nicht gleitenden Umgebungen ist der Anwender selbst für die korrekte Reihenfolge bei der Ausgabe und damit bei der Nummerierung verantwortlich.

v3.36

```
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. Intern wird dies durch Verwendung der tocbasic-Option `tablecaptionposition=top` erreicht. v3.46

Es sei darauf hingewiesen, dass mehrseitige Tabellen nicht als Gleitumgebungen gesetzt werden können. Für den automatischen Seitenumbruch von Tabellen werden Zusatzpakete wie longtable (siehe [Car04]) oder supertabular (siehe [BJ04]) benötigt.

Mit der Einstellung `captions=tablesignature` wird wieder die Voreinstellung der Formatierung als Tabellenunterschrift gewählt. Intern wird dies durch Verwendung der tocbasic-Option `tablecaptionposition=bottom` erreicht. v3.46

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 `\restylefloat` (siehe [Lin01]).

Natürlich existiert mit den Optionen `captions=figureheading` und `captions=figuresignature` auch die entsprechende Funktion für Abbildungen. Intern werden diese durch Verwendung der tocbasic-Option `figurecaptionposition=top` beziehungsweise `figurecaptionposition=bottom` realisiert. 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. v3.09 v3.46

Manchmal wird jedoch gewünscht, alle Gleitumgebungen mit Überschriften zu versehen. Daher gibt es bei KOMA-Script die Einstellungen `captions=heading` und `captions=signature`, mit der man die Formatierungen aller Gleitumgebungen entsprechend ändern kann. Intern werden diese durch Verwendung der tocbasic-Option `captionposition=top` beziehungsweise `captionposition=bottom` realisiert. v3.46

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]. Dies gilt ebenso für `\caption` innerhalb von neuen Gleitumgebungen, die mit float definiert wurden. Zusätzliche Unterstützung, die KOMA-Script bei Verwendung des float-Pakets bietet, finden Sie bei der Erklärung zu `komaabove` (siehe Seite 145). Als Alternative zu float sei auch die Verwendung von `\captionof` (siehe Seite 141) und `\DeclareNewTOC` (siehe Seite 400) erwähnt. Darüber hinaus sei bei Verwendung von float ausdrücklich Paket floatbytocbasic (siehe [Koh23b]) empfohlen.

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 `captionbeside` ist ab Seite 142 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 `captionbeside` (siehe Abschnitt 3.20, Seite 142) werden mit der untersten Grundlinie auf Höhe der untersten Grundlinie des Inhalts der Gleitumgebung ausgerichtet.

`centeredbeside`, `besidecentered`, `middlebeside`, `besidemiddle`
: Gleitumgebungstitel der Umgebung `captionbeside` (siehe Abschnitt 3.20, Seite 142) werden zum Inhalt der Gleitumgebung vertikal zentriert ausgerichtet.

`figureheading`, `figureabove`, `abovefigure`, `topatfigure`
v3.09 v3.46
: Bei Abbildungen wird (gegebenenfalls abweichend von `captions=signature`) die Formatierung als Überschrift gewählt. Diese Option basiert auf Option `figurecaptionposition=top` von Paket tocbasic. Siehe dazu Abschnitt 13.7, ab Seite 406.

`figuresignature`, `figurebelow`, `belowfigure`, `bottomatfigure`
v3.09 v3.46
: Bei Abbildungen wird (gegebenenfalls abweichend von `captions=heading`) die Formatierung als Unterschrift gewählt. Diese Option basiert auf Option `figurecaptionposition=bottom` von Paket tocbasic. Siehe dazu Abschnitt 13.7, ab Seite 406.

`heading`, `above`, `top`
v3.09 v3.46
: Gleitumgebungstitel werden als Überschriften formatiert. Dies hat jedoch keinen Einfluss darauf, ob sie über oder unter der Gleitumgebung platziert werden. Diese Option basiert auf Option `captionposition=top` von Paket tocbasic. Siehe dazu Abschnitt 13.7, ab Seite 406.

`innerbeside`, `besideinner`
: Gleitumgebungstitel der Umgebung `captionbeside` (siehe Abschnitt 3.20, Seite 142) werden bei doppelseitigem Satz in der Voreinstellung innen neben dem Inhalt der Gleitumgebung gesetzt. Dies entspricht im einseitigen Satz `captions=leftbeside`.

...

Tabelle 3.17.: Mögliche Werte für Option `captions` (*Fortsetzung*)

`leftbeside`, `besideleft`
: Gleitumgebungstitel der Umgebung `captionbeside` (siehe Abschnitt 3.20, Seite 142) 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 `captionbeside` (siehe Abschnitt 3.20, Seite 142) werden bei doppelseitigem Satz in der Voreinstellung außen neben dem Inhalt der Gleitumgebung gesetzt. Dies entspricht im einseitigen Satz `captions=rightbeside`.

`rightbeside`, `besideright`
: Gleitumgebungstitel der Umgebung `captionbeside` (siehe Abschnitt 3.20, Seite 142) werden in der Voreinstellung rechts neben dem Inhalt der Gleitumgebung gesetzt.

`signature`, `below`, `bot`, `bottom`
: v3.09 v3.46 Gleitumgebungstitel werden als Unterschriften formatiert. Dies hat jedoch keinen Einfluss darauf, ob sie über oder unter der Gleitumgebung platziert werden. Diese Option basiert auf Option `captionposition=bottom` von Paket tocbasic. Siehe dazu Abschnitt 13.7, ab Seite 406.

`tableheading`, `tableabove`, `abovetable`, `abovetabular`, `topattable`
: v3.46 Bei Tabellen wird (gegebenenfalls abweichend von `captions=signature`) die Formatierung als Überschrift gewählt. Diese Option basiert auf Option `tablecaptionposition=top` von Paket tocbasic. Siehe dazu Abschnitt 13.7, ab Seite 406.

`tablesignature`, `belowtable`, `belowtabular`, `bottomattable`
: v3.46 Bei Tabellen wird (gegebenenfalls abweichend von `captions=heading`) die Formatierung als Unterschrift gewählt. Diese Option basiert auf Option `tablecaptionposition=bottom` von Paket tocbasic. Siehe dazu Abschnitt 13.7, ab Seite 406.

...

Tabelle 3.18.: Voreinstellungen der Schrift für die Elemente der Tabellen- oder Abbildungsunterschrift bzw. -überschrift

| Element | Voreinstellung |
|---|---|
| caption | \normalfont |
| captionlabel | \normalfont |

Tabelle 3.17.: Mögliche Werte für Option captions (*Fortsetzung*)

| | |
|---|---|
| topbeside, besidetop | Gleitumgebungstitel der Umgebung captionbeside (siehe Abschnitt 3.20, Seite 142) werden mit der obersten Grundlinie auf Höhe der obersten Grundlinie des Inhalts der Gleitumgebung ausgerichtet. |

```
\caption[Verzeichniseintrag]{Titel}
\captionbelow[Verzeichniseintrag]{Titel}
\captionabove[Verzeichniseintrag]{Titel}
\captionof{Objekttyp}[Verzeichniseintrag]{Titel}
\captionaboveof{Objekttyp}[Verzeichniseintrag]{Titel}
\captionbelowof{Objekttyp}[Verzeichniseintrag]{Titel}
```

Bezüglich der generellen Funktion dieser Anweisungen zum Setzen eines Titels für Tabellen und Abbildungen sei Abschnitt 13.7, ab Seite 406 verwiesen.

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 60) verändert werden. Zuständig sind hier die Elemente caption und captionlabel (siehe Tabelle 3.2, Seite 61). Dabei wird die Schriftart für 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}
```

```
\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 genau den gleichnamigen Parametern von `\caption`, `\captionabove` oder `\captionbelow`. Der *Titel* wird 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 137) 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

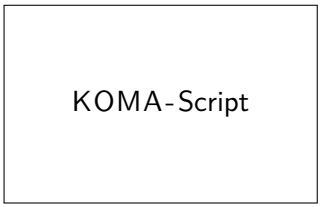



Abbildung 3.2.: Eine Bildbeschreibung weder über noch unter der Abbildung, sondern unten daneben

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 137) 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:** Abbildung 3.2 veranschaulicht die folgende Verwendung der `captionbeside`-Umgebung:

```
\begin{figure}
  \begin{captionbeside}%
    [Beispiel: Bildbeschreibung daneben, unten]%
    {Eine Bildbeschreibung weder über noch unter
      der Abbildung, sondern unten daneben}%
    [i][\dimexpr \linewidth
      +\marginparwidth+\marginparsep\relax][Opt]*
    \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` um `\marginparwidth` + `\marginparsep` nach außen erweitert. Der Titel steht innen neben der Abbildung. Damit erscheint die Abbildung selbst bis in die Marginalienspalte in den Rand gerückt.

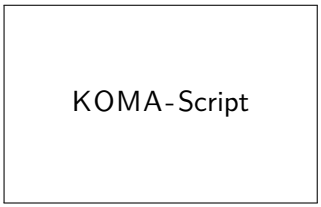



Abbildung 3.3.: Eine Bildbeschreibung weder über noch unter der Abbildung, sondern mittig daneben

Abbildung 3.3 zeigt die wenig empfehlenswerte Zentrierung der Bildbeschreibung mit:

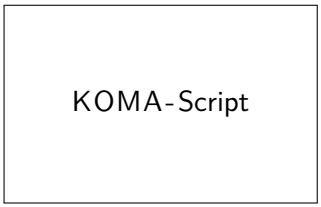
```
\KOMAoption{captions}{centeredbeside}
```

Demgegenüber kann die Ausrichtung oben, die in Abbildung 3.4 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 zuvor angegeben, sondern auch beispielsweise auf `\includegraphics` (siehe [Car17]) 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 oben, neben einem
      Beispielbild aus Paket \texttt{mwe}}%
    [i][\dimexpr\linewidth
      \dimexpr\marginparwidth+\marginparsep\relax]
    [0pt]*
    \raisebox{%
      \dimexpr\baselineskip-\totalheight\relax
    }{%
      \includegraphics{example-image-1x1}%
    }%
  \end{captionbeside}
  \label{fig:maincls.captionbesidetop}
\end{figure}
\end{document}
```

Abbildung 3.4.: Eine Bildbeschreibung weder über noch unter der Abbildung, sondern oben daneben




```
\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 der *Objekttyp* als zusätzliches, erstes Argument anzugeben.

```
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 zwei weitere `\...format`-Anweisungen. Diese sind wie folgt vordefiniert:

```
\newcommand*{\figureformat}{\figurename~\thefigure\autodot}
\newcommand*{\tableformat}{\tablename~\thetable\autodot}
```

Mit `\renewcommand` können diese leicht umdefiniert werden.

**Beispiel:** Hin und wieder wird gewünscht, dass die Beschreibungstexte ohne Label und Trennzeichen ausgegeben werden. Bei KOMA-Script genügen dafür folgende Definitionen:

```
\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*{`*Einzug*`}`
`\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 `\setcapindent` oder `\setcapindent*` geändert werden. Dabei gibt

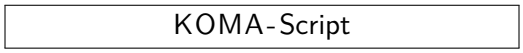



**Abbildung 3.5.:** Mit der Standardeinstellung, also wie bei Verwendung von `\setcaphanging`

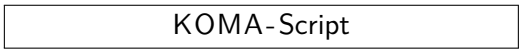



**Abbildung 3.6.:** Mit teilweise hängendem Einzug ab der zweiten Zeile durch Verwendung von `\setcapindent{1em}`

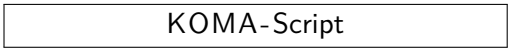



**Abbildung 3.7.:**
Mit hängendem Einzug ab der zweiten Zeile und Umbruch vor der Beschreibung durch Verwendung von `\setcapindent*{1em}`

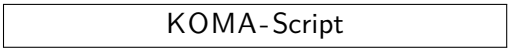



**Abbildung 3.8.:**
Mit Einzug lediglich in der zweiten Zeile und einem Umbruch vor der Beschreibung durch Verwendung von `\setcapindent{-1em}`

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 *Einzug* der Beschreibung stattdessen mit der Sternvariante der Anweisung: `\setcapindent*`. Mit einem negativen *Einzug* bei `\setcapindent` 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 138.

**Beispiel:** Die Abbildungen 3.5 bis 3.8 zeigen die Auswirkungen unterschiedlicher Einstellungen. Es wird deutlich, dass bei geringer Spaltenbreite der komplett hängende Einzug unvorteilhaft ist. Der Quelltext der zweiten Abbildung sei hier mit gekürzter 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 im Beispiel zu sehen ist, kann die Formatierung auch lokal innerhalb einer Gleitumgebung geändert werden. Die Änderung gilt dann nur für die eine Umgebung. Nachfolgende Abbildungen und Tabellen werden wieder mit den Grundeinstellungen oder den globalen Einstellungen, die Sie beispielsweise in der Dokumentpräambel vorgenommen haben, gesetzt.

```
\setcapwidth[Ausrichtung]{Breite}
\setcapdynwidth[Ausrichtung]{Breite}
\setcapmargin[Rand links]{Rand}
\setcapmargin*[Rand innen]{Rand}
```

v2.8q Mit Hilfe dieser Befehle kann die Breite und Anordnung der Beschreibung beeinflusst werden. Normalerweise steht die gesamte Text- oder Spaltenbreite 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 bestimmt. Die möglichen Ausrichtungen finden Sie in der folgenden Liste.

- `l` – links
- `r` – rechts
- `i` – innen: auf rechten Seiten links, auf linken Seiten rechts
- `o` – außen: auf rechten Seiten rechts, auf linken Seiten links

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 Pakets longtable.

v3.20 Zu beachten ist, dass die an `\setcapwidth` übergebene *Breite* wie bei `\setlength` zum Zeitpunkt der Zuweisung ausgewertet wird. Will man hingegen, dass *Breite* erst bei der Verwendung ausgewertet wird, kann man stattdessen `\setcapdynwidth` verwenden. Unterschiede gibt es beispielsweise, wenn Längen wie `\linewidth` oder andere Anweisungen als Argument verwendet werden.

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 `captions=nooneline` (siehe Seite 138) für die Beschreibungen, die mit dieser Randeinstellung gesetzt werden.

Soll die Beschreibung in einen Rand ragen, gibt man übrigens für das entsprechende Argument einfach einen negativen Wert an.

Tabelle 3.19.: Ausrichtungen für mehrzeilige Beschreibungen in Gleitumgebungen

| | |
|---|---|
| c | zentriert |
| j | Blocksatz |
| l | linksbündig |
| r | rechtsbündig |
| C | zentriert mit ragged2e |
| J | Blocksatz mit ragged2e |
| L | linksbündig mit ragged2e |
| R | rechtsbündig mit ragged2e |

`\setcaptionalignment[`*`Gleitumgebung`*`]{`*`Ausrichtung`*`}`

v3.25

Normalerweise werden mehrzeilige Beschreibungen im Blocksatz gesetzt. Dies entspricht `\setcaptionalignment{j}`. Manchmal wird allerdings eine davon abweichende Ausrichtung gewünscht, beispielsweise linksbündiger Flattersatz. Eine entsprechende Änderung ist mit `\setcaptionalignment` jederzeit möglich. Für *`Ausrichtung`* kann dabei genau einer der Buchstaben aus Tabelle 3.19 angegeben werden. Wird eine unbekannte *`Ausrichtung`* angegeben, so resultiert dies in einer Fehlermeldung.

Die vier Möglichkeiten mit Paket ragged2e stehen nur zur Verfügung, wenn das Paket vor Verwendung von `\setcaptionalignment` geladen wurde. Anderenfalls werden sie auf die entsprechenden Möglichkeiten ohne ragged2e abgebildet. Zur Sicherheit wird in diesem Fall eine Warnung ausgegeben.

Verwendet man den Befehl ohne den optionalen Parameter, so hängt das Ergebnis davon ab, ob der Aufruf innerhalb oder außerhalb einer Gleitumgebung stattfindet. Innerhalb einer Gleitumgebung wird dann die Ausrichtung für diese Gleitumgebung gesetzt. Außerhalb wird hingegen ein leerer optionaler Parameter angenommen.

Beim Aufruf mit einem leeren optionalen Parameter oder außerhalb einer Gleitumgebung auch komplett ohne optionalen Parameter wird die allgemeine Ausrichtung festgelegt. Diese wird immer dann verwendet, wenn keine Ausrichtung für den aktuellen Gleitumgebungstyp definiert ist.

Will man nur die Ausrichtung eines bestimmen Typs von Gleitumgebungen festlegen, ohne *`Ausrichtung`* auch für andere Arten von Gleitumgebungen zu verändern, so gibt man den Typ der Gleitumgebung, beispielsweise `figure` oder `table`, als optionalen Parameter *`Gleitumgebung`* an.

**Beispiel:** Sie wollen, dass Bildunterschriften auch dann vollständig zentriert unter den Bildern stehen, wenn sie mehrzeilig sind. Um das zunächst einmal nur für eine einzige Abbildung zu testen, verwenden Sie:

```
\begin{figure}
  \centering
  \setcaptionalignment{c}
```

```
    \includegraphics{example-image}
    \caption{\blindtext}
  \end{figure}
```

Da Sie mit dem Ergebnis zufrieden sind, verschieben Sie die Anweisung

```
\setcaptionalignment{c}
```

in die Dokumentpräambel. Daraufhin bemerken Sie allerdings, dass Ihnen diese Änderung für Tabellenüberschriften überhaupt nicht gefällt. Daher beschränken Sie mit

```
\setcaptionalignment[figure]{c}
```

die Zentrierung auf Abbildungen.

Etwas später stellen Sie fest, dass die Zentrierung doch nicht so günstig ist. Stattdessen wollen Sie nun lieber eine linksbündige Ausrichtung im Flattersatz haben. Also ändern Sie die Anweisung erneut zu:

```
\setcaptionalignment[figure]{l}
```

Allerdings gefällt Ihnen nun nicht, dass die Zeilen sehr unterschiedlich lang werden. Als Ursache machen Sie die fehlende Trennung aus, wodurch lange Wörter komplett in die nächste Zeile rutschen und so große Lücken hinterlassen. Also wollen Sie zusätzlich Trennung nach Bedarf ermöglichen. Dies ist mit Hilfe des Pakets ragged2e leicht möglich. Dabei genügt es allerdings nicht, das Paket mit

```
\usepackage{ragged2e}
```

zu laden. Auch Option `newcommands` beim Laden des Pakets bringt keine Abhilfe. Stattdessen muss zusätzlich die *Ausrichtung* geändert werden:

```
\usepackage{ragged2e}
\setcaptionalignment[figure]{L}
```

Beachten Sie den Großbuchstabe für *Ausrichtung*.

### origlongtable

Falls die Tabellenüberschriften des longtable-Pakets (siehe [Car04]) von den KOMA-Script-Klassen nicht umdefiniert werden sollen, kann die Option `origlongtable` beim Laden der Klasse gesetzt werden. Eine Einstellung per `\KOMAoptions` oder `\KOMAoption` wird nicht unterstützt.

`listof=`*Einstellung*

v3.00 Normalerweise werden Verzeichnisse von 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 wird in der Voreinstellung für die Überschriften der Verzeichnisse die oberste verfügbare Gliederungsebene unterhalb von `\part` verwendet. Bei scrbook und scrreprt ist das die
v3.06 Kapitelebene, bei scrartcl die Abschnittsebene. Mit Hilfe der Einstellung `listof=leveldown`
v3.15 kann hingegen die nächsttiefere Gliederungsebene verwendet werden. `listof=standardlevel` schaltet bei Bedarf wieder zurück auf die voreingestellte Gliederungsebene.

**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 113 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 76 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.

Die Einstellung `listof=flat` wird automatisch aktiviert, falls die Einstellung `listof=`
v3.06 `entryprefix` verwendet wird. Normalerweise ist es nicht sinnvoll jeden Eintrag in eines der Verzeichnisse der Gleitumgebungen mit einem Präfix wie »Abbildung« oder »Tabelle« zu versehen, da 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

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 Lösung, bei der nur Kapitel mit Gleitumgebungen im jeweiligen Verzeichnis angezeigt werden, finden Sie unter [Koh15]. 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 106 erläutert wird.

v3.00

Ein 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`<br>Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen durch einen Inhaltsverzeichniseintrag des Kapitels markiert. |
| `chaptergapline`, `onelinechaptergap`<br>Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen durch einen Abstand von einer Zeile markiert. |
| `chaptergapsmall`, `smallchaptergap`<br>Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen durch einen kleinen Abstand markiert. |

...

Tabelle 3.20.: Mögliche Werte für Option `listof` (*Fortsetzung*)

`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.

`indenttextentries`, `indentunnumbered`, `numberline`

v3.12 Die Eigenschaft `numberline` (siehe Abschnitt 13.2, Seite 375) wird für die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, gesetzt. Dadurch werden nicht nummerierte Einträge linksbündig mit dem Text von nummerierten Einträgen gleicher Ebene gesetzt. Allerdings bieten die KOMA-Script-Klassen selbst keine nicht nummerierten Einträge in diese Verzeichnisse. Dies hat daher nur Auswirkungen auf entsprechende Einträge, die nicht von den Klassen selbst, aber dennoch mit Hilfe von `\addxcontentsline` (siehe Abschnitt 13.2, Seite 371) erzeugt werden.

`leftaligntextentries`, `leftalignunnumbered`, `nonumberline`

v3.12 Die Eigenschaft `numberline` (siehe Abschnitt 13.2, Seite 375) wird für die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, gelöscht. Dadurch werden nicht nummerierte Einträge linksbündig mit der Nummer von nummerierten Einträgen gleicher Ebene gesetzt. Allerdings bieten die KOMA-Script-Klassen selbst keine nicht nummerierten Einträge in diese Verzeichnisse. Dies hat daher nur Auswirkungen auf entsprechende Einträge, die nicht von den Klassen selbst, aber dennoch mit Hilfe von `\addxcontentsline` (siehe Abschnitt 13.2, Seite 371) erzeugt werden.

`leveldown`

Die Verzeichnisse werden um eine Gliederungsebene nach unten verschoben.

...

Tabelle 3.20.: Mögliche Werte für Option `listof` (*Fortsetzung*)

`nochaptergap`, `ignorechapter`
: Kapitelanfänge werden in den Verzeichnissen der Gleitumgebungen nicht markiert.

`notoc`, `nottotoc`, `plainheading`
: Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten keinen Eintrag im Inhaltsverzeichnis.

`numbered`, `totocnumbered`, `tocnumbered`, `numberedtoc`, `numberedtotoc`
: Die Verzeichnisse der Gleitumgebungen, beispielsweise das Abbildungs- und das Tabellenverzeichnis, erhalten einen Eintrag im Inhaltsverzeichnis und werden nummeriert.

`standardlevel`
: Die Verzeichnisse liegen auf der üblichen Gliederungsebene.

`toc`, `totoc`, `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 151). Darüber hinaus wirken sich indirekt die Werte `listof` und `listofnumbered` für die Option `toc` (siehe Abschnitt 3.9, Seite 75) sowie die Werte `totoc` und `numbered` 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 sogenannte 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 Dokument 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}}
```

Damit ist `\marginline` also nur eine abkürzende Schreibweise.

Für Experten sind in Abschnitt 18.1, Seite 481 Probleme bei der Verwendung von `\marginpar` dokumentiert. Diese gelten ebenso für `\marginline`. Darüber hinaus wird in Kapitel 16 ein Paket vorgestellt, mit dem sich auch Notizspalten mit eigenem Seitenumbruch realisieren lassen.

## 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 105 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 101 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 [DGS+12] 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 18.9, Seite 510 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=numbered` 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 `bibliography=nottotoc`. Siehe hierzu auch Option `toc` in Abschnitt 3.9, insbesondere die Werte `bibliographynumbered`, `bibliography` und `nobibliography` ab Seite 75.

v3.12 In einigen Fällen wird nicht das gesamte Dokument mit einem einzigen Literaturverzeichnis versehen, sondern jedes Kapitel eines mit scrbook oder scrreprt gesetzten Dokuments erhält sein eigenes Literaturverzeichnis. In diesem Fall ist es sinnvoll, wenn das Literaturverzeichnis selbst nicht auf Kapitelebene, sondern etwas tiefer auf Abschnittsebene angesiedelt wird. Dies ist mit Option `bibliography=leveldown` zu erreichen. Die Einstellung kann auch verwendet werden, wenn das Literaturverzeichnis zusammen mit anderen Verzeichnissen unter einer gemeinsamen Überschrift erscheinen soll. Daher ist diese Option auch in scrartcl verfügbar.

Eine Zusammenfassung möglicher Werte für *Einstellung* ist Tabelle 3.21 zu entnehmen. 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 Option `bibliography=totoc` oder `bibliography=numbered` 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 [Dal10]).

**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
```

Tabelle 3.21.: Vordefinierte Werte für Option `bibliography` zur Einstellung der Form des Literaturverzeichnisses

| |
|---|
| `leveldown`<br>v3.12 Das Literaturverzeichnis wird um eine Gliederungsebene nach unten verschoben. |
| `notoc`, `nottotoc`, `plainheading`<br>Das Literaturverzeichnis erhält keinen Eintrag im Inhaltsverzeichnis und wird auch nicht nummeriert. |
| `numbered`, `tocnumbered`, `totocnumbered`, `numberedtoc`, `numberedtotoc`<br>Das Literaturverzeichnis erhält einen Eintrag im Inhaltsverzeichnis und wird nummeriert. |
| `oldstyle`<br>Es wird die klassische, kompakte Formatierung gewählt, bei der `\newblock` nur einen dehnbaren horizontalen Abstand darstellt. |
| `openstyle`<br>Es wird eine untergliederte, offene Formatierung gewählt, bei der `\newblock` einen Absatz darstellt. |
| `standardlevel`<br>v3.12 Das Literaturverzeichnis liegt auf der üblichen Gliederungsebene. |
| `toc`, `totoc`, `notnumbered`<br>Das Literaturverzeichnis erhält einen Eintrag im Inhaltsverzeichnis, ohne dass es nummeriert wird. |

```
sortiert. Bei mehreren Autoren wird nach dem
ersten Autor sortiert.\par\bigskip}
```

Mit `\bigskip` wird der Abstand nach der Präambel sichergestellt.

`\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 18.2, Seite 481).

## 3.24. Stichwortverzeichnis

Das Stichwortverzeichnis ist auch unter den Bezeichnungen Index oder Register bekannt. Zur generellen Verwendung eines Stichwortverzeichnisses sei auf [DGS+12] sowie auf [Lam87] und [Keh97] verwiesen. Wird ein Paket verwendet, das selbst Anweisungen und Umgebungen für 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 [Koh14]).

index=*Einstellung*

v3.00 In der Voreinstellung `index=nottotoc` 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
v3.18 `index=totoc` erreicht werden. Soll der Index entgegen aller Gepflogenheiten sogar nummeriert werden, so verwendet man Option `index=numbered`. Siehe hierzu auch Option `toc` mit dem Wert `index` oder `indexnumbered` in Abschnitt 3.9 ab Seite 75.

Werden beispielsweise mit Hilfe von splitidx (siehe [Koh14]) mehrere Stichwortverzeichnisse erstellt, so kann es sinnvoll sein, diese unter einer gemeinsamen Überschrift zusammenzufassen.
v3.18 Um dies zu ermöglichen, kann mit `index=leveldown` das Verzeichnis eine Gliederungsebene tiefer als üblich angesiedelt werden. Bei scrbook und scrreprt ist es dann also kein Kapitel mehr, sondern ein Abschnitt, bei scrartcl entsprechend ein Unterabschnitt. Option `index=`

Tabelle 3.22.: Mögliche Werte für Option `index` zur Einstellung des Stichwortverzeichnisses

| | |
|---|---|
| `leveldown` | Der Index wird um eine Gliederungsebene nach unten verschoben. |
| `notoc`, `nottotoc`, `plainheading` | Das Stichwortverzeichnis erhält keinen Eintrag im Inhaltsverzeichnis. |
| `numbered`, `tocnumbered`, `totocnumbered`, `numberedtoc`, `numberedtotoc` | Das Stichwortverzeichnis erhält einen Eintrag im Inhaltsverzeichnis und wird nummeriert. |
| `standardlevel` | Der Index liegt auf der üblichen Gliederungsebene. |
| `toc`, `totoc`, `notnumbered` | Das Stichwortverzeichnis erhält einen Eintrag im Inhaltsverzeichnis, ohne dass es nummeriert wird. |

v3.18 v3.18 v3.18 v3.18

v3.18 `standardlevel` ist das Gegenstück dazu und hebt ein eventuell zuvor verwendetes `index=leveldown` wieder auf.

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 87).

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.

# Briefe mit Klasse scrlttr2 oder Paket scrletter

v3.15 Das Paket scrletter verstärkt KOMA-Script seit Version 3.15. Es stellt die auf Briefe ausgelegte Funktionalität von scrlttr2 auch für andere Klassen bereit. Empfohlen wird die Verwendung mit einer der KOMA-Script-Klassen scrbook, scrreprt oder scrartcl, die im vorherigen Kapitel erklärt sind. Auf Anwenderwunsch funktioniert scrletter mit geringfügigen Einschränkungen aber auch mit den Standardklassen.

Ausgangspunkt für die Entwicklung von scrletter waren Nachfragen von Anwendern, die Elemente wie Gliederungsüberschriften, Gleitumgebungen oder ein Literaturverzeichnis auch in Briefen haben wollten. Umgekehrt gab es auch Wünsche nach der Verwendung der Variablen von scrlttr2 in den übrigen KOMA-Script-Klassen. Beides ist durch eine Kombination der gewünschten KOMA-Script-Klasse mit scrletter möglich.

Gegenüber der Briefklasse hat das Briefpaket einige, kleine Änderungen, die notwendig waren, um Konflikte mit den anderen Klassen zu vermeiden. Diese betreffen vor allem die Seitenstile und sind explizit dokumentiert (siehe Abschnitt 4.13, ab Seite 240).

## 4.1. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde. Falls Sie also Abschnitt 2.4 bereits gelesen und verstanden haben, können Sie auf Seite 163 mit Abschnitt 4.2 fortfahren.

```
\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`

v3.00 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 [Tea23] oder jeder LaTeX-Einführung, beispielsweise [DGS+12], beschrieben.

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

Gegenüber der nachfolgend vorgestellten Schnittstelle zu Einstellungen von Optionen hat `\documentclass` einen Nachteil, der unbedingt zu beachten ist: Anweisungen, Längen, Zähler und ähnliches können darin leicht zerbrechen. So führt die Verwendung einer LaTeX-Länge im Wert einer Option bei dieser Anweisung bei vielen Nicht-KOMA-Script-Klassen zu einer Fehlermeldung. Wertzuweisungen mit LaTeX-Längen oder LaTeX-Zählern sollten daher nie per `\documentclass`, sondern mit den nachfolgend dokumentierten Anweisungen `\KOMAoptions` oder `\KOMAoption` vorgenommen werden.

`\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. Mit der Anweisung `\KOMAoptions` kann man wie bei `\documentclass` oder `\usepackage` 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 `\KOMAoption` der einen *`Option`* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *`Werteliste`* durch Komma voneinander getrennt.

Soll ein *`Wert`* ein Gleichheitszeichen oder ein Komma enthalten, 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`«. Siehe dazu Teil II, Abschnitt 10.2, ab Seite 329.

Mit `\KOMAoptions` oder `\KOMAoption` gesetzte Optionen erreichen sowohl die KOMA-Script-Klasse als auch alle bereits geladenen KOMA-Script-Pakete, die diese Optionen kennen. Ist eine Option oder ein Wert insgesamt unbekannt, so wird die Option einschließlich des Wertes von scrbase als fehlerhaft gemeldet.

## 4.2. Kompatibilität zu früheren Versionen von KOMA-Script

Es gilt sinngemäß, was in Abschnitt 2.5 geschrieben wurde. Allerdings existiert diese Möglich-
scrlttr2 keit bei scrlttr2 bereits seit Version 2.9t, während scrletter sie nicht bietet, sondern sozusagen immer von `version=last` ausgeht. Falls Sie also Abschnitt 2.5 bereits gelesen und verstanden haben, können Sie in Abschnitt 4.3 auf Seite 165 fortfahren.

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 führen aber Verbesserungen und Korrekturen an der Klasse zu Änderungen im Verhalten, inbesondere beim Umbruch. Dies ist jedoch manchmal eher unerwünscht.

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

scrlttr2 Seit Version 2.9t besteht bei scrlttr2 die Wahl, ob eine Quelldatei, soweit irgend möglich, auch zukünftig bei einem LaTeX-Lauf zu exakt demselben Ergebnis führen soll oder ob sie jeweils entsprechend der Anpassungen der neusten Version der Klasse 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 rückwirkende Kompatibilität verzichtet. Wird die Option ohne Wertangabe verwendet, v3.01a 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.

**Beispiel:** Die Beispielbriefe dieses Kapitels sollen alle Möglichkeiten nutzen, die in der neusten Version von KOMA-Script zur Verfügung stehen. Dazu muss beim Laden der Klasse die Kompatibilität entsprechend gesetzt werden:

```
\documentclass[version=last]{scrlttr2}
```

Hier wurde einfach mit dem symbolischen Wert `last` die neuste Version gewählt.

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

## 4.3. Entwurfsmodus

scrlttr2 Für scrlttr2 gilt sinngemäß, was in Abschnitt 3.3 geschrieben wurde. Falls Sie also Abschnitt 3.3 bereits gelesen und verstanden haben, können Sie nach dem Ende dieses Abschnitts auf Seite 165 mit Abschnitt 4.4 fortfahren. Das Paket scrletter bietet selbst keinen Entwurfsmodus, sondern verlässt sich diesbezüglich auf die verwendete Klasse.

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
overfullrule=Ein-Aus-Wert
```

scrlttr2 Mit Option `draft` 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 41 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 [Sch13].

v3.25 Da Option `draft` bei verschiedenen Paketen zu allerlei unerwünschten Effekten führen kann, bietet KOMA-Script die Möglichkeit, die Markierung für überlange Zeilen auch über Option `overfullrule` zu steuern. Auch hier gilt, dass bei aktivierter Option die Markierung angezeigt wird.

## 4.4. 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 scrlttr2 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 selbstständig. 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 4.1, ab Seite 162.

Einige Einstellungen der KOMA-Script-Klassen haben 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 26 zu finden.

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- und Rückseite bedruckt werden. Daher spielt bei Briefen normalerweise auch der vertikale Ausgleich keine Rolle. Sollten Sie diesen trotzdem benötigen sei auf die in Abschnitt 3.4, Seite 58 erklärten Anweisungen `\raggedbottom` und `\flushbottom` verwiesen.

## 4.5. 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 192). 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. Hauptunterschied zwischen Variablen und Anweisungen ist, dass Anweisungen üblicherweise Aktionen auslösen, während der Inhalt einer Variablen normalerweise aus einem Text besteht, der dann von einer Anweisung ausgegeben wird. Er kann aber auch von mehreren Anweisungen an unterschiedlichen Stellen verwendet und so mehrfach ausgegeben werden. 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 Klasse und dem Paket bereits definierten und von ihnen verwendeten Variablen in den nachfolgenden Abschnitten. Tabelle 4.1 gibt eine Übersicht über alle definierten Variablen.

Tabelle 4.1.: Von scrlttr2 und scrletter 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 215)

`backaddress`
: Rücksendeadresse für Fensterbriefumschläge (Abschnitt 4.10, Seite 215)

`backaddressseparator`
: Trennzeichen innerhalb der Rücksendeadresse (Abschnitt 4.10, Seite 215)

...

Tabelle 4.1.: Von scrlttr2 und scrletter unterstützte Variablen (*Fortsetzung*)

`ccseparator`
: Trennzeichen zwischen Verteilertitel und Verteiler (Abschnitt 4.7, Seite 185)

`customer`
: Geschäftszeilenfeld »Kundennummer« (Abschnitt 4.10, Seite 224)

`date`
: Datum (Abschnitt 4.10, Seite 223)

`emailseparator`
: Trennzeichen zwischen E-Mail-Bezeichnung und E-Mail-Adresse (Abschnitt 4.10, Seite 208)

`enclseparator`
: Trennzeichen zwischen Anlagetitel und Anlagen (Abschnitt 4.7, Seite 187)

`faxseparator`
: Trennzeichen zwischen Faxbezeichner und Faxnummer (Abschnitt 4.10, Seite 208)

`firstfoot`
v3.08
: Seitenfuß des Briefbogens (Abschnitt 4.10, Seite 235)

`firsthead`
v3.08
: Kopf des Briefbogens (Abschnitt 4.10, Seite 213)

`fromaddress`
: Absenderadresse ohne Absendername (Abschnitt 4.10, Seite 203)

`frombank`
: Bankverbindung des Absenders (Abschnitt 4.10, Seite 237)

`fromemail`
: E-Mail-Adresse des Absenders (Abschnitt 4.10, Seite 208)

`fromfax`
: Faxnummer des Absenders (Abschnitt 4.10, Seite 208)

`fromlogo`
: Anweisungen zum Setzen des Absenderlogos (Abschnitt 4.10, Seite 211)

`frommobilephone`
v3.12
: Handynummer des Absenders (Abschnitt 4.10, Seite 208)

...

Tabelle 4.1.: Von scrlttr2 und scrletter unterstützte Variablen (*Fortsetzung*)

`fromname`
: vollständiger Absendername (Abschnitt 4.10, Seite 203)

`fromphone`
: Telefonnummer des Absenders (Abschnitt 4.10, Seite 208)

`fromurl`
: eine URL des Absenders (Abschnitt 4.10, Seite 208)

`fromzipcode`
: Postleitzahl des Absenders für den Port-Payé-Kopf bei `addrfield=PP` (Abschnitt 4.10, Seite 215)

`invoice`
: Geschäftszeilenfeld »Rechnungsnummer« (Abschnitt 4.10, Seite 224)

`location`
: erweiterte Absenderangabe (Abschnitt 4.10, Seite 221)

`mobilephoneseparator`
: Trennzeichen zwischen Handybezeichner und Handynummer (Abschnitt 4.10, Seite 208)

`myref`
: Geschäftszeilenfeld »Mein Zeichen« (Abschnitt 4.10, Seite 224)

`nextfoot`
v3.08
: Seitenfuß im Seitenstil `headings` oder `myheadings` (Abschnitt 4.13, Seite 244)

`nexthead`
v3.08
: Kopf im Seitenstil `headings` oder `myheadings` (Abschnitt 4.13, Seite 244)

`phoneseparator`
: Trennzeichen zwischen Telefonbezeichner und Telefonnummer (Abschnitt 4.10, Seite 208)

`place`
: Ort (Abschnitt 4.10, Seite 215)

`placeseparator`
: Trennzeichen zwischen Ort und Datum (Abschnitt 4.10, Seite 225)

...

Tabelle 4.1.: Von scrlttr2 und scrletter unterstützte Variablen (*Fortsetzung*)

`PPdatamatrix`
: Anweisungen zum Setzen einer Data-Matrix bei der Einstellung `addrfield=PP` (Abschnitt 4.10, Seite 215)

`PPcode`
: Code zur Identifizierung des Absenders bei Einstellung `addrfield=PP` (Abschnitt 4.10, Seite 215)

`signature`
: Signatur unter Unterschrift und Grußformel (Abschnitt 4.10.7, Seite 232)

`specialmail`
: Versandart (Abschnitt 4.10, Seite 215)

`subject`
: Betreff (Abschnitt 4.10, Seite 228)

`subjectseparator`
: Trennzeichen zwischen Betrefftitel und Betreff (Abschnitt 4.10, Seite 228)

`title`
: Brieftitel (Abschnitt 4.10, Seite 228)

`toaddress`
: Empfängeradresse ohne Empfängername (Abschnitt 4.10, Seite 215)

`toname`
: vollständiger Empfängername (Abschnitt 4.10, Seite 215)

`yourmail`
: Geschäftszeilenfeld »Ihr Schreiben« (Abschnitt 4.10, Seite 224)

`yourref`
: Geschäftszeilenfeld »Ihr Zeichen« (Abschnitt 4.10, Seite 224)

`zipcodeseparator`
: Trennzeichen zwischen der Bezeichnung und dem Inhalt der Variablen `fromzipcode` (Abschnitt 4.10, Seite 215)

\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 mit geändert werden. Mit der Sternvariante \setkomavar* kann auch nur die *Bezeichnung* der Variablen *Name* gesetzt werden.

**Beispiel:** In Briefen ist es üblich, den Absender im Briefkopf stehen zu haben. Dazu muss KOMA-Script 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 an den Stellen, an denen KOMA-Script 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.

Ein leeres optionales Argument ist also nicht gleichbedeutend mit dem Weglassen des optionalen Arguments.

```
\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 19.1, Seite 513 zu entnehmen.

```
\Ifkomavar{Name}{Dann-Teil}{Sonst-Teil}
```

v3.08

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 4.20 ab Seite 255) 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}
```

v3.28

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.

## 4.6. Pseudolängen

Längen werden bei LaTeX mit den drei Anweisungen `\newlength`, `\setlength` und `\addtolength` 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.

Bitte beachten Sie unbedingt, dass die Pseudolängen zwar intern als Makros implementiert sind, bei den Befehlen zur Nutzung der Pseudolängen jedoch nur die Namen anzugeben sind. Diese werden wie die Namen von LaTeX-Zählern und im Gegensatz zu Makros oder echten Längen ohne umgekehrten Schrägstrich geschrieben!

Historisch gesehen arbeitet TeX mit einem festen Satz an Registern. Es gibt jeweils 256 Register für Token, Boxen, Zähler, Abstände (englisch: *skip*) und Größen (englisch: *dimension*). Für LaTeX-Längen, die mit `\newlength` angefordert werden, werden Abstandsregister belegt. Sind alle diese

Register verbraucht, kann man keine weiteren Längen definieren. Sowohl scrlttr2 als auch scrletter würden 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. Dies war 2002 der Hauptgrund für die Einführung der Pseudolängen bei scrlttr2.

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. Bei scrletter wurde das Konzept der Pseudolängen aus Gründen der Kompatibilität übernommen.

Eine Auflistung aller von KOMA-Script definierten und verwendeten Pseudolängen ist Tabelle 4.2 zu entnehmen. Dabei ist auch angegeben, wo in den nachfolgenden Abschnitten nähere Erklärungen zu der jeweiligen Pseudolänge zu finden sind.

Abbildung 4.1 auf Seite 177 zeigt eine schematische Darstellung der wichtigsten Abstände auf dem Briefbogen. Dabei sind neben den Pseudolängen für die veränderbaren Abstände zusätzlich in heller Schrift die Längen angegeben, die für einige wenige fest programmierte Abstände verwendet werden. Aus Gründen der Übersichtlichkeit wurde in der Darstellung auf einige weniger häufig benötigte Pseudolängen jedoch verzichtet.

Tabelle 4.2.: Von scrlttr2 und scrletter verwendete Pseudolängen

`backaddrheight`
: Höhe der Rücksendeadresse am oberen Rand des Anschriftfeldes (Abschnitt 4.10.3, Seite 220)

`bfoldmarklength`
: Länge der unteren horizontalen Faltmarke (Abschnitt 4.10.1, Seite 200)

`bfoldmarkvpos`
: Abstand der unteren horizontalen Faltmarke von der oberen Kante des Papiers (Abschnitt 4.10.1, Seite 200)

`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 4.10.8, Seite 237)

`firstfootvpos`
: Abstand des Brieffußes von der oberen Kante des Papiers (Abschnitt 4.10.8, Seite 237)

...

Tabelle 4.2.: Von scrlttr2 und scrletter verwendete Pseudolängen (*Fortsetzung*)

`firstfootwidth`
: Breite des Brieffußes (Abschnitt 4.10.8, Seite 237)

`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 4.10.2, Seite 202)

`firstheadvpos`
: Abstand des Briefkopfes von der oberen Kante des Papiers (Abschnitt 4.10.2, Seite 202)

`firstheadwidth`
: Breite des Briefkopfes (Abschnitt 4.10.2, Seite 202)

`foldmarkhpos`
: Abstand der horizontalen Faltmarken von der linken Kante des Papiers (Abschnitt 4.10.1, Seite 201)

`foldmarkvpos`
: Abstand der vertikalen Faltmarken von der oberen Kante des Papiers (Abschnitt 4.10.1, Seite 201)

`fromrulethickness`
: Dicke einer optionalen horizontalen Linie im Briefkopf (Abschnitt 4.10.2, Seite 207)

`fromrulewidth`
: Länge einer optionalen horizontalen Linie im Briefkopf (Abschnitt 4.10.2, Seite 207)

`lfoldmarkhpos`
: Abstand der vertikalen Faltmarke von der linken Kante des Papiers (Abschnitt 4.10.1, Seite 201)

`lfoldmarklength`
: Länge der vertikalen Faltmarke (Abschnitt 4.10.1, Seite 201)

`locheight`
: Höhe der Absenderergänzung, falls der Wert nicht 0 ist; bei 0 wird stattdessen `toaddrheight` verwendet (Abschnitt 4.10.4, Seite 222)

...

Tabelle 4.2.: Von scrlttr2 und scrletter verwendete Pseudolängen (*Fortsetzung*)

`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 4.10.4, Seite 222)

`locvpos`
: Abstand der Absenderergänzung von der oberen Papierkante, falls der Wert nicht 0 ist; bei 0 wird stattdessen `toaddrvpos` verwendet (Abschnitt 4.10.4, Seite 222)

`locwidth`
: Breite des Feldes für die Absenderergänzung, wobei bei einem Wert von 0 die Breite automatisch aufgrund der in Abschnitt 4.10, Seite 221 beschriebenen Option `locfield` berechnet wird (Abschnitt 4.10.4, Seite 222)

`mfoldmarklength`
: Länge der mittleren horizontalen Faltmarke (Abschnitt 4.10.1, Seite 200)

`mfoldmarkvpos`
: Abstand der mittleren horizontalen Faltmarke von der oberen Kante des Papiers (Abschnitt 4.10.1, Seite 200)

`pfoldmarklength`
: Länge der Lochermarke (Abschnitt 4.10.1, Seite 200)

`PPdatamatrixvskip`
: vertikaler Abstand zwischen Port-Payé-Kopf und Data-Matrix bei `addrfield=PP` (Abschnitt 4.10.3, Seite 220)

`PPheadheight`
: Höhe für den Port-Payé-Kopf (Abschnitt 4.10.3, Seite 220)

`PPheadwidth`
: Breite des linken Port-Payé-Feldes bei `addrfield=PP` (Abschnitt 4.10.3, Seite 220)

`refaftervskip`
: vertikaler Abstand nach der Geschäftszeile (Abschnitt 4.10.5, Seite 227)

`refhpos`
: Abstand der Geschäftszeile von der linken Papierkante, wobei bei einem Wert von 0 automatisch relativ zur Papierbreite zentriert wird (Abschnitt 4.10.5, Seite 226)

...

Tabelle 4.2.: Von scrlttr2 und scrletter verwendete Pseudolängen (*Fortsetzung*)

`refvpos`
: Abstand der Geschäftszeile von der oberen Kante des Papiers (Abschnitt 4.10.5, Seite 226)

`refwidth`
: Breite der Geschäftszeile (Abschnitt 4.10.5, Seite 226)

`sigbeforevskip`
: vertikaler Abstand zwischen Gruß und Signatur (Abschnitt 4.10.7, Seite 234)

`sigindent`
: Einzug der Signatur gegenüber dem Textkörper (Abschnitt 4.10.7, Seite 234)

`specialmailindent`
: linker Einzug der Versandart innerhalb des Anschriftfeldes (Abschnitt 4.10.3, Seite 220)

`specialmailrightindent`
: rechter Einzug der Versandart innerhalb des Anschriftfeldes (Abschnitt 4.10.3, Seite 220)

`subjectaftervskip`
: vertikaler Abstand nach dem Betreff (Abschnitt 4.10.6, Seite 231)

`subjectbeforevskip`
: zusätzlicher vertikaler Abstand vor dem Betreff (Abschnitt 4.10.6, Seite 231)

`subjectvpos`
: Abstand des Betreffs von der oberen Kante des Papiers, wobei ein Wert von 0 stattdessen den Betreff gemäß Option `subject` setzt (Abschnitt 4.10.6, Seite 231)

`tfoldmarklength`
: Länge der oberen horizontalen Faltmarke (Abschnitt 4.10.1, Seite 200)

`tfoldmarkvpos`
: Abstand der oberen horizontalen Faltmarke von der oberen Kante des Papiers (Abschnitt 4.10.1, Seite 200)

`toaddrheight`
: Höhe des Anschriftfeldes (Abschnitt 4.10.3, Seite 219)

...

Tabelle 4.2.: Von scrlttr2 und scrletter verwendete Pseudolängen (*Fortsetzung*)

| | |
|---|---|
| `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 4.10.3, Seite 218) |
| `toaddrindent` | linker und rechter Einzug der Anschrift innerhalb des Anschriftfeldes (Abschnitt 4.10.3, Seite 219) |
| `toaddrvpos` | Abstand des Anschriftfeldes von der oberen Kante des Papiers (Abschnitt 4.10.3, Seite 218) |
| `toaddrwidth` | Breite des Anschriftfeldes (Abschnitt 4.10.3, Seite 219) |

`\newplength{`*`Name`*`}`

v3.26

Mit Hilfe dieser Anweisung wird eine neue Pseudolänge definiert. Die neue Pseudolänge ist dann über ihren *Namen* eindeutig identifiziert. Jeder Name kann also nur einmal vergeben werden.

Da der Anwender selbst normalerweise keine eigenen Pseudolängen definieren muss, handelte es sich bei diesem Befehl bis KOMA-Script 3.25 um keine Benutzeranweisung. Stattdessen existierte bis dahin nur `\@newplength` mit derselben Funktionalität. Diese sollte nicht mehr verwendet werden.

`\Ifplength{`*`Pseudolänge`*`}{`*`Dann-Code`*`}{`*`Sonst-Code`*`}`

v3.27

Mit dieser Anweisung kann geprüft werden, ob eine *Pseudolänge* definiert ist. Ist dies der Fall, so wird der *Dann-Code* ausgeführt, anderenfalls wird der *Sonst-Code* ausgeführt.

`\useplength{`*`Name`*`}`

Mit Hilfe dieser Anweisung wird auf den Wert der Pseudolänge mit dem angegebenen *Namen* zugegriffen. Anwender benötigen dies eher selten. Innerhalb von `lco`-Dateien (siehe Abschnitt 4.20 ab Seite 255) wird die Anweisung häufig bei der Berechnung abhängiger Pseudolängen verwendet.

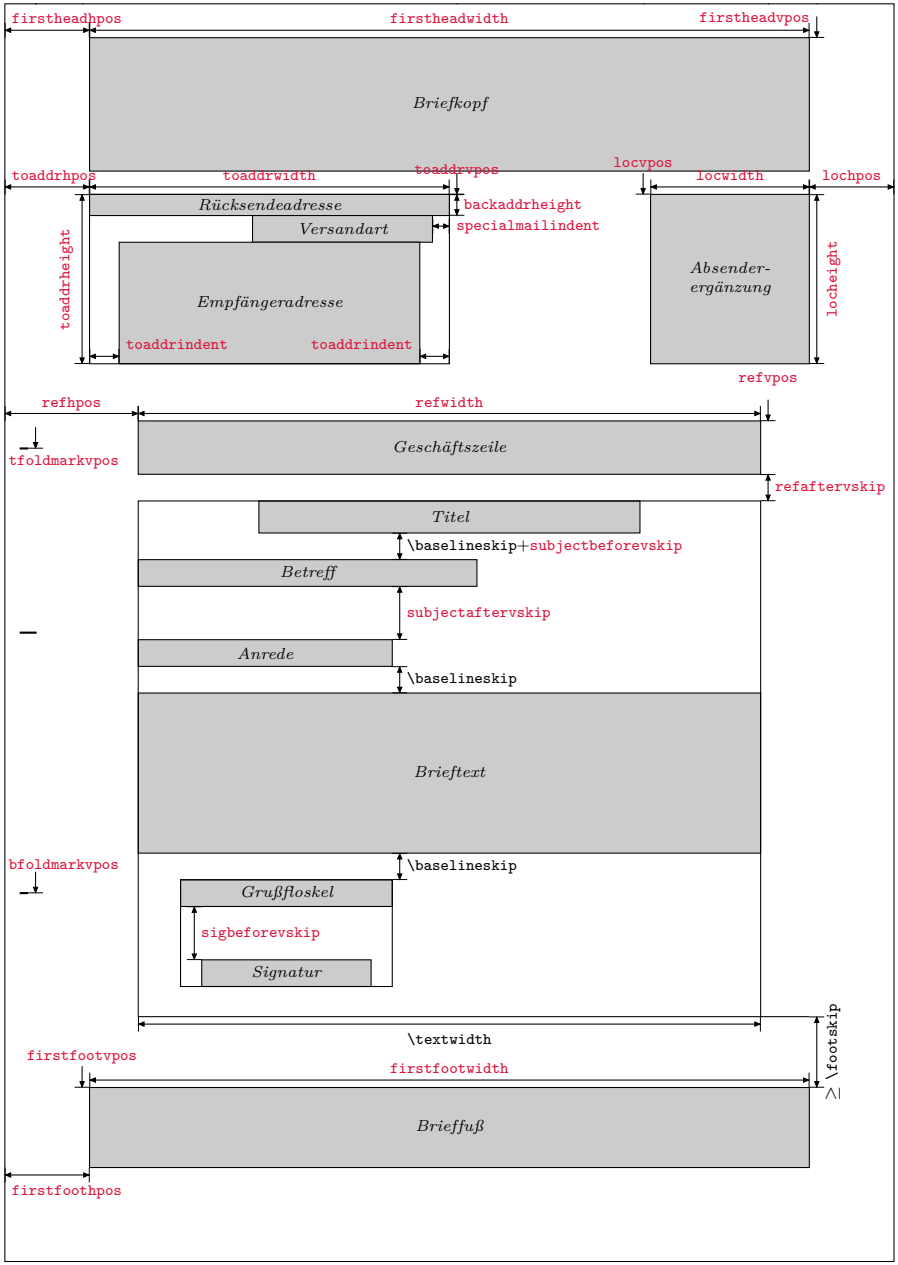



Abbildung 4.1.: Schematische Darstellung der wichtigsten Pseudolängen für den Briefbogen angelehnt an das Ergebnis bei Verwendung von `DIN.1co`

```
\setplength[Faktor]{Pseudolänge}{Wert}
\addtoplength[Faktor]{Pseudolänge}{Wert}
```

v3.26 Mit Hilfe von \setplength kann einer *Pseudolänge* das Vielfache eines *Wertes* zugewiesen werden. Der *Faktor* wird dabei als optionales Argument übergeben (siehe auch \setlengthtoplength, Seite 178).

Mit \addtoplength kann man zu einer *Pseudolänge* das Vielfache eines *Wertes* addieren. Auch dabei wird der *Faktor* als optionales Argument übergeben.

Um einer *Pseudolänge* das Vielfache einer anderen Pseudolänge zuzuweisen oder zu ihr zu addieren, verwendet man innerhalb von *Wert* die Anweisung \useplength. Um von einer *Pseudolänge* den Wert einer anderen *Pseudolänge* zu subtrahieren, verwendet man gleichzeitig als *Faktor* ein Minuszeichen oder -1 oder einen anderen negativen Faktor.

Da der Anwender selbst normalerweise keine Pseudolängen ändern muss, handelte es sich bis KOMA-Script 3.25 bei diesen Befehlen um keine Benutzeranweisungen. Stattdessen existierten bis dahin nur \@setplength und \@addtoplength mit derselben Funktionalität. Diese sollten nicht mehr verwendet werden.

```
\setplengthtowidth[Faktor]{Pseudolänge}{Inhalt}
\setplengthtoheight[Faktor]{Pseudolänge}{Inhalt}
\setplengthtodepth[Faktor]{Pseudolänge}{Inhalt}
\setplengthtototalheight[Faktor]{Pseudolänge}{Inhalt}
```

v3.26 Die ersten drei Anweisungen sind vergleichbar mit \settowidth, \settoheight und \settodepth aus dem LaTeX-Kern, setzen aber keine Länge, sondern eine *Pseudolänge*. Entsprechend \setplength sind sie ebenfalls um einen optionalen *Faktor* erweitert. Sie setzen also eine *Pseudolänge* auf die Breite, Höhe oder Tiefe von *Inhalt* multipliziert mit dem optional angegebenen *Faktor*. Die zusätzliche Anweisung \setplengthtototalheight setzt die *Pseudolänge* auf die Summe der Höhe und Tiefe von *Inhalt* multipliziert mit dem optionalen *Faktor*.

```
\setlengthtoplength[Faktor]{Länge}{Pseudolänge}
\addtolengthplength[Faktor]{Länge}{Pseudolänge}
```

Mit der Anweisung \setlengthtoplength kann man einer LaTeX-*Länge* das Vielfache einer *Pseudolänge* zuweisen. Auch hier wird ein *Faktor* nicht direkt der *Pseudolänge* vorangestellt, sondern als optionales Argument übergeben. Mit \addtolengthplength wird die mit *Faktor* multiplizierte *Pseudolänge* zur *Länge* addiert.

## 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 Brief-

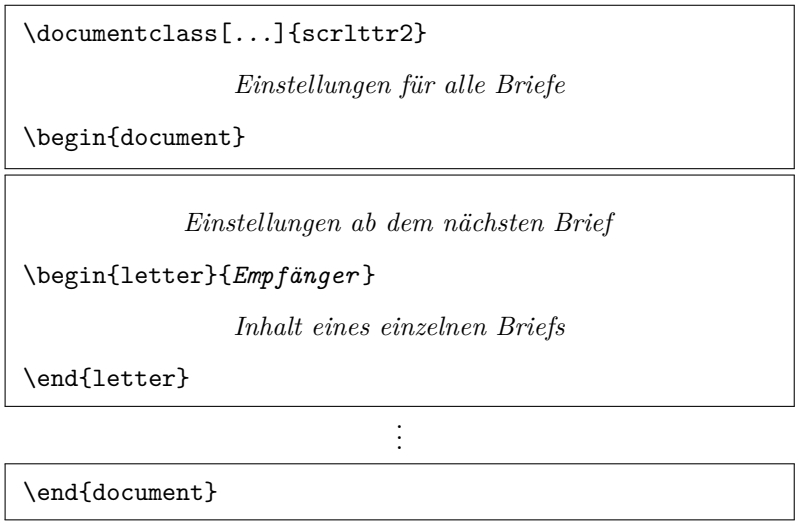



Abbildung 4.2.: Genereller Aufbau eines Briefdokuments mit beliebig vielen einzelnen Briefen (den Aufbau eines einzelnen Briefs zeigt Abbildung 4.3)

dokument mehrere Briefe enthalten. Wie in Abbildung 4.2 veranschaulicht, 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.

Bei Verwendung von scrletter ändert sich lediglich, dass eine andere Klasse geladen und dafür zusätzlich `\usepackage{scrletter}` noch vor den Einstellungen für alle Briefe einzufügen ist. Für das Setzen von Optionen für scrletter sei auf Abschnitt 4.1, ab Seite 162 verwiesen.

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 [BB13]) verwendet wird oder sprachabhängige Variablen von scrlttr2 oder scrletter 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 4.3 verdeutlicht wird, bestehen die einzelnen Briefe wiederum aus einer Einleitung, dem eigentlichen Brieftext und einem Schlussteil. In der Einleitung werden alle Einstellungen vorgenommen, die ab diesem oder nur für diesen einen Brief gelten sollen. Es sei ausdrücklich darauf hingewiesen, dass Einstellungen, die vor oder zwischen den einzelnen Briefen getroffen werden, für alle nachfolgenden Briefe gelten. 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.

Bei Verwendung von scrletter spricht nichts dagegen, vor, zwischen oder nach Briefen weitere

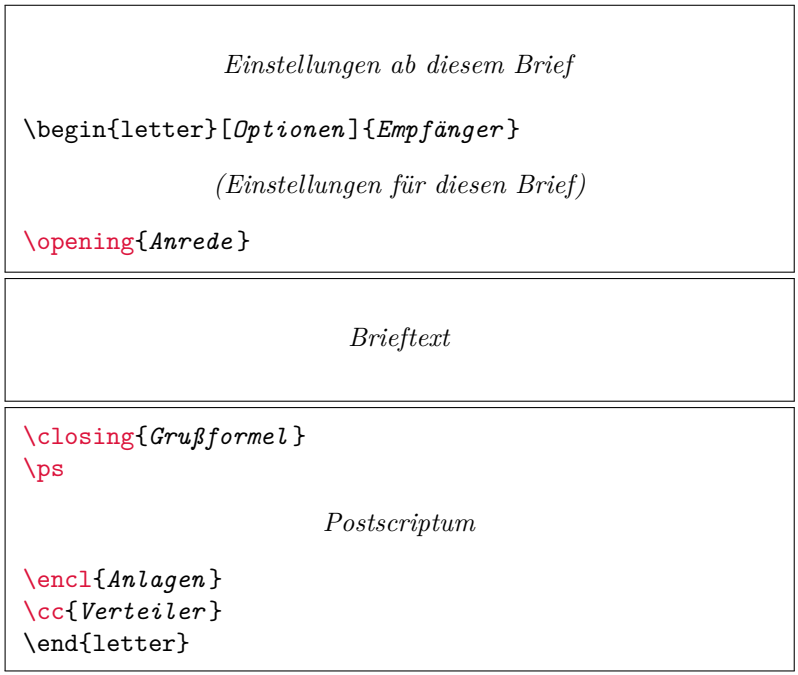



Abbildung 4.3.: Genereller Aufbau eines einzelnen Briefes innerhalb eines Briefdokuments (siehe Abbildung 4.2)

Dokumententeile einzufügen, die nicht im Briefkontext stehen sollen. So kann man beispielsweise Anschreiben und Lebenslauf in einem Dokument zusammenfassen.

`\begin{letter}[`*`Optionen`*`]{`*`Empfänger`*`}`...`\end{letter}`

Die Briefumgebung `letter` ist der Hauptzweck der Briefklasse und des Briefpakets. Bei scrlttr2 und scrletter kann man der Briefumgebung *Optionen* mit auf den Weg geben, die dann per `\KOMAoptions` nur für diese Umgebung ausgeführt werden.

Der *Empfänger* wird als obligatorischer Parameter an die Umgebung übergeben. Dabei dient der doppelte Backslash als Trennzeichen zwischen einzelnen Teilen der Anschrift. Diese 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 182.

v3.27

Briefe werden immer einspaltig und ohne vertikalen Ausgleich gesetzt. Letzteres kann man mit Hilfe von \AtBeginLetter und der in Abschnitt 3.4, Seite 58 erklärten Anweisungen \flushbottom ändern.

```
\AtBeginLetter{Anweisungen}
\AtEndLetter{Anweisungen}
```

Wie in [Tea06] erwähnt, gibt es bei LaTeX die Möglichkeit, während des LaTeX-Laufs eines Dokuments zusätzliche *Anweisungen* ausführen zu lassen. Zu diesem Zweck stellt der LaTeX-Kern die Anweisungen \AtEndOfClass und \AtBeginDocument zur Verfügung. Man nennt solche Eingriffspunkte auch *hooks*, also Haken. KOMA-Script fügt zwei weitere Haken hinzu,

v2.95

die mit \AtBeginLetter und \AtEndLetter mit Inhalt versehen werden können. Wie man schon daran erkennt, dass die LaTeX-Kern-Anweisungen für Haken nicht in [Tea23] sondern in [Tea06] dokumentiert sind, sind diese Anweisungen eigentlich eher für Paket- und Klassenautoren gedacht. Bei Briefen kann es jedoch sinnvolle Anwendungen für die beiden neuen Haken auch auf Benutzerebene geben.

**Beispiel:** Angenommen, Sie haben mehrere Briefe in einem Dokument und verwenden eine eigene Anweisung, um in den Briefen einen Fragebogen zu setzen. Die Fragen werden mit Hilfe eines Zählers nummeriert. Da KOMA-Script diesen Zähler nicht kennt, würde er auch im Gegensatz zur Seitenzahl am Anfang eines neuen Briefes nicht zurückgesetzt. Bei 10 Fragen je Brief, hätte damit die erste Frage im fünften Brief die Nummer 41 statt der Nummer 1. Sie lösen das, indem Sie am Anfang jedes Briefes den Zähler zurücksetzen lassen:

```
\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 [Car99b]).

letter
\thisletter
\letterlastpage

v3.19 Für den Fall, dass sich mehrere Briefe in einem Dokument befinden, werden die Briefe intern von KOMA-Script durchnummeriert. Hierfür ist seit Version 3.19 der Zähler `letter` definiert, der mit jedem `\begin{letter}` referenzierbar um eins erhöht wird.

**Beispiel:** Kommen wir auf das Beispiel zu `\AtBeginLetter` zurück. Statt den Zähler explizit innerhalb von `\begin{letter}` zurückzusetzen, kann dies auch implizit erfolgen, indem der Zähler `Frage` abhängig von `letter` definiert wird:

```
\newcounter{Frage}[letter]
\newcommand{\Frage}[1]{%
  \refstepcounter{Frage}\par
  \noindent\begin{tabularx}{\textwidth}{l@{}X}
    \theFrage:~ & #1\\
  \end{tabularx}%
}%
```

Damit wird der Zähler automatisch zu Beginn jedes Briefs wieder auf Null zurückgesetzt, so dass die erste Frage in jedem Brief wieder mit der Nummer Eins beginnt.

Will man sich den aktuellen Wert von `letter` ausgeben lassen, so ist das wie gewohnt mit `\theletter` möglich. Wie bereits erwähnt, ist der Zähler aber auch referenzierbar. Das bedeutet, man könnte am Anfang eines Briefes mit `\label{`*`Labelname`*`}` ein Label setzen und mit `\ref{`*`Labelname`*`}` dann an beliebiger Stelle im Dokument darauf verweisen. Innerhalb des Briefes selbst erhält man dasselbe Ergebnis auch ganz ohne Label mit `\thisletter`.

Für Label innerhalb von Serienbriefen ist es notwendig, diesen einen über alle Briefe hinweg eindeutigen Namen zu geben. Auch dafür kann `\thisletter` verwendet werden. Intern arbeitet KOMA-Script für diesen Zweck ebenfalls mit `\thisletter`, um auf der letzten Seite eines jeden Briefes ein Label zu setzen. Dadurch ist es möglich, mit `\letterlastpage` jederzeit innerhalb des Briefes die Nummer der letzten Seite des Briefes auszugeben. Da `\letterlastpage` über `\label` und `\pageref` arbeitet, ist die Ausgabe allerdings erst nach mehreren LaTeX-Läufen – meist zwei oder drei – gültig. Achten Sie gegebenenfalls auf entsprechende *Rerun*-Meldungen in der Terminal-Ausgabe oder der `log`-Datei.

\opening{*Anrede*}

Dies ist eine der wichtigsten Anweisungen in Briefen. Vordergründig wird damit die *`Anrede`*, beispielsweise »Sehr geehrte Frau ...«, gesetzt. Tatsächlich setzt diese Anweisung aber auch alle Elemente des Briefbogens wie die Faltmarken, den Briefkopf, die Anschrift, die Absendeergänzung, 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.

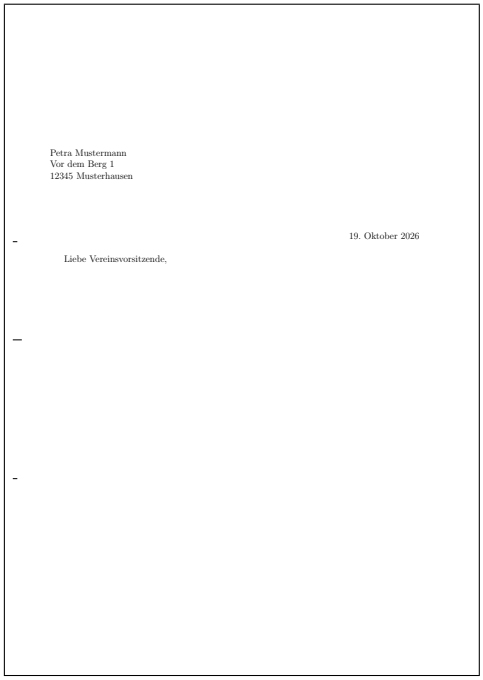



Abbildung 4.4.: Ergebnis eines minimalistischen Briefes nur mit Anschrift und Anrede (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

**Beispiel:** Kommen wir auf das Beispiel von Seite 180 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 4.4.

`\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 den Inhalt der Variablen `signature` als Signatur. Näheres zur Signatur und deren Konfiguration ist Abschnitt 4.10.7 ab Seite 232 zu entnehmen.

**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 jedes 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 4.5 aus.

`\ps`

Diese Anweisung schaltet 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. KOMA-Script 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 jedes halbe Jahr vorsieht. Ich fordere den
```

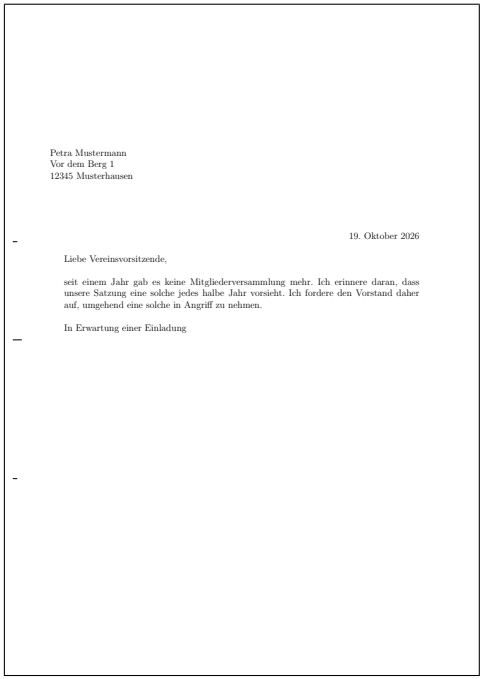



Abbildung 4.5.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text und Grußfloskel (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

```
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.
\end{letter}
\end{document}
```

sieht dann wie in Abbildung 4.6 aus.

`\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 die *`Bezeichnung`* 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. Sollen die einzelnen Einträge untereinander gesetzt werden, können sie durch doppelten Backslash voneinander getrennt angegeben werden.

**Beispiel:** Der Beispielbrief soll dieses Mal nicht nur an die Vorsitzende, sondern mit Verteiler auch an alle Mitglieder des Vereins gehen:

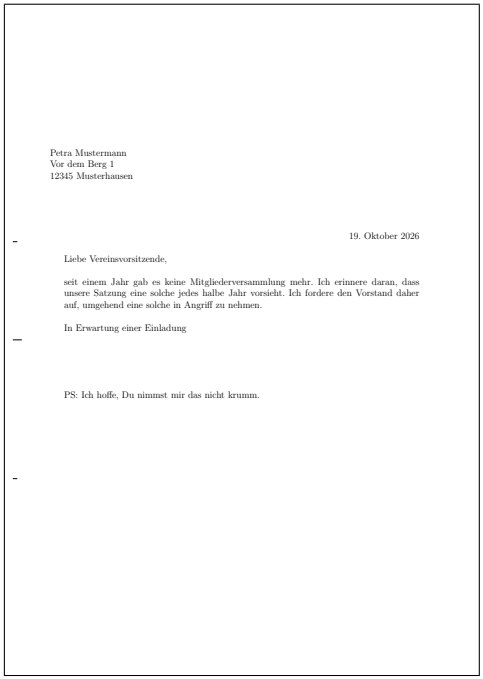



Abbildung 4.6.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel und Postskriptum (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

```
\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 jedes 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 4.7 zu sehen.

Vor dem Verteiler wird automatisch ein Abstand eingefügt.

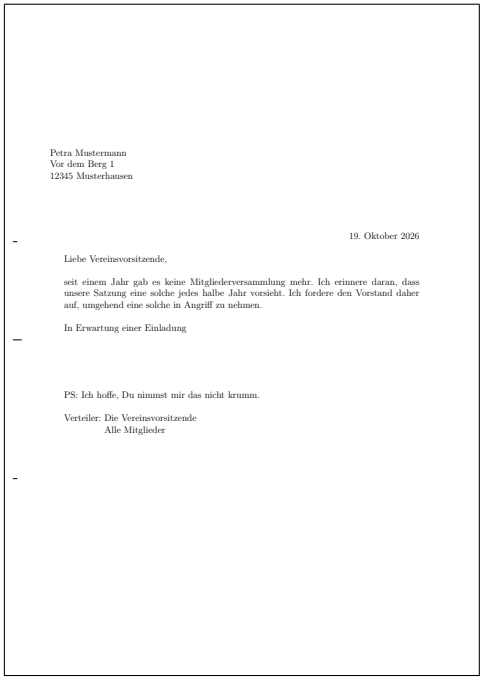



Abbildung 4.7.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum und Verteiler (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

```
\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\\
    12345 Musterhausen%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche jedes halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
```

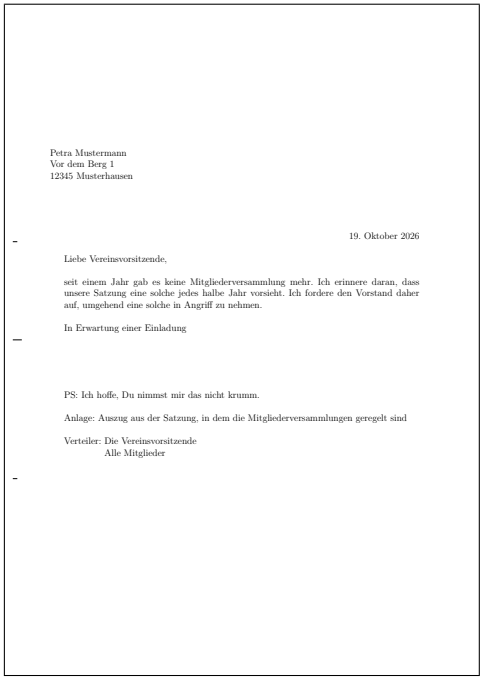



Abbildung 4.8.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen und Verteiler (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe)

```
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
\ps PS: Ich hoffe, Du nimmst mir das nicht krumm.
\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 4.8 zu sehen.

## 4.8. Wahl der Schriftgröße für das Dokument

scrlttr2

Für scrlttr2 gilt sinngemäß, was in Abschnitt 3.5 geschrieben wurde. Paket scrletter bietet selbst hingegen keine Schriftgrößenauswahl, sondern verlässt sich diesbezüglich vollständig auf die verwendete Klasse. Falls Sie also Abschnitt 3.5 bereits gelesen und verstanden haben, können Sie beim Beispiel am Ende dieses Abschnitts auf Seite 189 fortfahren. Wenn Sie dagegen scrletter verwenden, können Sie auch direkt zu Abschnitt 4.9 auf Seite 190 springen.

`fontsize=`*`Größe`*

scrlttr2

Während von den Standardklassen und den meisten anderen Klassen nur eine sehr beschränkte Anzahl an Schriftgrößen unterstützt wird, bietet scrlttr2 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 `\normalsize` und die davon abhängigen Schriftgrößen der Befehle `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\large`, `\Large`, `\LARGE`, `\huge` und `\Huge` 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 2.6, Seite 40). Wird diese Neuberechnung jedoch vorgenommen, so erfolgt sie auf Basis der jeweils gültigen Grundschriftgröße. Die Auswirkungen des Wechsels der Grundschriftgröße auf zusätzlich geladene Pakete oder die verwendete Klasse sind von diesen Paketen und der Klasse abhängig. Es können also Fehler auftreten, die nicht als Fehler von KOMA-Script angesehen werden.

Diese Option sollte keinesfalls als Ersatz für `\fontsize` (siehe [Tea05]) missverstanden werden. Sie sollte auch nicht anstelle einer der von der Grundschrift abhängigen Schriftgrößenanweisungen, `\tiny` bis `\Huge`, verwendet werden! Die Verwendung innerhalb eines Absatzes ist aus diesem Grund auch explizit verboten! 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 jedes 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%
  }
\opening{Liebe Vereinsvorsitzende,}
seit einem Jahr gab es keine Mitgliederversammlung
mehr. Ich erinnere daran, dass unsere Satzung eine
solche jedes 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 4.9.

## 4.9. Textauszeichnungen

Es gilt sinngemäß, was in Abschnitt 3.6 geschrieben wurde. Falls Sie also Abschnitt 3.6 bereits gelesen und verstanden haben, können Sie sich auf Tabelle 4.3, Seite 192 beschränken und ansonsten auf Seite 196 mit Abschnitt 4.10 fortfahren.

LaTeX verfügt über eine ganze Reihe von Anweisungen zur Textauszeichnung. Näheres zu den normalerweise definierten Möglichkeiten ist [DGS+12], [Tea23] und [Tea05] zu entnehmen.

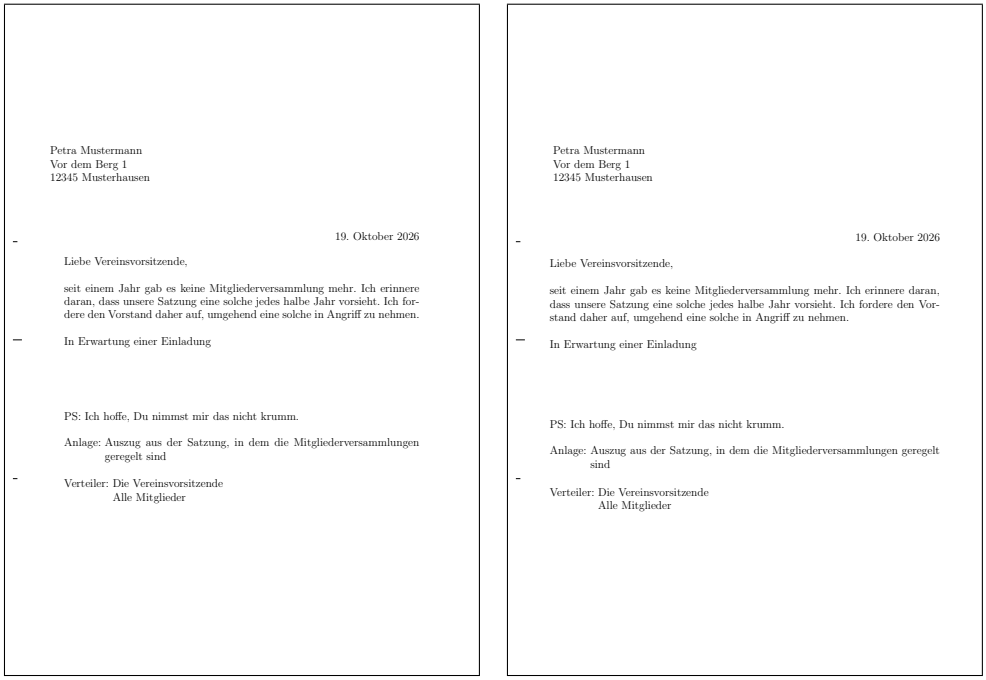



Abbildung 4.9.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen, Verteiler und ungesund großer Schrift (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe); links wurde die Schriftgröße als optionales Argument von `letter` gesetzt, rechts als optionales Argument von `\documentclass`

`sfdefaults=`*`Ein-Aus-Wert`*
`\maybesffamily`
`\textmaybesf{`*`Text`*`}`

v3.39

Die Anweisungen `\maybesffamily` und `\textmaybesf` verhalten sich je nach Einstellung von Option `sfdefaults` unterschiedlich. Als *`Ein-Aus-Wert`* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5 verwendet werden. Nur bei aktivierter Option führt `\maybesffamily` zu `\sffamily` und `\textmaybesf` zu `\textsf`. Dies ist auch die Voreinstellung. KOMA-Script selbst verwendet `\maybesffamily` in der Voreinstellung der Elemente `descriptionlabel`, `backaddress`, `refname` und `lettertitle`. `\maybesffamily` darf also auch als Teil der *`Befehle`* der nachfolgend erklärten Anweisungen `\setkomafont` und `\addtokomafont` verwendet werden.

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

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 Schriftattribute umgeschaltet werden. In der Regel werden dies Befehle wie `\rmfamily`, `\sffamily`, `\ttfamily`, `\upshape`, `\itshape`, `\slshape`, `\scshape`, `\mdseries`, `\bfseries`, `\normalfont` oder einer der Befehle `\Huge`, `\huge`, `\LARGE`, `\Large`, `\large`, `\normalsize`, `\small`, `\footnotesize`, `\scriptsize` und `\tiny` sein. Die Erklärung zu diesen Befehlen entnehmen Sie bitte [DGS+12], [Tea23] oder [Tea05]. Auch Farbumschaltungen wie `\normalcolor` sind möglich (siehe [Car17] 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 4.3 aufgelistet. Die Voreinstellungen sind den jeweiligen Abschnitten zu entnehmen.

Mit der Anweisung `\usekomafont` kann die aktuelle Schriftart auf die für das angegebene *Element* umgeschaltet werden. Ein allgemeines Beispiel für die Anwendung sowohl von `\setkomafont` als auch `\usekomafont` finden Sie in Abschnitt 3.6, Seite 61.

Tabelle 4.3.: Elemente, deren Schrift bei der Klasse scrlttr2 oder dem Paket scrletter mit `\setkomafont` und `\addtokomafont` verändert werden kann

| | |
|---|---|
| `addressee` | Name und Anschrift im Anschriftfenster (Abschnitt 4.10, Seite 215) |
| `backaddress` | Rücksendeadresse für einen Fensterbriefumschlag (Abschnitt 4.10, Seite 215) |
| `descriptionlabel` | Label, also das optionale Argument von `\item`, in einer `description`-Umgebung (Abschnitt 4.16, Seite 252) |

...

Tabelle 4.3.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`enumeratelabel`
v3.44
: Grundeinstellung für die voreingestellten Aufzählungsnummern der Umgebung `enumerate` (Abschnitt 4.16, Seite 252)

`foldmark`
: Faltmarke auf dem Briefpapier; ermöglicht Änderung der Linienfarbe (Abschnitt 4.10, Seite 198)

`footnote`
: Marke und Text einer Fußnote (Abschnitt 4.15, Seite 248)

`footnotelabel`
: Marke einer Fußnote; Anwendung erfolgt nach dem Element `footnote` (Abschnitt 4.15, Seite 248)

`footnotereference`
: Referenzierung der Fußnotenmarke im Text (Abschnitt 4.15, Seite 248)

`footnoterule`
v3.07
: Linie über dem Fußnotenapparat (Abschnitt 4.15, Seite 250)

`fromaddress`
: Absenderadresse im Briefkopf (Abschnitt 4.10, Seite 203)

`fromname`
: Name des Absenders im Briefkopf abweichend von `fromaddress` (Abschnitt 4.10, Seite 203)

`fromrule`
: Linie im Absender im Briefkopf; gedacht für Farbänderungen (Abschnitt 4.10, Seite 203)

`itemizelabel`
v3.33
: Grundeinstellung für die voreingestellten Aufzählungszeichen der Umgebung `itemize` (siehe Abschnitt 4.16, Seite 251)

`labelenumi`
v3.44
: Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumi` (siehe Abschnitt 4.16, Seite 252)

...

Tabelle 4.3.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`labelenumii`

v3.44 Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumii` (siehe Abschnitt 4.16, Seite 252)

`labelenumiii`

v3.44 Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumiii` (siehe Abschnitt 4.16, Seite 252)

`labelenumiv`

v3.44 Schriftart für die Verwendung in der Definition der Aufzählungsnummer `\labelenumiv` (siehe Abschnitt 4.16, Seite 252)

`labelinglabel`

Label, also das optionale Argument der `\item`-Anweisung, und Trennzeichen, also das optionale Argument der `labeling`-Umgebung, in einer `labeling`-Umgebung (Abschnitt 4.16, Seite 253)

`labelingseparator`

Trennzeichen, also das optionale Argument der `labeling`-Umgebung, in einer `labeling`-Umgebung; Anwendung erfolgt nach dem Element `labelinglabel` (Abschnitt 4.16, Seite 253)

`labelitemi`

v3.33 Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemi` (siehe Abschnitt 4.16, Seite 251)

`labelitemii`

v3.33 Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemii` (siehe Abschnitt 4.16, Seite 251)

`labelitemiii`

v3.33 Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemiii` (siehe Abschnitt 4.16, Seite 251)

`labelitemiv`

v3.33 Schriftart für die Verwendung in der Definition des Aufzählungszeichens `\labelitemiv` (siehe Abschnitt 4.16, Seite 251)

`pagefoot`

wird je nach Seitenstil nach `pageheadfoot` auf den Seitenfuß angewendet (Abschnitt 4.13, Seite 243)

...

Tabelle 4.3.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

`pagehead`
: wird je nach Seitenstil nach `pageheadfoot` auf den Seitenkopf angewendet (Abschnitt 4.13, Seite 243)

`pageheadfoot`
: Seitenkopf und Seitenfuß bei allen von KOMA-Script definierten Seitenstilen (Abschnitt 4.13, Seite 243)

`pagenumber`
: Seitenzahl im Kopf oder Fuß der Seite (Abschnitt 4.13, Seite 243)

`pagination`
: alternative Bezeichnung für `pagenumber`

`placeanddate`
v3.12
: Ort und Datum, falls statt einer Geschäftszeile nur eine Datumszeile verwendet wird (Abschnitt 4.10, Seite 225)

`refname`
: Bezeichnung der Felder in der Geschäftszeile (Abschnitt 4.10, Seite 224)

`refvalue`
: Werte der Felder in der Geschäftszeile (Abschnitt 4.10, Seite 224)

`specialmail`
: Versandart im Anschriftfenster (Abschnitt 4.10, Seite 215)

`lettersubject`
v3.17
: Betreff in der Brieferöffnung (Abschnitt 4.10, Seite 228)

`lettertitle`
v3.17
: Titel in der Brieferöffnung (Abschnitt 4.10, Seite 228)

`toaddress`
: Abweichung vom Element `addressee` für die Anschrift (ohne Name) des Empfängers im Anschriftfeld (Abschnitt 4.10, Seite 215)

`toname`
: Abweichung vom Element `addressee` für den Namen des Empfängers im Anschriftfeld (Abschnitt 4.10, Seite 215)

```
\usefontofkomafont{Element}
\useencodingofkomafont{Element}
\usesizeofkomafont{Element}
\usefamilyofkomafont{Element}
\useseriesofkomafont{Element}
\useshapeofkomafont{Element}
```

v3.12 Manchmal werden in der Schrifteinstellung eines Elements auch Dinge vorgenommen, die mit der Schrift eigentlich gar nichts zu tun haben, obwohl dies ausdrücklich nicht empfohlen wird. Soll dann nur die Schrifteinstellung, aber keine dieser zusätzlichen Einstellungen ausgeführt werden, so kann statt `\usekomafont` die Anweisung `\usefontofkomafont` verwendet werden. Diese Anweisung übernimmt nur die Schriftgröße und den Grundlinienabstand, die Codierung (engl. *encoding*), die Familie (engl. *family*), die Strichstärke oder Ausprägung (engl. *font series*) und die Form oder Ausrichtung (engl. *font shape*).

Mit den übrigen Anweisungen können auch einzelne Schriftattribute übernommen werden. Dabei übernimmt `\usesizeofkomafont` sowohl die Schriftgröße als auch den Grundlinienabstand.

Vor dem Missbrauch der Schrifteinstellungen wird dennoch dringend gewarnt (siehe Abschnitt 18.5, Seite 485)!

## 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 4.6 ab Seite 171) erreicht. Eine schematische Darstellung des Briefbogens und der dafür verwendeten Variablen ist in Abbildung 4.10 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.

### 4.10.1. 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.

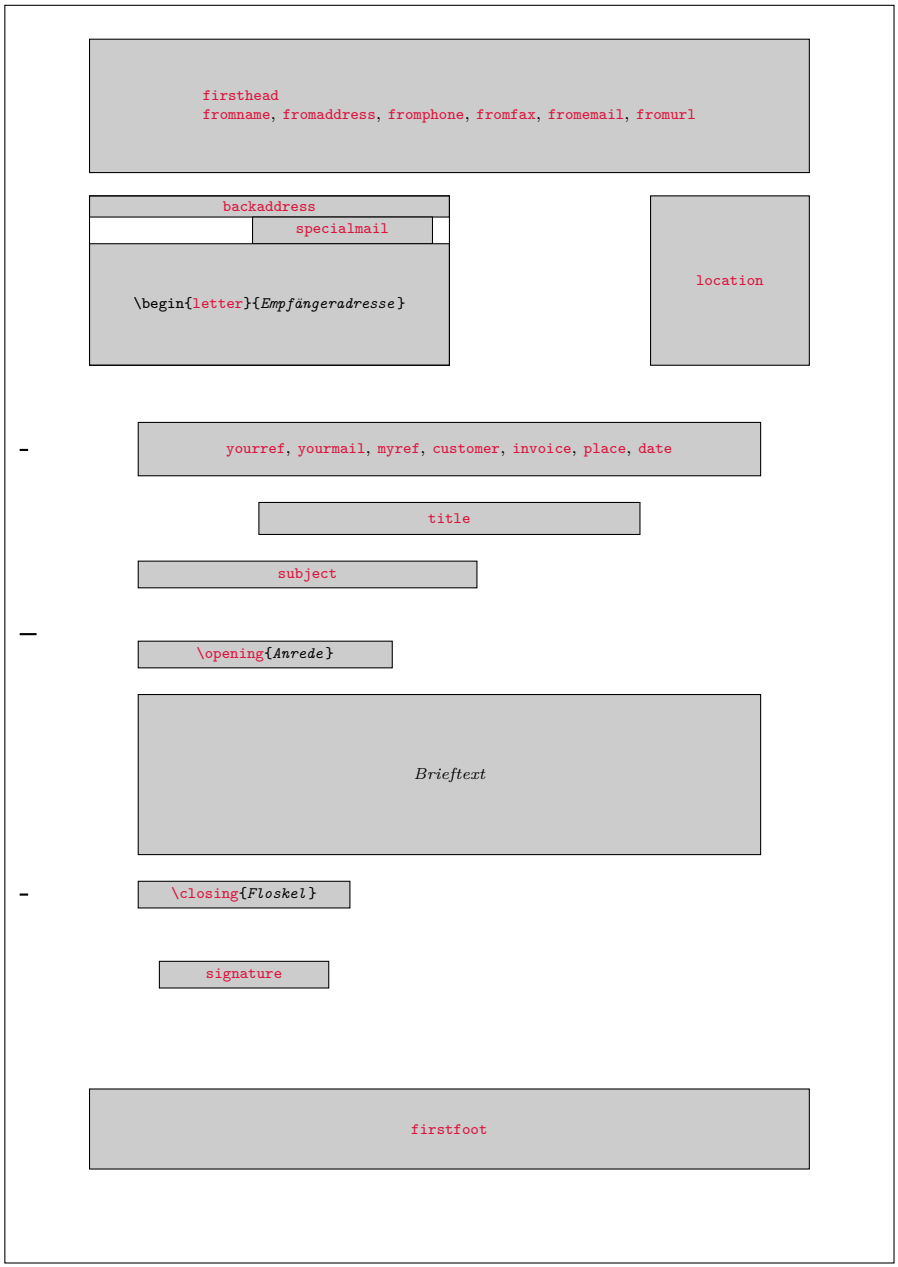



Abbildung 4.10.: Schematische Darstellung des Briefbogens mit den wichtigsten Anweisungen und Variablen für die skizzierten Elemente

Tabelle 4.4.: 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 |

`foldmarks=`*Einstellung*

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 4.6, Seite 171).

Über die Option `foldmarks` können entweder mit den Standardwerten für einfache Schalter, die in Tabelle 2.5, Seite 41 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 4.4 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 der Faltmarken ist von den Einstellungen des Anwenders beziehungsweise der `lco`-Dateien (siehe Abschnitt 4.20 ab Seite 255) abhängig. Voreingestellt sind `true` und `TBMPL`.

v2.97e

**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:

```
\KOMAoptions{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}
```

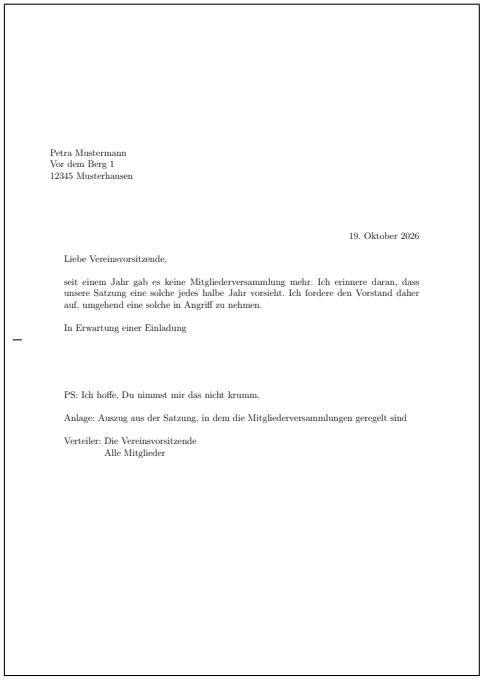



Abbildung 4.11.: Ergebnis eines kleinen Briefes mit Anschrift, Anrede, Text, Grußfloskel, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe)

```
\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 jedes 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 4.11 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 192) verwendet.

Voreingestellt ist keine Änderung.

```
\setplength{tfoldmarkvpos}{Länge}
\setplength{mfoldmarkvpos}{Länge}
\setplength{bfoldmarkvpos}{Länge}
```

KOMA-Script kennt für Briefe scrlttr2 insgesamt drei in der vertikalen Platzierung konfigurierbare Faltmarken. Die Position der oberen Faltmarke vom oberen Papierrand wird von der Pseudolänge `tfoldmarkvpos` bestimmt. Für die Position der mittleren Faltmarke ist Pseudolänge `mfoldmarkvpos`, für die untere Faltmarke `bfoldmarkvpos` zuständig. Mit der Locher- oder Seitenmittenmarke kommt noch eine weitere horizontale Marke dazu. Diese wird jedoch immer in der vertikalen Seitenmitte platziert.

v2.97e

Die obere und untere Faltmarke dienen nicht der exakten Drittelung des Papiers beim Falten. Vielmehr 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 von den `lco`-Dateien abhängig. Eine Besonderheit stellt `DINmtext` dar. Hier wird von einem Briefumschlag im Format C6/5 (auch »C6 lang« genannt) ausgegangen. Briefe, die mit dieser Option erstellt wurden, sind normalerweise nicht für Fensterbriefumschläge im Format C5 oder 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 Seite 198) explizit aktiviert wurde.

```
\setplength{tfoldmarklength}{Länge}
\setplength{mfoldmarklength}{Länge}
\setplength{bfoldmarklength}{Länge}
\setplength{pfoldmarklength}{Länge}
```

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.

\setplength{foldmarkhpos}{*Länge*}

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 Seite 198).

\setplength{lfoldmarkhpos}{*Länge*}

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. In den japanischen `lco`-Dateien (siehe Abschnitt 4.20 ab Seite 255) ist daher ein Wert von 202 mm voreingestellt. Mit der Voreinstellung Null der übrigen `lco`-Dateien wird auch dann keine Marke ausgegeben, wenn sie per Option `foldmarks` (siehe Seite 198) aktiviert wird.

\setplength{lfoldmarklength}{*Länge*}

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.

\setplength{foldmarkvpos}{*Länge*}

v2.97e Die Pseudolänge gibt den Abstand aller vertikalen Faltmarken vom oberen Papierrand an. Normalerweise sind das 3,5 mm. Ob die Faltmarken überhaupt gesetzt werden, hängt außerdem von der Option `foldmarks` ab (siehe Seite 198). Derzeit gibt es nur eine einzige vertikale Faltmarke, die als linke vertikale Faltmarke bezeichnet wird.

\setplength{foldmarkthickness}{*Länge*}

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!

### 4.10.2. 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 scrlttr2 und scrletter wird der Briefkopf jedoch unabhängig vom Seitenstil von der Anweisung `\opening` ausgegeben.

`firsthead=`*Ein-Aus-Wert*

v2.97e

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 41 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 unterzubringen. Gleichzeitig dient diese Option als zentraler Schalter, um die Erweiterungen der Briefkopfgestaltung überhaupt zu aktivieren oder zu deaktivieren. Sind die Erweiterungen deaktiviert, so bleiben diverse Optionen für den Absender ohne Wirkung. Mögliche Werte für `fromalign` sind Tabelle 4.5 zu entnehmen. Voreingestellt ist der Wert `left`.

`\setplength{firstheadvpos}{`*Länge*`}`

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.

`\setplength{firstheadhpos}{`*Länge*`}`

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` und infolge dessen horizontale Zentrierung.

`\setplength{firstheadwidth}{`*Länge*`}`

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.

Tabelle 4.5.: Mögliche Werte für Option `fromalign` zur Platzierung des Absenders auf dem Briefbogen

| | |
|---|---|
| `center`, `centered`, `middle` | Der Absender wird innerhalb des Briefkopfes zentriert; ein Logo wird gegebenenfalls am Anfang der erweiterten Absenderangabe platziert; die Erweiterungen der Briefkopfgestaltung werden aktiviert. |
| `false`, `no`, `off` | Die einfache Form des Absender wird verwendet; die Erweiterungen der Briefkopfgestaltung werden deaktiviert; die Optionen `fromrule`, `fromphone`, `frommobilephone`, `fromemail`, `fromurl` und `fromlogo` werden wirkungslos. |
| `left` | Der Absender steht linksbündig im Briefkopf; ein Logo wird gegebenenfalls rechtsbündig platziert; die Erweiterungen der Briefkopfgestaltung werden aktiviert. |
| `locationleft`, `leftlocation` | Der 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` | Der 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` | Der Absender steht rechtsbündig im Briefkopf; ein Logo wird gegebenenfalls linksbündig platziert; die Erweiterungen der Briefkopfgestaltung werden aktiviert. |

`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 4.7, Seite 209) 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 4.6. Die Länge der Linie wird über die Pseudolänge `fromrulewidth` bestimmt (siehe Seite 207).

In der Voreinstellung ist die Linie im erweiterten Briefkopf nicht aktiviert. Im einfachen Briefkopf wird die Linie immer nach dem Namen gesetzt.

Tabelle 4.6.: Mögliche Werte für Option `fromrule` zur Platzierung einer horizontalen Linie im Absender des erweiterten Briefkopfes von scrlttr2 und scrletter

| | |
|---|---|
| `afteraddress`, `below`, `on`, `true`, `yes` | Linie unterhalb des kompletten Absenders |
| `aftername` | Linie direkt unter dem Namen des Absenders |
| `false`, `no`, `off` | keine Linie |

Unter dem Namen folgt die Anschrift des Absenders. Diese wird über die Variable `fromaddress` bestimmt. Im Briefkopf wird dabei die *Bezeichnung* (siehe auch Tabelle 4.7, Seite 209) nicht gesetzt.

Die Schrift, die für den Absender verwendet wird, kann über das Element `fromaddress` eingestellt werden. Abweichungen davon können für den Absendernamen über das Element `fromname` und für die optionale Linie über das Element `fromrule` eingestellt werden. In der Voreinstellung erfolgt keinerlei Schriftumschaltung. Bei der Linie ist die Möglichkeit der Schriftumschaltung hauptsächlich dazu gedacht, die Farbe ändern zu können. Siehe hierzu [Ker07].

**Beispiel:** Geben wir nun dem Absender 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 jedes 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.
```

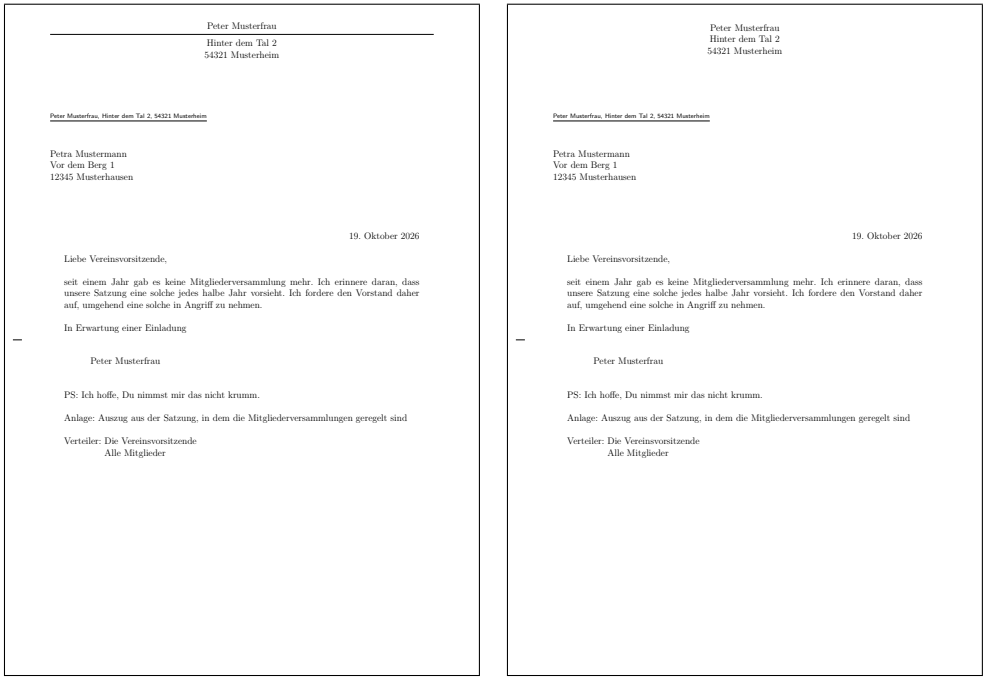



Abbildung 4.12.: Ergebnis eines kleinen Briefes mit Absender, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen und Verteiler (Datum und Faltmarken entstammen den Voreinstellungen für DIN-Briefe); links der einfache Briefkopf mit `fromalign=false`, rechts der erweiterte Briefkopf mit `fromalign=center`

```
\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 nur der einfache Briefkopf verwendet. Das Ergebnis ist in Abbildung 4.12 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 4.12 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 4.10.7 ab Seite 232 zu finden.

Nun soll der Brief mit aktivierter Erweiterung für den Briefkopf mit Hilfe der Option

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
solche jedes 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 4.13 rechts zu sehen. Links steht zum Vergleich ein Beispiel mit dem einfachen Briefkopf.

Ein wichtiger Hinweis betrifft noch die Absenderadresse: Innerhalb der Absenderadresse werden einzelne Teilangaben durch doppelten Backslash voneinander getrennt. Solche Teilangaben 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 und hätten außerdem gegebenenfalls Auswirkungen auf die Rücksendeadresse (siehe Variable backaddress, Seite 215) und Signatur (siehe Variable signature, Seite 232).

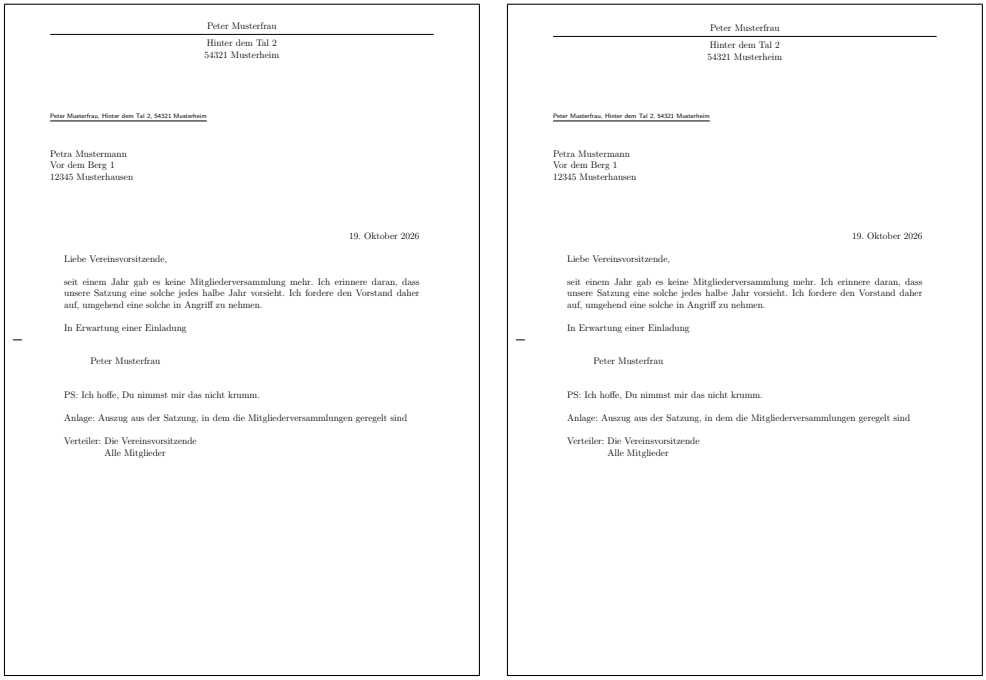



Abbildung 4.13.: Ergebnis eines kleinen Briefes mit Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links der einfache Briefkopf mit `fromalign=false`, rechts der erweiterte Briefkopf mit `fromalign=center`

`\setplength{fromrulethickness}{`*Länge*`}`
`\setplength{fromrulewidth}{`*Länge*`}`

Wie bereits bei Option `fromrule` auf Seite 203 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. Die voreingestellte Dicke, `fromrulethickness`, der Linie beträgt 0,4 pt.

v2.97c

```
symbolicnames=Wert
fromphone=Ein-Aus-Wert
frommobilephone=Ein-Aus-Wert
fromfax=Ein-Aus-Wert
fromemail=Ein-Aus-Wert
fromurl=Ein-Aus-Wert
\setkomavar{fromphone}[Bezeichnung]{Inhalt}
\setkomavar{frommobilephone}[Bezeichnung]{Inhalt}
\setkomavar{fromfax}[Bezeichnung]{Inhalt}
\setkomavar{fromemail}[Bezeichnung]{Inhalt}
\setkomavar{fromurl}[Bezeichnung]{Inhalt}
\setkomavar{phoneseparator}[Bezeichnung]{Inhalt}
\setkomavar{mobilephoneseparator}[Bezeichnung]{Inhalt}
\setkomavar{faxseparator}[Bezeichnung]{Inhalt}
\setkomavar{emailseparator}[Bezeichnung]{Inhalt}
\setkomavar{urlseparator}[Bezeichnung]{Inhalt}
```

v3.12 Mit Hilfe der fünf Optionen `fromphone`, `frommobilephone`, `fromfax`, `fromemail` und `fromurl` kann bestimmt werden, ob die Telefonnummer, die Mobiltelefonnummer, 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 41 verwendet werden. Voreingestellt ist jeweils `false`. Die Inhalte selbst werden über die gleichnamigen Variablen bestimmt. Die Voreinstellungen für die dabei verwendeten Bezeichnungen sind Tabelle 4.7 zu entnehmen, die verwendeten Trennzeichen, die zwischen der *Bezeichnung* und dem *Inhalt* einer Variablen eingefügt werden, Tabelle 4.8.

v3.12 v3.27 Mit Option `symbolicnames` kann diese Voreinstellung auf einen Schlag geändert werden. Die Option versteht die Ein-Aus-Werte für einfache Schalter, wie sie in Tabelle 2.5, Seite 41 angegeben sind. Die Aktivierung der Option entspricht dabei dem *Wert* `marvosym`, wodurch statt der sprachabhängigen Bezeichner `\emailname`, `\faxname`, `\mobilephonename` und `\phonename` Symbole aus dem marvosym-Paket verwendet werden. Gleichzeitig entfällt der Doppelpunkt bei der Definition der Trennzeichen. Für die URL entfallen in diesem Fall sowohl der sprachabhängige Bezeichner als auch das Trennzeichen. Mit `symbolicnames=fontawesome` oder `symbolicnames=awesome` werden stattdessen Symbole von Paket fontawesome verwendet. Dabei wird auch für die URL ein passendes Symbol aktiviert. Es ist zu beachten, dass das Paket marvosym oder fontawesome gegebenenfalls selbst in der Dokumentpräambel zu laden ist, falls mit der Option erst nach `\begin{document}` die Verwendung eines dieser Pakete aktiviert wird.

**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

Tabelle 4.7.: Vordefinierte Bezeichnungen der Variablen für die Absenderangaben im Briefkopf (die Bezeichnungen und Inhalte der verwendeten Variablen für Trennzeichen sind Tabelle 4.8 zu entnehmen)

| |
|---|
| fromemail<br>`\usekomavar*{emailseparator}\usekomavar{emailseparator}` |
| fromfax<br>`\usekomavar*{faxseparator}\usekomavar{faxseparator}` |
| frommobilephone<br>`\usekomavar*{mobilephoneseparator}\usekomavar{mobilephoneseparator}` |
| fromname<br>`\headfromname` |
| fromphone<br>`\usekomavar*{phoneseparator}\usekomavar{phoneseparator}` |
| fromurl<br>`\usekomavar*{urlseparator}\usekomavar{urlseparator}` |

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}
\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 jedes 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.
```

Tabelle 4.8.: Vordefinierte Bezeichnungen und Inhalte der Trennzeichen für die Absenderangaben im Briefkopf ohne Option `symbolicnumbers`

| Name | Bezeichnung | Inhalt |
|---|---|---|
| `emailseparator` | `\emailname` | `:~` |
| `faxseparator` | `\faxname` | `:~` |
| `mobilephoneseparator` | `\mobilephonename` | `\usekomavar{phoneseparator}` |
| `phoneseparator` | `\phonename` | `:~` |
| `urlseparator` | `\wwwname` | `:~` |

```
\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 4.14 links ist jedoch ernüchternd. Die Optionen werden ignoriert. Das liegt an Option `fromalign=false`. Es muss also, wie in Abbildung 4.14, rechts beispielsweise Option `fromalign=center` verwendet werden:

```
\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 jedes halbe Jahr vorsieht. Ich fordere den
Vorstand daher auf, umgehend eine solche in
Angriff zu nehmen.
\closing{In Erwartung einer Einladung}
```

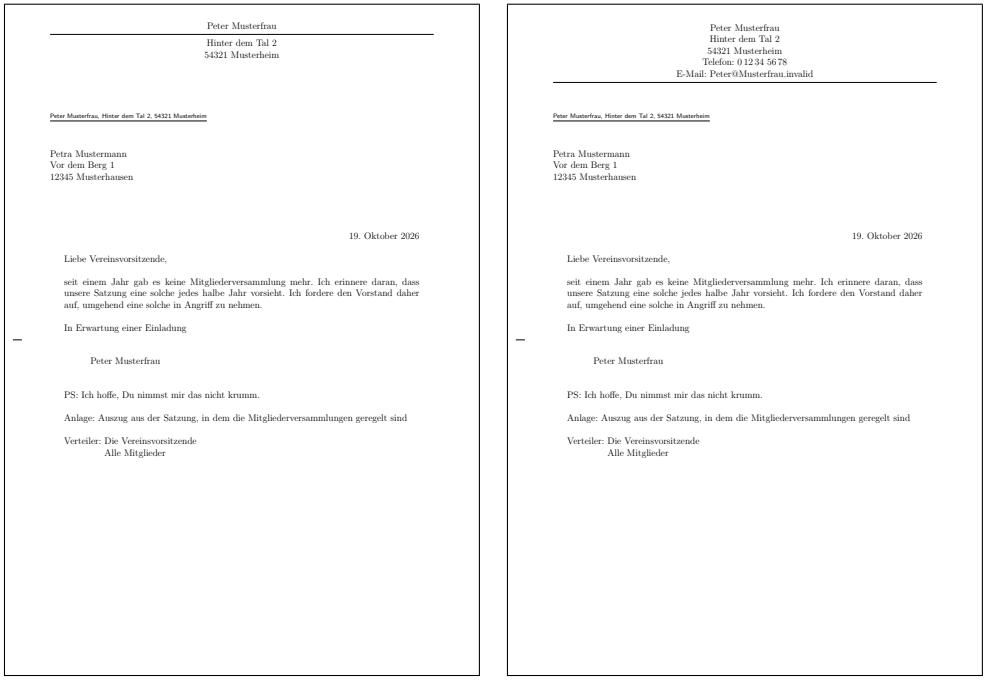



Abbildung 4.14.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links der einfache Briefkopf mit `fromalign=false`, rechts der erweiterte Briefkopf mit `fromalign=center`

```
\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 weiterer Alternativen mit linksbündigem Absender durch Einstellung `fromalign=left` und rechtsbündigem Absender durch Einstellung `fromalign=right` zeigt Abbildung 4.15.

`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 41 verwendet werden. Voreingestellt ist `false`, also kein Logo. Das Logo selbst wird über

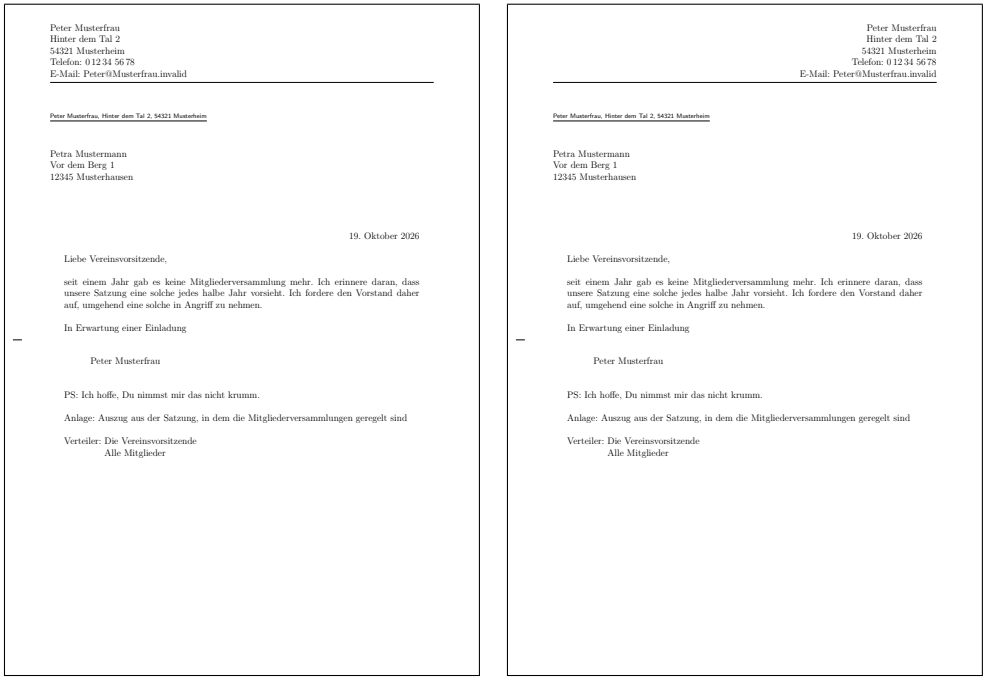



Abbildung 4.15.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links mit linksbündigem Kopf durch `fromalign=left`, rechts mit Option `fromalign=right` und damit rechtsbündigem Kopf

die Variable `fromlogo` definiert. Die *`Bezeichnung`* für das Logo ist in der Voreinstellung leer und wird von KOMA-Script auch nicht verwendet.

**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 [Car17]) ein.

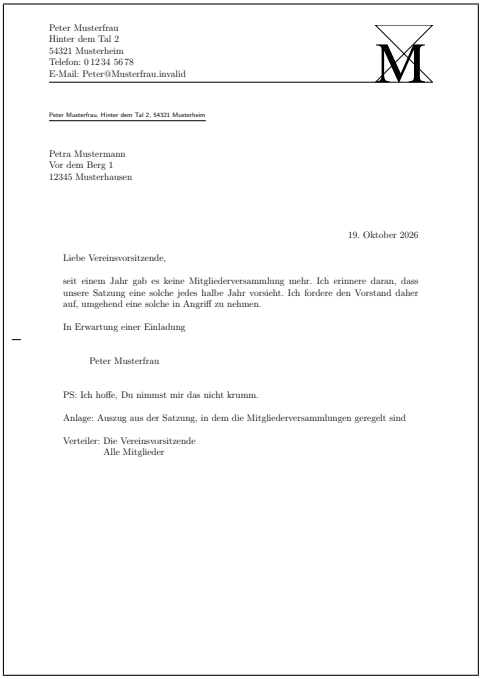
```
\documentclass[foldmarks=true,foldmarks=blmtP,
  fromrule=afteraddress,
  fromphone,fromemail,fromlogo,
  version=last]{scrlttr2}
\usepackage[ngerman]{babel}
\usepackage{graphics}
\begin{document}
\setkomavar{fromname}{Peter Musterfrau}
\setkomavar{fromaddress}{Hinter dem Tal 2\\
```

```
                              54321 Musterheim}
\setkomavar{fromphone}{0\,12\,34~56\,78}
\setkomavar{fromemail}{Peter@Musterfrau.invalid}
\setkomavar{fromlogo}{\includegraphics{musterlogo}}
\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 jedes 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 4.16 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 [Tea23]) 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 Zugriff auf andere Variablen mit Hilfe von `\usekomavar` nehmen. Die *Bezeichnung* der Variablen `firsthead` wird von KOMA-Script nicht verwendet. Ein ausführliches Beispiel für die Definition eines Briefkopfes ist beispielsweise im Anhang von [Koh20a] und [Koh20b] zu finden.







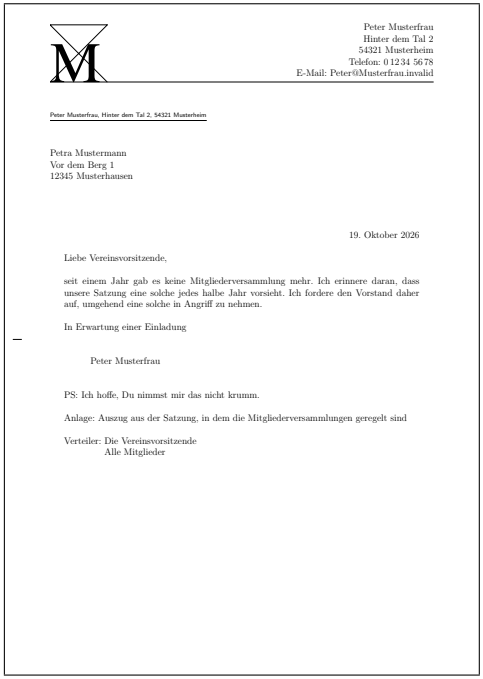



Abbildung 4.16.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Trennlinie, Anschrift, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe); links, oben mit linksbündigem Absender, rechts daneben mit zentriertem Absender und rechts mit rechtsbündigem Absender

### 4.10.3. Anschrift

Unter der Anschrift versteht man normalerweise nur den Namen und die Adresse des Empfängers. Aber auch die Versandart, beispielsweise bei Infobriefen, oder die Rücksendeadresse werden als Teil des Anschriftfeldes gesetzt.

```
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 mit `true` die Verwendung eines Anschriftfeldes. Die Option versteht die in Tabelle 4.9 angegebenen Werte für den *Modus*. Bei den Werten `true`, `topaligned`, `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 180) bestimmt. Diese Angaben werden außerdem in die Variablen `toname` und `toaddress` kopiert.

v3.03

v2.97c

Die voreingestellten Schriftarten können über die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, ab Seite 192) 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.

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 41 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

Tabelle 4.9.: Mögliche Werte für Option `addrfield` zur Auswahl der Art der Anschrift

| |
|---|
| `backgroundimage`, `PPbackgroundimage`, `PPBackgroundImage`, `PPBackGroundImage`, `ppbackgroundimage`, `ppBackgroundImage`, `ppBackGroundImage`<br>Es wird eine Anschrift mit einer in Variable `addresseeimage` abgelegten Hintergrundgrafik als Port-Payé-Kopf (P. P.-Kopf), aber ohne Rücksendeadresse und Versandart gesetzt. |
| `false`, `off`, `no`<br>Es wird keine Anschrift gesetzt. |
| `image`, `Image`, `PPimage`, `PPImage`, `ppimage`, `ppImage`<br>Eine in Variable `addresseeimage` abgelegte Abbildung wird als Anschrift mit Port-Payé gesetzt. Adressinformationen und Angaben für Rücksendeadresse, Versandart oder Priorität werden ignoriert. |
| `PP`, `pp`, `PPexplicite`, `PPExplicite`, `ppexplicite`, `ppExplicite`<br>Es wird eine Anschrift mit explizit über die Variablen `fromzipcode`, `place` und `PPcode` ausgefülltem Port-Payé-Kopf (P. P.-Kopf), gegebenenfalls mit Priorität und über Variable `PPdatamatrix` gesetzter Data-Matrix, aber ohne Rücksendeadresse und Versandart gesetzt. |
| v3.17 `topaligned`, `alignedtop`<br>Es wird eine Anschrift mit Rücksendeadresse und Versandart oder Priorität gesetzt. Die Anschrift wird dabei unter der Versandart nicht vertikal zentriert. |
| `true`, `on`, `yes`<br>Es wird eine Anschrift mit Rücksendeadresse und Versandart oder Priorität gesetzt. |

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` ersetzt 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
v3.39 `backaddress` konfigurierbar. Voreingestellt ist hierbei `\maybesffamily` (siehe Tabelle 4.10). Vor der Anwendung der konfigurierten Schriftumschaltung wird noch `\scriptsize` ausgeführt.

v3.17 Während die Adresse in der Voreinstellung `addrfield=true` im für die Anschrift verfügbaren Platz vertikal zentriert wird, entfällt die Zentrierung mit `addrfield=topaligned`. Sie wird dann oben bündig im verfügbaren Platz gesetzt.

Zwischen Rücksendeadresse und Empfängeradresse kann bei der Standardeinstellung `addrfield=true` noch eine optionale Versandart gesetzt werden. Diese wird genau dann

Tabelle 4.10.: Voreinstellungen für die Schrift der Elemente des Anschriftfensters

| Element | Voreinstellung |
|---|---|
| addressee | |
| backaddress | \maybesffamily |
| PPdata | \sffamily |
| PPlogo | \sffamily\bfseries |
| priority | \fontsize{10pt}{10pt}\sffamily\bfseries |
| prioritykey | \fontsize{24.88pt}{24.88pt}\selectfont |
| specialmail | |
| toaddress | |
| toname | |

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 220) festgelegt. Die voreingestellte Schriftart des v2.97c Elements `specialmail`, die Sie Tabelle 4.10 entnehmen können, kann mit Hilfe der Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 4.9, Seite 192) verändert werden.

v3.03 Wird hingegen mit `priority=A` oder `priority=B` (siehe Tabelle 4.11) 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 4.10 entnehmen können, lassen sich mit den Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 4.9, Seite 192) ändern.

v3.03 Bei `addrfield=PP` wird im Port-Payé-Kopf die Postleitzahl aus der Variablen `fromzipcode`

Tabelle 4.11.: Mögliche Werte für Option `priority` zur Auswahl einer internationalen Priorität im Adressfeld

`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 anstelle 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 anstelle der Versandart.

und der Ort aus der Variablen `place` gesetzt. Dabei wird der Postleitzahl, also dem *Inhalt* von `fromzipcode`, die *Bezeichnung* der Variablen `fromzipcode`, gefolgt vom *Inhalt* von `zipcodeseparator` vorangestellt. Für diese *Bezeichnung* hängt die Voreinstellung von der verwendeten `lco`-Datei (siehe Abschnitt 4.20 ab Seite 255) ab. Für den *Inhalt* von `zipcodeseparator` ist hingegen »`\,--\,`« 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 Tabelle 4.10 zu entnehmen und kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) verändert werden.

Für den Port-Payé-Schriftzug »P.P.« kommt dagegen die Schrift des Elements `PPlogo` zur Anwendung. Dessen Voreinstellung ist ebenfalls Tabelle Tabelle 4.10 zu entnehmen.

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 219) sowie `PPheadwidth` und `PPheadheight` (siehe Seite 220) bestimmt. Für die Anordnung der Data-Matrix ist die Pseudolänge `PPdatamatrixvskip` (siehe Seite 220) 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.

`\setplength{toaddrvpos}{`*Länge*`}`
`\setplength{toaddrhpos}{`*Länge*`}`

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 Seite 215).

\setplength{toaddrheight}{*Länge*}

Diese Pseudolänge gibt die Höhe des Anschriftfeldes einschließlich der Versandart an. Ob Name und Adresse des Empfängers unter Berücksichtigung der Versandart im Anschriftfeld vertikal zentriert werden, hängt von Option addrfield ab.

\setplength{toaddrwidth}{*Länge*}

Diese Pseudolänge gibt die Breite des Anschriftfensters an. Sie 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 SNmmarg anstelle von SN.

\setplength{toaddrindent}{*Länge*}

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 Seite 215) 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. Des Weiteren wird toaddrindent bei den genannten Einstellungen für addrfield auch für den Abstand zum rechten Rand des Anschriftfensters verwendet.

\setplength{backaddrheight}{*Länge*}

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 zur Rücksendung 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 Seite 215) und `backaddress` (siehe Seite 215). Ein Abschalten der Rücksendeadresse ist beispielsweise sinnvoll, wenn gar keine Fensterbriefumschläge verwendet werden.

\setplength{specialmailindent}{*Länge*}
\setplength{specialmailrightindent}{*Länge*}

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.

\setplength{PPheadheight}{*Länge*}
\setplength{PPheadwidth}{*Länge*}

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 Seite 215) 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 KOMA-Script selbstständig in 20,74 pt. Den normalerweise voreingestellten Wert von 0 mm für `PPheadwidth` ändert KOMA-Script selbstständig in 42 mm.

\setplength{PPdatamatrixvskip}{*Länge*}

v3.03 Durch diese Pseudolänge wird der vertikale Abstand zwischen dem Port-Payé-Kopf und der Data-Matrix bei `addrfield=PP` (siehe Seite 215) festgelegt. Den normalerweise voreingestellten Wert von 0 mm ändert KOMA-Script selbstständig in 9 mm. Die Data-Matrix wird rechtsbündig zum Port-Payé-Kopf gesetzt.

Tabelle 4.12.: Mögliche Werte für Option `locfield` zur Wahl der Breite des Feldes für die Absenderergänzung (Erklärung im Text beachten!)

| | |
|---|---|
| `narrow` | schmales Feld für die Absenderergänzung |
| `wide` | breites Feld für die Absenderergänzung |

### 4.10.4. Absenderergänzung

Der freie Platz neben der Anschrift kann für zusätzliche Angaben zum Absender genutzt werden.

`locfield=`*`Einstellung`*

Der Inhalt der Absenderergänzung neben der Anschrift ist frei wählbar. Je nach Einstellung der oben erklärten Option `fromalign` wird sie außerdem für das Logo des Absenders oder den Absender selbst mitverwendet. Die Breite dieses Feldes kann beispielsweise in einer `lco`-Datei (siehe Abschnitt 4.20) 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. Dies ist bei der Mehrzahl der `lco`-Dateien der Fall, die KOMA-Script beiliegen. Siehe hierzu auch die Erklärungen zur Pseudolänge `locwidth`, Seite 222. Mögliche Werte für die Option sind Tabelle 4.12 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 jedes 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 4.17 gesetzt.

```
\setplength{locheight}{Länge}
\setplength{lochpos}{Länge}
\setplength{locvpos}{Länge}
\setplength{locwidth}{Länge}
```

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, oberen Papierecke an. Die 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` anhand der Breite des Papiers, der Breite des Anschriftfensters, des Abstandes des Anschriftfensters von der linken, oberen Papierecke und Option `locfield` (siehe Seite 221) gesetzt werden. Wie 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 Seite 218).

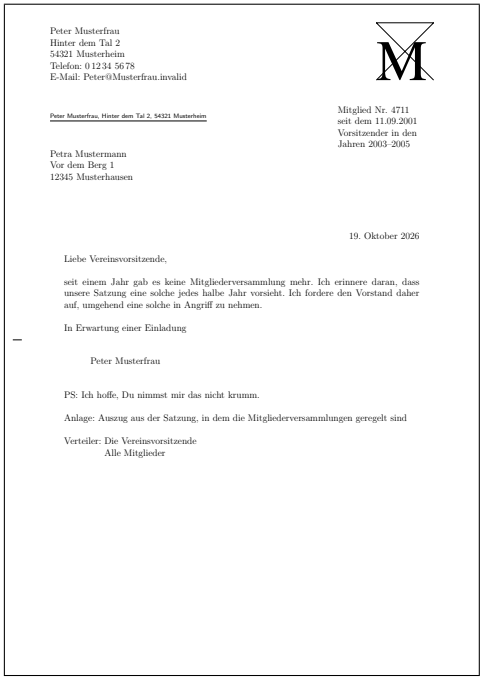



Abbildung 4.17.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke (das Datum entstammt den Voreinstellungen für DIN-Briefe)

### 4.10.5. Geschäftszeilen

Die Geschäftszeilen enthalten typischerweise Angaben wie Namenskürzel oder Durchwahlnummern. KOMA-Script setzt automatisch nur nicht leere Felder. Ist nur das Datum nicht leer, wird die Geschäftzeile zur Datumszeile.

`numericaldate=`*`Ein-Aus-Wert`*

Mit dieser Option kann zwischen der sprachabhängigen Standarddarstellung des Datums in `\today` und einem ebenfalls sprachabhängigen rein numerischen Datum umgeschaltet werden. Die Standarddarstellung wird nicht von KOMA-Script bereitgestellt. Sie kann wahlweise vom LaTeX-Kern oder einem Paket wie babel oder isodate stammen. Das kurze numerische Datum wird hingegen von scrlttr2 selbst erzeugt. Die Option versteht die Standardwerte für einfache Schalter, die in Tabelle 2.5, Seite 41 angegeben sind. Voreingestellt ist mit `false` die Verwendung der Standarddarstellung.

`\setkomavar{date}[`*`Bezeichnung`*`]{`*`Inhalt`*`}`

Das Datum wird im *`Inhalt`* der Variablen `date` abgelegt. Voreingestellt ist das mit `\date` gesetzte Datum, das selbst mit `\today` voreingestellt ist. Damit ist der *`Inhalt`* der Variablen nur indirekt von Option `numericaldate` abhängig.

Tabelle 4.13.: Mögliche Werte für Option `refline` zur Konfiguration der Geschäftszeile

| | |
|---|---|
| `dateleft` | |
| v3.09 | Das Datum steht automatisch links in der Geschäftszeile. |
| `dateright` | |
| v3.09 | Das Datum steht automatisch rechts in der Geschäftszeile. |
| `narrow` | Die Breite der Geschäftszeile richtet sich nach dem Satzspiegel. |
| `nodate` | |
| v3.09 | Das Datum wird nicht automatisch in die Geschäftszeile gesetzt. |
| `wide` | Die Breite der Geschäftszeile richtet sich nach Anschrift und Absenderergänzung. |

Gesetzt wird das Datum normalerweise als Teil der Geschäftszeile. Wenn die Geschäftszeile ansonsten leer bleibt, wird allerdings nur eine Datumszeile, bestehend aus dem Ort und dem Datum, gesetzt. Trotzdem haben auch in diesem Fall die Einstellungen der nachfolgend beschriebenen Option `refline` Auswirkungen auf diese Datumszeile. Weitere Informationen zum Ort finden Sie in der Beschreibung zur Variablen `placeseparator` auf Seite 225.

`refline=`*`Einstellung`*

Bei scrlttr2 und scrletter 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. Mögliche Werte für diese Option sind
v3.09 Tabelle 4.13 zu entnehmen. Voreingestellt sind `narrow` und `dateright`.

```
\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. Ihre Bedeutung ist Tabelle 4.1, Seite 166 zu entnehmen. Jede dieser Variablen hat auch eine vordefinierte *`Bezeichnung`*, die in Tabelle 4.14 zu finden ist. Wie weitere Variablen zur Geschäftszeile hinzugefügt werden können, ist in Abschnitt 19.1 ab Seite 513 erklärt.

v2.97c Schriftart und Farbe der Feldbezeichnung und des Feldinhalts können über die beiden Elemente `refname` und `refvalue` geändert werden. Dazu werden die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) verwendet. Die Voreinstellungen der beiden Elemente sind Tabelle 4.15 zu entnehmen.

Tabelle 4.14.: 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 |

\setkomavar{placeseparator}[*Bezeichnung*]{*Inhalt*}

Sind bis auf date alle Variablen der Geschäftszeile leer, so wird keine echte Geschäftszeile gesetzt. Stattdessen werden dann nur 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.

v3.12 Für eine solche Datumszeile mit Ort findet nicht die Schrifteinstellung des Elements refvalue Anwendung. Stattdessen wird das Element placeanddate verwendet, dessen leere Voreinstellung mit Hilfe der Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 4.9, Seite 192) geändert werden kann.

Die in Tabelle 4.13 angegebenen Datumseinstellungen für refline werden auch von der Datumszeile beachtet. Voreingestellt ist rechtsbündig.

**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}
```

Tabelle 4.15.: Voreinstellungen für die Schrift der Elemente der Geschäftszeile

| Element | Voreinstellung |
|---|---|
| refname<br>refvalue | \maybesffamily\scriptsize |

v3.39

```
\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 jedes 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}
```

In Abbildung 4.18 erscheint damit der Ort, gefolgt von den automatischen Trennzeichen vor dem Datum. Dieses Datum wurde im Beispielcode über Variable `date` explizit gesetzt, damit bei einem späteren LaTeX-Durchlauf des Briefes das ursprüngliche Datum erhalten bleibt und nicht automatisch das Datum des LaTeX-Laufs verwendet wird.

`\setplength{refvpos}{`*Länge*`}`

Diese Pseudolänge gibt den Abstand der Geschäftszeile von der Oberkante des Papiers an. Ihr Wert wird in den vordefinierten `lco`-Dateien unterschiedlich eingestellt.

`\setplength{refwidth}{`*Länge*`}`
`\setplength{refhpos}{`*Länge*`}`

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. Damit wird festgelegt, dass die verfügbare Breite erst innerhalb von `\opening` ermittelt wird. Diese Breite richtet sich dann nach der Einstellung der Option `refline` (siehe Seite 224). Gleichzeitig wird `refhpos` entsprechend der Option gesetzt.

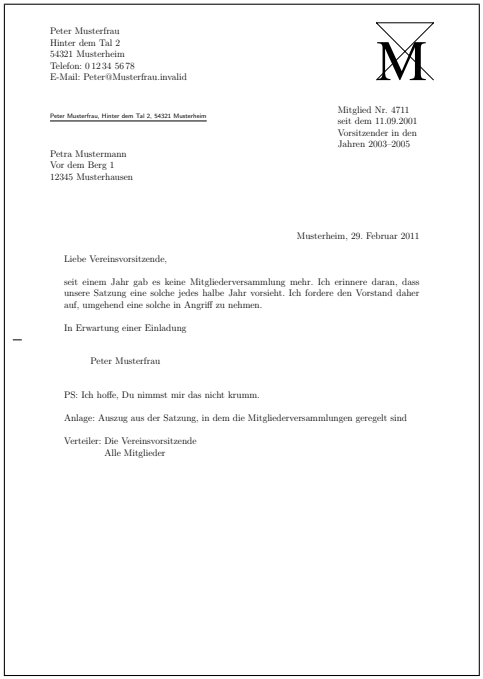



Abbildung 4.18.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke

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 nicht von der Option `refline` bestimmt, sondern **`refhpos`** ist der Abstand der Geschäftszeile von der linken Papierkante. 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 den Abständen des Satzspiegels von der linken und rechten Papierkante entspricht. Bei auf dem Papier horizontal zentriertem Satzspiegel wird also auch die Geschäftszeile zentriert.

In der Regel werden diese Sonderfälle für die meisten Anwender 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.

`\setplength{refaftervskip}{`*Länge*`}`

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 Textbereichs der ersten Seite aus.

### 4.10.6. Titel und 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.

`\setkomavar{title}[`*Bezeichnung*`]{`*Inhalt*`}`

Bei KOMA-Script kann ein Brief zusätzlich mit einem Titel versehen werden. Der Titel wird zentriert in der Schriftgröße `\LARGE` unterhalb der Geschäftszeile ausgegeben.

v3.17 Die Schriftart für das Element `lettertitle` kann mit den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9 ab Seite 192) geändert werden. Dabei sind auch Größenangaben erlaubt. Die Größe `\LARGE` wird der Schriftauswahl innerhalb von KOMA-Script immer vorangestellt und ist daher auch nicht Teil der Voreinstellung `\normalcolor \maybesffamily\bfseries`. Bei scrlttr2 kann als Alias für `lettertitle` auch `title` verwendet werden. Bei Verwendung von scrletter mit einer KOMA-Script-Klasse ist das nicht möglich, da bei diesen Klassen bereits ein Element `title` mit abweichender Einstellung für den Dokumenttitel existiert.

v3.39 scrlttr2

**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 setzen, legt man den *Inhalt* der Variablen `subject` entsprechend fest. Mit Option `subject=titled` 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 4.16). 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 `subject=beforeopening` erst nach der Anrede gesetzt werden soll. Eine andere Formatierung kann mit `subject=underlined` und `subject=centered` oder `subject=right` eingestellt werden.

v2.97c

Mögliche Werte für Option `subject` sind Tabelle 4.17 zu entnehmen. Es wird darauf hingewiesen, dass bei Einstellung `subject=underlined` der Betreff in eine Zeile passen muss!

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 192). Für

Tabelle 4.16.: Vordefinierte Bezeichnungen der Variablen für den Betreff

| Name | Bezeichnung |
|---|---|
| subject | \usekomavar*{subjectseparator}% \usekomavar{subjectseparator} |
| subjectseparator | \subjectname |

v3.17 scrlttr2

das Element lettersubject ist als Schrift \normalcolor\bfseries voreingestellt. Bei Klasse scrlttr2 kann als Alias für lettersubject auch subject verwendet werden. Bei Verwendung von Paket scrletter mit einer KOMA-Script-Klasse ist das nicht möglich, da bei diesen Klassen bereits ein Element subject mit abweichender Einstellung für den Dokumenttitel existiert.

**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 zugehörige 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 jedes halbe Jahr vorsieht. Ich fordere den
```

Tabelle 4.17.: Mögliche Werte für Option `subject` zur Platzierung und Formatierung eines Betreffs

| | |
|---|---|
| `afteropening` | Betreff nach der Anrede setzen. |
| `beforeopening` | Betreff vor der Anrede setzen (Voreinstellung). |
| `centered` | Betreff zentrieren. |
| `left` | Betreff linksbündig setzen (Voreinstellung). |
| `right` | Betreff rechtsbündig setzen. |
| `titled` | Betreff mit Titel versehen. |
| `underlined` | Betreff unterstreichen. (Hinweis im Text beachten!) |
| `untitled` | Betreff nicht mit Titel versehen (Voreinstellung). |

```
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.

Voreingestellt sind `subject=beforeopening`, `subject=left` sowie `subject=untitled`.

`\setplength{subjectvpos}{`*Länge*`}`

v3.01 Ist der Wert dieser Pseudolänge 0 pt, so bestimmt die Option `subject` (siehe Seite 228) 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




Abbildung 4.19.: Ergebnis eines kleinen Briefes mit erweitertem Absender, Logo, Anschrift, Absenderergänzung, Ort, Datum, Betreff, Anrede, Text, Grußfloskel, Signatur, Postskriptum, Anlagen, Verteiler und Lochermarke

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. Das sollte in den allermeisten Fällen genügen.

```
\setplength{subjectbeforevskip}{Länge}
\setplength{subjectaftervskip}{Länge}
```

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 gegebenenfalls 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 von Klasse und Paket zwei Zeilen.

### 4.10.7. Schlussgruß

Dass der Schlussgruß mit \closing gesetzt wird, wurde bereits in Abschnitt 4.7, Seite 183 erklärt. Unter Schlussgruß und Unterschrift wird häufig noch eine Signatur als eine Art Erläuterung oder Klartext der Unterschrift gesetzt.

\setkomavar{signature}[*Bezeichnung*]{*Inhalt*}
\sigspace

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.

v3.44 Bei Briefen mit mehreren Unterzeichners können die einzelnen Signaturen innerhalb der Variablen signature mit \and voneinander getrennt werden. Sie werden dann blockweise nebeneinander gesetzt. Zwischen den einzelnen Signaturen wird jeweils \sigspace eingefügt. Dieses ist mit \quad also einem horizontalen Abstand von einem Geviert voreingestellt.

\raggedsignature
\sigalign

Grußfloskel und Erläuterung der Unterschrift werden innerhalb einer gemeinsamen 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 Seite 234) bestimmt. Durch den Befehl \raggedsignature wird die Ausrichtung innerhalb der Box bestimmt. In den vordefinierten lco-Dateien ist die Anweisung entweder auf \centering oder auf \raggedright (nur KOMAold) gesetzt. Um innerhalb der Box eine linksbündige, zentrierte oder rechtsbündige Ausrichtung zu erhalten, kann der Befehl mit \renewcommand zu \raggedleft, \centering oder \raggedright umdefiniert werden.

v3.44 Erfahrenen Anwender ist auch die Verwendung von \let gestattet. Andere Definitionen sind nur zulässig, wenn gleichzeitig \sigalign definiert wird. In diesem Fall muss \sigalign zu einer gültigen Spaltendefinition expandieren, beispielsweise l, c, r oder auch p{.5\textwidth}. \sigalign ist maßgeblich für die Ausrichtung innerhalb einer Signatur, falls diese mehrzeilig ist.

**Beispiel:** Herr Musterfrau will sich nun wirklich wichtig machen und deshalb in der Signatur nochmals darauf hinweisen, dass er selbst schon Vereinsvorsitzender war. Deshalb

ändert er die Variable signature. Außerdem will er, dass die Signatur linksbündig unter dem Schlussgruß steht, und definiert dazu \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 jedes 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}
```


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 jedes 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

Das Ergebnis ist in Abbildung 4.20 zu sehen.

```
\setplength{sigindent}{Länge}
\setplength{sigbeforevskip}{Länge}
```

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` festgelegten 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.

### 4.10.8. Briefbogenfuß

Die erste Seite eines Briefes, der Briefbogen, enthält nicht nur einen eigenen Kopf, den Briefkopf. Diese Seite enthält auch einen eigenen Fuß, den Briefbogenfuß. Dieser wird ebenfalls nicht über den Seitenstil, sondern unmittelbar von `\opening` ausgegeben.

`shrinkfirstpage=`*Ein-Aus-Wert*
`enlargefirstpage=`*Ein-Aus-Wert*

Die erste Seite eines Briefes fällt aufgrund der vielen Konsultationselemente, wie dem Briefkopf oder der Anschrift, immer aus dem normalen Satzspiegel. Von KOMA-Script 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.

v3.48 Soll der Textbereich nicht automatisch verkleiner werden, um Kollisionen von Text und Fuß zu verhindern, so kann dies mit `shrinkfirstpage=false` erreicht werden. Soll der Textbereich auch automatisch mit `\enlargethispage` vergrößert werden, falls der Fuß der ersten Seite dies erlaubt, so kann das mit `enlargefirstpage=true` erreicht werden. Es passt dann bestenfalls etwas mehr Text auf die erste Seite. Siehe hierzu auch die Erklärung zur Pseudolänge `firstfootvpos` auf Seite 237. Als *Ein-Aus-Wert* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 41 verwendet werden.

v3.48 Seit Version 3.48 ist KOMA-Script außerdem etwas tolerant bei der Kollisionserkennung. Sollte die Verkleinerung oder die Vergrößerung geringer als `\maxdepth` ausfallen, so unterbleibt sie automatisch. `\maxdepth` wird vom Paket typearea normalerweise auf die Hälfte von `\topskip` eingestellt. `\topskip` wiederum entspricht in der Regel der Grundschriftgröße und ist damit von Option `fontsize` abhängig. Bei scrlttr2 gilt dies nur, wenn nicht per Option `version` Kompatibilität zu einer früheren Version erzwungen wird.

`firstfoot=`*Ein-Aus-Wert*

v2.97e Diese Option bestimmt, ob der Briefbogenfuß überhaupt gesetzt wird. Das Abschalten mit `firstfoot=false` hat Auswirkungen wenn gleichzeitig die zuvor dokumentierte Option `enlargefirstpage` 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 41 angegeben sind. Voreingestellt ist das Setzen des Briefbogenfußes.

`\setkomavar{firstfoot}[`*Bezeichnung*`]{`*Inhalt*`}`

v3.08 Der Inhalt der Variablen `firstfoot` und damit des Fußes der ersten Seite ist in der Voreinstellung leer. 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:

```
\setkomavar{firstfoot}{%
  \parbox[b]{\linewidth}{%
```

```
    \centering\def\\{, }\usekomavar{frombank}%
  }%
}
```

Natürlich können Sie für das Trennzeichen auch eine eigene Variable definieren.

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@{}}{Geschäftsführung:}\\
      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. Es wird empfohlen, eine solche Definition für die mehrfache Verwendung in unterschiedlichen Dokumenten in einer eigenen `lco`-Datei abzulegen. Mit

```
\setkomavar{frombank}{IBAN DE21~87654321~13456789\\
                      bei der HansWurstBank\\
                      BIC GRMLDEHD000}
```

kann die Bankverbindung dann im Dokument passend dazu gesetzt werden.

Im Beispiel wurde ein mehrzeiliger Fuß gesetzt. Bei einer Kompatibilitätseinstellung ab Version 2.9u (siehe `version` in Abschnitt 4.2, Seite 164) reicht der Platz dafür in der Regel nicht aus. Sie sollten dann `firstfootvpos` (siehe Seite 237) entsprechend verringern.

\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 221) oder wie im Beispiel in den Fuß zu setzen.

\setplength{firstfootvpos}{*Länge*}

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 Textbereichs mit Hilfe von `\enlargethispage` verkleinert. Mit Hilfe der Option `enlargefirstpage` (siehe Seite 235) kann dafür gesorgt werden, dass die Höhe des Textbereichs 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 4.2, Seite 164) wird außer bei `KOMAold` und `NF` in allen vordefinierten `lco`-Dateien (siehe Abschnitt 4.20) der Fuß abhängig vom Satzspiegel gesetzt und `enlargefirstpage` ignoriert. 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 Wird der Briefbogenfuß mit Option `firstfoot=false` (siehe Seite 235) abgeschaltet, so wird `firstfootvpos` ignoriert und stattdessen `\paperheight` angenommen. Es bleibt dann ein minimaler unterer Rand von `\footskip`.

\setplength{firstfoothpos}{*Länge*}

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` und infolge dessen horizontale Zentrierung.

\setplength{firstfootwidth}{*Länge*}

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.

## 4.11. Absatzauszeichnung

In der Einleitung zu Abschnitt 3.10 ab Seite 80 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 für Briefe das einheitliche Erscheinungsbild aller Dokumente aus einer Quelle, die so genannte *Corporate Identity*.

Über diese Überlegungen hinaus gilt Abschnitt 3.10 sinngemäß. Falls Sie also Abschnitt 3.10 bereits gelesen und verstanden haben, können Sie nach dem Ende dieses Abschnitts auf Seite 239 mit Abschnitt 4.12 fortfahren. Dies gilt ebenso, wenn Sie nicht mit Klasse scrlttr2, sondern mit Paket scrletter arbeiten. Das Paket bietet keine eigenen Einstellungen für die Absatzauszeichnung, sondern verlässt sich dabei ganz auf die verwendete Klasse.

scrlttr2

`parskip=`*`Methode`*

Bei Briefen findet man häufiger ein Layout mit Absatzabstand anstelle des voreingestellten Absatzeinzugs. Die KOMA-Script-Klasse scrlttr2 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, freigelassen. Mit dem Pluszeichen wird am Zeilenende mindestens ein Drittel und mit dem Stern mindestens ein Viertel einer normalen Zeile freigelassen. Mit der Minus-Variante werden keine Vorkehrungen für die letzte Zeile eines Absatzes getroffen.

scrlttr2

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 am Ende des Absatzes sichtbar.

v3.08

Neben den sich so ergebenden acht Kombinationen ist es noch möglich, als *`Methode`* die Werte für einfache Schalter aus Tabelle 2.5, Seite 41 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 auf Seite 81.

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.

Voreingestellt ist bei KOMA-Script `parskip=false`. Hierbei gibt es keinen Absatzabstand, sondern einen Absatzeinzug von 1 em.

## 4.12. Erkennung von rechten und linken Seiten

Es gilt sinngemäß, was in Abschnitt 3.11 geschrieben wurde. Falls Sie also Abschnitt 3.11 bereits gelesen und verstanden haben, können Sie in Abschnitt 4.13 auf Seite 240 fortfahren.

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. In der Regel werden Briefe einseitig gesetzt. Sollen Briefe mit einseitigem Layout jedoch auf Vorder- und Rückseite gedruckt oder ausnahmsweise tatsächlich doppelseitige Briefe erstellt werden, kann unter Umständen das Wissen, ob man sich auf einer Vorder- oder einer Rückseite befindet, nützlich sein.

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

v3.28

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 aktuell 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 ungerader 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 18.1, Seite 481 zu finden.

## 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.

`\letterpagestyle`

v3.19 scrlttr2

Der für Briefe voreingestellte Seitenstil wird durch den Inhalt dieser Anweisung bestimmt. In der Voreinstellung von scrlttr2 ist die Anweisung leer definiert. Das bedeutet, dass der Seitenstil von Briefen dem des restlichen Dokuments entspricht. Dies ist deshalb sinnvoll, weil scrlttr2 für reine Briefdokumente gedacht ist und es dafür einfacher ist, den Seitenstil wie gewohnt mit `\pagestyle` global einzustellen.

scrletter

Da sowohl Seitenstil `plain` als auch der Stil `headings` anderer Klassen vom gewünschten Seitenstil für Briefe abweicht, ist für das Paket scrletter der Seitenstil `plain.letter` in der Anweisung `\letterpagestyle` gespeichert. Damit werden alle Briefe mit dem zum Seitenstil `letter` gehörenden `plain`-Seitenstil gesetzt unabhängig davon, was für das restliche Dokument als Seitenstil eingestellt ist.

**Beispiel:** Sie wollen auch bei Verwendung von Paket scrletter, dass die Briefe in dem Seitenstil gesetzt werden, der für das Dokument selbst mit `\pagestyle` eingestellt wurde. Dazu schreiben sie die Anweisung

```
\renewcommand*{\letterpagestyle}{}
```

in die Dokumentpräambel. Dabei ist übrigens der Stern bei `\renewcommand*` wichtig!

Natürlich haben die Anweisung `\pagestyle` und `\thispagestyle` innerhalb eines Briefes Vorrang vor dem innerhalb von `\begin{letter}` über `\letterpagestyle` eingestellten Seitenstil.

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

scrlttr2

Mit diesen Optionen kann bei scrlttr2 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 41 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.

Beim Seitenstil empty (siehe Seite 241) haben die beiden Optionen headsepline und footsepline selbstverständlich keine Auswirkung. Bei diesem Seitenstil soll ja auf Seitenkopf und Seitenfuß 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 aber nicht, dass Kopf oder Fuß räumlich weiter vom Textkörper weggerückt werden müssten. Stattdessen sollten sie bei der Berechnung des Satzspiegels als zum Textkörper gehörend betrachtet werden. Dies wird bei scrlttr2 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.

Die Optionen führen selbst keine automatische Neuberechnung des Satzspiegels aus. Zur Neuberechnung des Satzspiegels siehe Option DIV mit den Werten last oder current (Seite 37) oder Anweisung \recalctypearea (Seite 40) in Kapitel 2.

Das Paket scrlayer-scrpage (siehe Kapitel 5) bietet weitere Einflussmöglichkeiten für Linien im Kopf und Fuß und kann auch mit scrlttr2 kombiniert werden. Das Paket scrletter verwendet hingegen automatisch scrlayer-scrpage zur Definition der Seitenstile letter und
scrletter plain.letter. Die von scrletter definierten Seitenstile unterliegen damit den Regeln jenes Pakets. Dies betrifft insbesondere das Setzen der Linien in Kopf und Fuß des plain-Seitenstils plain.letter. Siehe dazu in Abschnitt 5.5, Seite 293 und Seite 295 die Optionen headsepline und plainheadsepline. Auch Einstellungen wie automark sind für den Seitenstil letter von einiger Bedeutung.

pagenumber=*Position*

Mit Hilfe dieser Option kann bestimmt werden, ob und wo eine Seitenzahl auf Folgeseiten ge-
scrlttr2 setzt werden soll. Die Option wirkt sich bei scrlttr2 auf die Seitenstile headings, myheadings
scrletter und plain und bei scrletter auf letter und plain.letter aus. Sie beeinflusst außerdem die Voreinstellung der Seitenstile des Pakets scrlayer-scrpage, soweit sie vor dem Laden des Pakets 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 4.18 zu entnehmen. Voreingestellt ist botcenter.

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

scrlttr2 Bei Briefen mit scrlttr2 wird zwischen vier verschiedenen Seitenstilen unterschieden. Dagegen
scrletter definiert scrletter nur zwei eigene Seitenstile.

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 182) mit anderen Mitteln

Tabelle 4.18.: Mögliche Werte für Option `pagenumber` zur Positionierung der Paginierung innerhalb der Seitenstile

| |
|---|
| `bot`, `foot`<br>Seitenzahl im Fuß ohne Änderung der horizontalen Position |
| `botcenter`, `botcentered`, `botmittle`, `footcenter`, `footcentered`, `footmiddle`<br>Seitenzahl zentriert innerhalb des Fußes |
| `botleft`, `footleft`<br>Seitenzahl links im Fuß |
| `botright`, `footright`<br>Seitenzahl rechts im Fuß |
| `center`, `centered`, `middle`<br>Seitenzahl zentriert ohne Änderung der vertikalen Position |
| `false`, `no`, `off`<br>keine Seitenzahl |
| `head`, `top`<br>Seitenzahl im Kopf ohne Änderung der horizontalen Position |
| `headcenter`, `headcentered`, `headmiddle`, `topcenter`, `topcentered`, `topmiddle`<br>Seitenzahl zentriert innerhalb des Kopfes |
| `headleft`, `topleft`<br>Seitenzahl links im Kopf |
| `headright`, `topright`<br>Seitenzahl rechts im Kopf |
| `left`<br>Seitenzahl links ohne Änderung der vertikalen Position |
| `right`<br>Seitenzahl rechts ohne Änderung der vertikalen Position |

gesetzt werden. Die Klasse scrlttr2 verlässt sich bei diesem Seitenstil auf den LaTeX-Kern, während bei scrletter der Stil von scrlayer bereitgestellt wird.

scrlttr2 `headings` ist bei scrlttr2 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 203 und Seite 228). 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 scrlayer-scrpage auch \markleft und \markdouble (siehe Abschnitt 5.5, Seite 291), zur Verfügung.

scrletter Da scrletter intern scrlayer-scrpage verwendet, wird ein eventuell von der Klasse bereitgestellter Seitenstil headings als Alias von scrheadings umdefiniert. Näheres zu diesem Seitenstil ist in Kapitel 5 auf Seite 273 zu erfahren.

scrletter `letter` wird nur von scrletter definiert, da der Seitenstil `headings` im allgemeinen bereits von den Klassen belegt ist. Dies geschieht mit Hilfe von scrlayer-scrpage aus Kapitel 5, Seite 267. Bei der Einstellung automark=true übernimmt `letter` dann die Rolle, die bei scrlttr2 von headings ausgefüllt wird. Bei automark=false übernimmt `letter` dagegen die Rolle von myheadings.

Durch die Verwendung von scrlayer-scrpage kann das veraltete Paket scrpage2 oder das mit KOMA-Script wenig kompatible fancyhdr nicht zusammen mit scrletter verwendet werden.

scrlttr2 `myheadings` ist bei scrlttr2 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 scrlayer-scrpage stehen außerdem \markleft und \markdouble zur Verfügung.

scrletter Bei scrletter übernimmt der Seitenstil letter ebenfalls die Rolle von `myheadings`.

scrlttr2 `plain` ist bei scrlttr2 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.

scrletter Da scrletter intern scrlayer-scrpage verwendet, wird der Seitenstil plain als Alias von plain.scrheadings umdefiniert. Näheres zu diesem Seitenstil ist in Kapitel 5 auf Seite 273 zu erfahren.

scrletter `plain.letter` wird von scrletter zusammen mit letter definiert. Dies geschieht mit Hilfe von scrlayer-scrpage. Nach der Aktivierung von letter bis zum Ende des Briefes ist `plain` dann ein Alias für diesen Stil.

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. KOMA-Script verwendet `\thispagestyle{empty}` selbst innerhalb von \opening für die erste Briefseite.

Für die Schriftart von Kopf und Fuß der Seite gibt es das gemeinsame Element `pagehheadfoot`. Bei Verwendung von scrlayer-scrpage und damit auch bei Verwendung von scrletter wird im Kopf zusätzlich das Element `pagehead` verwendet. Ohne das Paket ist bei scrlttr2 hingegen `pagehead` eine alternative Bezeichnung für `pageheadfoot`. Für den Fuß ist zusätzlich das Element `pagefoot` zuständig, das nach `pageheadfoot` in mit Variable `nextfoot` oder per Paket scrlayer-scrpage (siehe Kapitel 5, Seite 278) definierten Seitenstilen zur Anwendung kommt. Das Element für die Seitenzahl innerhalb des Kopfes oder Fußes heißt `pagenumber`. Die Voreinstellungen sind in Tabelle 3.8, Seite 86 zu finden. Beachten Sie dazu auch das Beispiel aus Abschnitt 3.12, Seite 86.

v3.00

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

In den meisten Fällen werden die Möglichkeiten, die KOMA-Script über Optionen und Variablen für die Gestaltung des Seitenkopfes und -fußes auf Folgeseiten zur Verfügung stellt, vollkommen ausreichen. Dies gilt umso mehr, als man zusätzlich mit `\markboth` und `\markright` die Möglichkeit hat, die Angaben zu ändern, die KOMA-Script in den Kopf setzt. Die Anweisungen `\markboth` und `\markright` können insbesondere mit dem Seitenstil `myheadings` beziehungsweise `letter` genutzt werden. Bei Verwendung des Pakets scrlayer-scrpage gilt dies auch für den Seitenstil `scrheadings`. Außerdem stehen dann die Anweisungen `\markleft` und `\markdouble` zur Verfügung (siehe Abschnitt 5.5, Seite 291).

```
\setkomavar{nexthead}[Bezeichnung]{Inhalt}
\setkomavar{nextfoot}[Bezeichnung]{Inhalt}
```

In einigen wenigen Fällen will man 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 bei scrlttr2 im Seitenstil `headings` oder `myheadings` und bei scrletter im Seitenstil `letter` frei. Dazu setzt man den gewünschten Aufbau als *`Inhalt`* der Variablen `nexthead` beziehungsweise `nextfoot`.

scrlttr2
scrletter
v3.08

Da die Seitenstile von scrlttr2 nicht für mehrzeilige Köpfe und Füße ausgelegt sind, geben sie die beider Variablen in einer horizontalen Box aus. Damit sind zunächst weder Absätze noch Zeilenumbrüche oder ähnliches möglich. Innerhalb des Inhalts von `nexthead` und `nextfoot` können aber beispielsweise mit Hilfe der `\parbox`-Anweisung (siehe [Tea23]) 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 *`Inhalt`* mit Hilfe von `\usekomavar` auch auf weitere Variablen zugegriffen werden.

Als Alternative für mehrzeilige Köpf und Füße bietet sich die Verwendung des in Kapitel 5 beschriebenen Pakets scrlayer-scrpage an. Das Paket scrletter verwendet für die Definition von Seitenstil `letter` ohnehin bereits scrlayer-scrpage und ist daher von der genannten Beschränkung nicht betroffen.

Die *Bezeichnung* wird von KOMA-Script bei beiden Variablen nicht genutzt.

## 4.14. Vakatseiten

Es gilt sinngemäß, was in Abschnitt 3.13 geschrieben wurde. Falls Sie also Abschnitt 3.13 bereits gelesen und verstanden haben, können Sie auf Seite 247 mit Abschnitt 4.15 fortfahren.

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 KOMA-Script auch für den Fall von doppelseitigen Briefen Einstellungen für Vakatseiten. Da die hier vorgestellten Anweisungen aber in Briefen kaum Verwendung finden, wurde hier auf Beispiele verzichtet. Bei Bedarf orientieren Sie sich bitte an den Beispielen in Abschnitt 3.13 ab Seite 90.

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

v3.00 Mit Hilfe dieser Option kann man den *Seitenstil* der Vakatseite bestimmen, die bei Bedarf von den Anweisungen `\cleardoublepage`, `\cleardoubleoddpage` oder `\cleardoubleevenpage` eingefügt wird, um bis zur gewünschten Seite zu umbrechen. Als *Seitenstil* sind dabei alle bereits definierten Seitenstile (siehe Abschnitt 4.13 ab Seite 240 und Kapitel 5 ab Seite 267) 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.

v3.00 Ab Version 3.00 werden in der Voreinstellung entsprechend der typografischen Gepflogenheiten Vakatseiten mit dem Seitenstil `empty` erzeugt.

```
\clearpage
\cleardoublepage
\cleardoublepageusingstyle{Seitenstil}
\cleardoubleemptypage
\cleardoubleplainpage
\cleardoublestandardpage
\cleardoubleoddpage
\cleardoubleoddpageusingstyle{Seitenstil}
\cleardoubleoddemptypage
\cleardoubleoddplainpage
\cleardoubleoddstandardpage
\cleardoubleevenpage
\cleardoubleevenpageusingstyle{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 2.6, 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 Seitenkopf zu unterdrücken. Analog dazu wird bei der Anweisung `\cleardoubleoddemptypage` der Seitenstil `empty` verwendet, um sowohl Seitenkopf als auch Seitenfuß 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 `\cleardoubleoddpageusingstyle` anzugeben. Dabei kann jeder bereits definierte Seitenstil (siehe auch Kapitel 5) verwendet werden.

Die Arbeitsweise der Anweisungen `\cleardoublestandardpage`, `\cleardoubleemptypage`, `\cleardoubleplainpage` und der ein Argument erwartenden Anweisung `\cleardoublepageusingstyle` entspricht bei der Klasse scrlttr2 ebenso wie die Standard-Anweisung `\cleardoublepage` den entsprechenden, zuvor erklärten Anweisungen. Die übrigen Anweisungen sind bei scrlttr2 nur aus Gründen der Vollständigkeit definiert. Näheres zu diesen ist bei Bedarf Abschnitt 3.13, Seite 92 zu entnehmen.

Im doppelseitigen Satz führt `\cleardoubleoddpage` immer zur nächsten ungeraden Seite und `\cleardoubleevenpage` zur nächsten geraden Seite. Eine gegebenenfalls einzufügende Vakatseite wird mit dem über Option `cleardoublepage` festgelegten Seitenstil ausgegeben.

## 4.15. Fußnoten

Es gilt sinngemäß, was in Abschnitt 3.14 geschrieben wurde. Falls Sie also Abschnitt 3.14 bereits gelesen und verstanden haben, können Sie auf Seite 250 mit Abschnitt 4.16 fortfahren.

scrletter Wird keine KOMA-Script-Klasse verwendet, stützt sich scrletter auf das Paket scrextend. Siehe daher bei Verwendung von scrletter ebenfalls Abschnitt 7.11 ab Seite 313.

Die Anweisungen zum Setzen von Fußnoten sind in jeder LaTeX-Einführung, beispielsweise [DGS+12], zu finden. KOMA-Script bietet darüber hinaus aber auch noch die Möglichkeit, die Form der Fußnoten zu verändern.

Da Fußnoten in Briefen eher selten verwendet werden, wurde auf Beispiele in diesem Abschnitt weitgehend verzichtet. Sollten Sie Beispiele benötigen, können Sie solche in Abschnitt 3.14 ab Seite 93 finden.

`footnotes=`*`Einstellung`*
`\multfootsep`

v3.00 Fußnoten werden im Text in der Voreinstellung mit kleinen, hochgestellten Ziffern markiert. Werden in der Voreinstellung `footnotes=nomultiple` 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 handele.

Mit `footnotes=multiple` werden Fußnoten, die unmittelbar aufeinander folgen, stattdessen mit einem Trennzeichen aneinander gereiht. Das in `\multfootsep` definierte Trennzeichen ist als

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

definiert. Es ist also mit einem Komma vorbelegt. Dieses kann umdefiniert werden.

Der gesamte Mechanismus ist kompatibel zu footmisc, Version 5.3d bis 5.5b (siehe [Fai11]) 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 94.

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

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 zuvor erklärten Anweisung \multfootsep gespeichert.

Beispiele und ergänzende Hinweise sind Abschnitt 3.14 ab Seite 95 zu entnehmen.

\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. 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. Ein passendes Beispiel ist in Abschnitt 3.14 auf Seite 96 zu finden. Ab LaTeX 2021-05-01 wird die Anweisung übrigens von LaTeX selbst bereitgestellt.

v3.33

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

KOMA-Script setzt 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. 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 auf Seite 97 veranschaulicht die verschiedenen Parameter. 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.

Auf die Fußnote einschließlich der Markierung findet außerdem die für das Element `footnote` eingestellte Schriftart Anwendung. Die Schriftart der Markierung kann jedoch mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) für das Element `footnotelabel` davon abweichend eingestellt werden. Siehe hierzu auch Tabelle 3.2, Seite 61. Voreingestellt ist jeweils keine Umschaltung der Schrift. Bitte missbrauchen Sie das Element nicht für andere Zwecke, beispielsweise zur Verwendung von Flattersatz in den Fußnoten (siehe `\raggedfootnote`, Seite 250).

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}}
```

Dabei findet die Schriftart für das Element `footnotereference` Anwendung (siehe Tabelle 4.3, Seite 192). Die Markierungen im Text und in der Fußnote selbst sind also identisch. Die Schriftart kann mit den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) jedoch geändert werden.

Beispiele für die Verwendung von `\deffootnote` finden Sie in Abschnitt 3.14, Seite 97.

`\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 4.9, Seite 192) 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.

scrletter Verwendet man das Paket scrletter nicht mit einer KOMA-Script-Klasse, sondern beispielsweise mit einer Standardklasse, so existieren Anweisung `\setfootnoterule` und Element `footnoterule` nicht.

`\raggedfootnote`

v3.23 In der Voreinstellung werden die Fußnoten bei KOMA-Script genau wie bei den Standardklassen im Blocksatz gesetzt. Es ist aber auch möglich, die Formatierung abweichend vom restlichen Dokument zu ändern. Dazu ist `\raggedfootnote` umzudefinieren. Gültige Definitionen wären `\raggedright`, `\raggedleft`, `\centering`, `\relax` oder entsprechend der Voreinstellung eine leere Definition. Auch die Ausrichtungsbefehle des Pakets ragged2e sind zulässig (siehe [Sch09]). Ein passendes Beispiel ist in Abschnitt 3.14, Seite 98 zu finden.

```
\ExecuteDoHook{footnote/text/begin}
\ExecuteDoHook{footnote/text/end}
```

v3.36 Für Experten gibt es außerdem zwei Haken vom Typ *do-hook* (siehe Abschnitt 10.8 ab Seite 349). Der erste davon wird ganz zu Beginn von `\@makefntext` noch vor `\raggedfootnote` ausgeführt. Der zweite am Ende noch bevor der Absatz beendet ist. Derzeit werden beide Haken von KOMA-Script selbst nicht verwendet.

## 4.16. Listen

Es gilt sinngemäß, was in Abschnitt 3.18 geschrieben wurde. Falls Sie also Abschnitt 3.18 bereits gelesen und verstanden haben, können Sie auf Seite 254 mit Abschnitt 4.17 fortfahren.

scrletter Das Paket scrletter definiert selbst keine Listenumgebungen, sondern überlässt diese der verwendeten Klasse. Ist dies keine KOMA-Script-Klasse, so lädt es scrextend. Allerdings werden vom Paket scrextend nur die Umgebungen `labeling`, `addmargin` und `addmargin*` definiert.

Alle anderen Listenumgebungen bleiben der Verantwortung der verwendeten Klasse überlassen.

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.

Auf Beispiele wurde in diesem Abschnitt verzichtet. Sie finden solche in Abschnitt 3.18 ab Seite 126 oder in jeder LaTeX-Einführung.

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

scrlttr2

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.

v3.33

Sie können diese leicht mit `\renewcommand` umdefinieren. Die Schriftart der Zeichen für die einzelnen Ebenen kann außerdem bei den KOMA-Script-Klassen mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) für die Elemente `labelitemi`, `labelitemii`, `labelitemiii` und `labelitemiv` geändert werden. In der Voreinstellung verwenden diese alle die Schrifteinstellung für Element `itemizelabel`. Lediglich Element `labelitemii` verwendet zusätzlich `\bfseries`. Die Voreinstellung von `itemizelabel` selbst ist `\normalfont`. Die einzelnen Stichpunkte der Umgebung werden mit `\item` eingeleitet. In Abschnitt 3.18 auf Seite 126 finden Sie ein Beispiel für die Anwendung.

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

scrlttr2

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

v3.44

Schriftart der Nummern beziehungsweise Label für die einzelnen Ebenen kann außerdem bei den KOMA-Script-Klassen mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) für die Elemente `labelenumi`, `labelenumii`, `labelenumiii` und `labelenumiv` geändert werden. In der Voreinstellung verwenden diese alle die Schrifteinstellung für Element `enumeratelabel`. Die Voreinstellung für `enumeratelabel` selbst ist leer, also keine Änderung der Schrift. Die einzelnen Stichpunkte werden wieder mit `\item` eingeleitet. In Abschnitt 3.18 auf Seite 129 finden Sie ein Beispiel für die Anwendung.

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

scrlttr2

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 4.9, Seite 192) für das Element `descriptionlabel` (siehe Tabelle 4.3, Seite 192) geändert werden. In der Voreinstellung wird `\maybesffamily\bfseries` verwendet. In Abschnitt 3.18 auf Seite 129 finden Sie ein Beispiel für die Anwendung.

v2.8p

v3.39

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

Eine andere Form der Stichwortliste ist nur bei den KOMA-Script-Klassen und scrextend vorhanden: die `labeling`-Umgebung. Im Unterschied zur zuvor vorgestellten Umgebung `description` kann bei `labeling` ein Muster angegeben werden, dessen Länge die Einrücktiefe aller Stichpunkte ergibt. Zwischen Stichpunkt und Beschreibungstext kann ein optionales

v3.02

*Trennzeichen* festgelegt werden. Die Schriftart, die für die Hervorhebung des Schlüsselworts verwendet wird, kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) für das Element `labelinglabel` (siehe Tabelle 4.3, Seite 192) geändert werden. Für die davon abweichende Schriftart der Trennzeichen ist das Element `labelingseparator` (siehe ebenfalls Tabelle 4.3, Seite 192) zuständig. Beide sind in der Voreinstellung leer, es findet also keine Schriftumschaltung statt. In Abschnitt 3.18 auf Seite 130 finden Sie ein Beispiel für die Anwendung der Umgebung.

scrlttr2

```
\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 `\\*` anstelle von \\.

In Abschnitt 3.18 auf Seite 131 finden Sie ein Beispiel für die Anwendung.

scrlttr2

```
\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. Wollen Sie dort den Einzug verhindern, müssen Sie die `\noindent`-Anweisung voranstellen. In Abschnitt 3.18 auf Seite 133 finden Sie ein Beispiel für die Anwendung.

```
\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 soll. 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. Im Beispiel in Abschnitt 3.18 auf Seite 135 wird dies gezeigt.

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 4.12, Seite 239) und \ifthispagewasodd (Abschnitt 18.1, Seite 481).

## 4.17. Mathematik

Da in Briefen ausladende Mathematik in Form nummerierter Gleichungen und Formeln eher ungewöhnlich ist, gibt es bei scrlttr2 die Optionen leqno und fleqn, die in Abschnitt 3.19, Seite 136 für scrbook, scrreprt und scrartcl dokumentiert sind, nicht. Dennoch können die vom LaTeX-Kern oder von Zusatzpaketen wie amsmath bereitgestellten Mathematikumgebungen verwendet werden.

## 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 ist stattdessen zu raten, eine der KOMA-Script-Klassen aus Kapitel 3 mit dem Paket scrletter zu kombinieren. In diesem Fall können Gleitumgebungen, wie für die Klasse dokumentiert, auch in Briefen

verwendet werden. Die Möglichkeiten zur Definition eigener Gleitumgebungen mit Hilfe von tocbasic, wie sie in Kapitel 13 dokumentiert sind, können ebenfalls genutzt werden.

## 4.19. Randnotizen

Es gilt sinngemäß, was in Abschnitt 3.21 geschrieben wurde. Falls Sie also Abschnitt 3.21 bereits gelesen und verstanden haben, können Sie auf Seite 255 mit Abschnitt 4.20 fortfahren.

Außer dem eigentlichen Textbereich, der normalerweise den Satzspiegel ausfüllt, existiert in Dokumenten noch die sogenannte Marginalienspalte. In dieser können Randnotizen gesetzt werden. Bei Briefen sind Randnotizen allerdings eher unüblich und sollten äußerst sparsam eingesetzt werden.

```
\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`.

Ein ausführliches Beispiel hierzu finden Sie in Abschnitt 3.21, Seite 155.

Für Experten sind in Abschnitt 18.1, Seite 481 Probleme bei der Verwendung von `\marginpar` dokumentiert. Diese gelten ebenso für `\marginline`. Darüber hinaus wird in Kapitel 16 ein Paket vorgestellt, mit dem sich auch Notizspalten mit eigenem Seitenumbruch realisieren lassen.

## 4.20. *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. KOMA-Script bietet hierfür die `lco`-Dateien an. Die Endung `lco` steht für l*etter* c*lass* o*ption*, also Briefklassenoption. Dennoch finden diese Dateien für scrletter ebenso Anwendung.

In `lco`-Dateien können alle Anweisungen verwendet werden, die auch an der Stelle im Dokument verwendet werden könnten, an der sie mit `\LoadLetterOption` geladen werden. Außerdem können interne Anweisungen verwendet werden, die für Paketautoren freigegeben sind.

KOMA-Script liegen bereits einige `lco`-Dateien bei. Die Dateien `DIN.lco`, `DINmtext.lco`, `DIN5008A.lcoDIN5008A.lco`, `DIN5008B.lcoDIN5008B.clo`, `SN.lco`, `SNleft.lco`,

v3.04 UScommercial9, UScommercial9DW und NF.lco dienen dazu, scrlttr2 und scrletter 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 Kompatibilität zu scrlettr zu verbessern. Diese Klasse wurde schon vor über 24 Jahren aus KOMA-Script entfernt. Es wird daher nicht mehr näher darauf eingegangen. 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 4.19, Seite 259 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 lppl-de.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 in Verbindung setzen. Beispiele für teilweise sehr komplexe lco-Dateien finden sich unter anderem unter [KDP] und in [Koh03].

```
\LoadLetterOption{Name}
\LoadLetterOptions{Liste von Namen}
```

scrlttr2 Bei scrlttr2 können lco-Dateien direkt über \documentclass geladen werden. 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. Das Paket scrletter bietet diese Möglichkeit nicht! Hier bleibt nur

v3.14 lco-Dateien über \LoadLetterOption oder \LoadLetterOptions zu laden. Für scrlttr2 wird dies ebenfalls ausdrücklich empfohlen!

\LoadLetterOption und \LoadLetterOptions können auch zu einem späteren Zeitpunkt, selbst nach \begin{document} und sogar innerhalb einer anderen lco-Datei verwendet werden. Der *Name* der lco-Datei wird in diesen Fällen ebenfalls ohne Endung übergeben. Während als Argument von \LoadLetterOption der *Name* von genau einer lco-Datei erwartet wird, versteht \LoadLetterOptions eine durch Komma separierte *Liste von Namen*. Die zu den Namen gehörenden lco-Dateien werden dann in der Reihenfolge der Angabe in der Liste geladen.

**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 (siehe auch den Tipp auf Seite 259):

```
\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. Er erreicht 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 Beispiel von Seite 233 erheblich zusammen:

```
\documentclass[version=last]{scrlttr2}
\LoadLetterOption{ich}
```

```
\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 jedes 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 dabei natürlich nicht, wie ein Vergleich von Abbildung 4.20, Seite 234 mit Abbildung 4.21 zeigt.

Bitte beachten Sie, dass im Beispiel in der Bankverbindung der Datei `ich.lco` für »ß« die TEX-Schreibweise »`\ss`« verwendet wurde. Dies hat seinen Grund darin, dass während des Ladens der Klasse weder ein Paket zur Sprachumschaltung, beispielsweise für die neue, deutsche Rechtschreibung mit `\usepackage[ngerman]{babel}`, noch bei älteren LATEX-Versionen ein Paket für die Eingabecodierung, beispielsweise mit `\usepackage[utf8]{inputenc}` für moderne Editoren, geladen ist. Wird mit Sicherheit eine LATEX-Version ab April 2018 verwendet und wird die `lco`-Datei UTF-8 codiert, so können Umlaute und Sonderzeichen natürlich auch direkt eingegeben werden.

In Tabelle 4.19 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 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 224). Sollten Sie damit ein Problem haben, erstellen Sie sich eine eigene `lco`-Datei, die zunächst `SN` lädt und in der `toaddrhpos` (siehe Seite 218)




Abbildung 4.21.: 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

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.

Zu den `lco`-Dateien `DIN5008A` und `DIN5008B` sei angemerkt, dass die entsprechenden Vorschriften gewisse Spielräume aufweisen und, wie diversen Anfragen beim Autor zu entnehmen ist, viele Anwender diese nicht nur auszureizen wünschen, sondern auch die eine oder andere Abweichung von der Norm bevorzugen. Die beiden Dateien implementieren jedoch jeweils nur eine einzige Interpretation der Norm. Der Leser sei daher daran erinnert, dass diese Dateien lediglich als Vorlagen zu begreifen sind, um das Erstellen eigener angepasster `lco`-Dateien zu erleichtern.

Tabelle 4.19.: Vordefinierte `lco`-Dateien

| | |
|---|---|
| `DIN` | voreingestellter Parametersatz für Briefe im Format A4 nach DIN 676; geeignet für Fensterbriefumschläge in den Formaten C4, C5, C6 und C6/5 (C6 lang) |

...

Tabelle 4.19.: Vordefinierte `lco`-Dateien (*Fortsetzung*)

`DIN5008A`
: experimenteller Parametersatz für Briefe angelehnt an Variante A im Format A4 nach DIN 5008; geeignet für Fensterbriefumschläge in den Formaten C4, C5, C6 und C6/5 (C6 lang)

`DIN5008B`
: experimenteller Parametersatz für Briefe angelehnt an Variante B im Format A4 nach DIN 5008; geeignet für Fensterbriefumschläge in den Formaten C6 und C6/5 (C6 lang)

`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`
: existiert nur noch aus Kompatibilitätsgründen; die Verwendung wird nicht mehr empfohlen

`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 bereitstellt.

`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)

...

Tabelle 4.19.: Vordefinierte `lco`-Dateien (*Fortsetzung*)

`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)

`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 1/2 in 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 4.19.: 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 5/8 in 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 |

## 4.21. Adressdateien und Serienbriefe

Als besonders lästig wird bei Briefen immer das Eintippen der Adressen und das Erstellen von Serienbriefen betrachtet. KOMA-Script bietet hierfür eine minimalistische Unterstützung.

`\adrentry{`*Name*`}{`*Vorname*`}{`*Adresse*`}{`*Tel.*`}{`*F1*`}{`*F2*`}{`*Kommentar*`}{`*Kürzel*`}`

Mit scrlttr2 und scrletter 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. KOMA-Script 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*`}`

Bevor Klagen aufkommen, dass insgesamt nur zwei freie Felder zu wenig seien: KOMA-Script verfügt 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 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}

\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*}

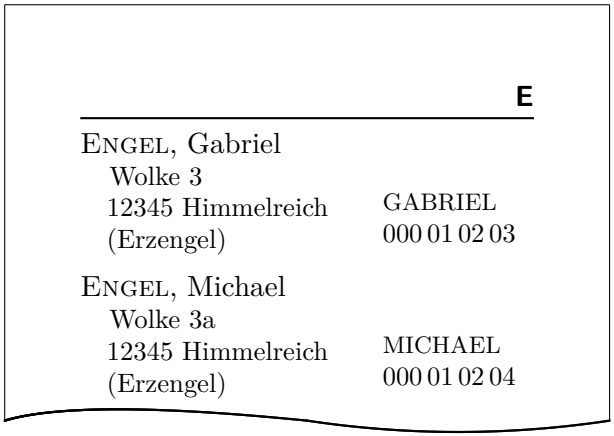
Es ist auch möglich, die Informationen einer `.adr`-Datei in Adressverzeichnisse oder Telefonlisten umzuwandeln. Sie benötigen dazu zusätzlich das adrconv-Paket von Axel Kielhorn (siehe [Kie10]). In diesem Paket sind interaktive LaTeX-Dokumente enthalten, mit deren Hilfe sehr einfach entsprechende Listen 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 und scrletter selbst ignorieren 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}
\adrchar{K}
\adrentry{Kohm}{Markus}
         {Freiherr-von-Drais-Stra\ss e 66\\
           68535 Edingen-Neckarhausen}
         {+49~62\,03~1\,??\,??}{}{}
         {\"Uberhaupt kein Engel}
         {KOMA}
```

Diese verarbeiten Sie nun unter Verwendung des Dokuments `adrdir.tex` aus [Kie10]. Eine Seite des Ergebnisses sieht dann etwa so aus:




Dabei wird der Buchstabe in der Kopfzeile von `\adrchar` erzeugt, wenn man die Frage »Namen in der Kopfzeile?« verneint. Siehe dazu die Definition in `adrdir.tex`.

Das Paket adrconv kann auch dazu verwendet werden, um aus Adressdatenbanken im BibTeX-Format mit Einträgen wie:

```
@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},
}
```

Adressdateien für die KOMA-Script-Briefklasse oder das KOMA-Script-Briefpaket zu erzeugen. Näheres zum adrconv-Paket ist der zugehörigen Anleitung zu entnehmen.

v3.12

# Kopf- und Fußzeilen mit scrlayer-scrpage

Das Paket scrlayer ist einer der grundlegenden Bausteine in KOMA-Script. Es bietet ein Ebenenmodell und darauf basierend ein Seitenstil-Modell. Für die direkte Verwendung durch den durchschnittlichen Anwender ist die Schnittstelle dieses Pakets in seiner Mächtigkeit jedoch nicht leicht zu durchschauen. Aufbauend auf scrlayer bietet scrlayer-scrpage eine Benutzerschnittstelle, die einigen Anwendern teilweise vielleicht noch vom veralteten Paket scrpage2 vertraut ist. Allerdings wurde jene Schnittstelle stark erweitert, vereinheitlicht und auf allgemeine Neuerungen in KOMA-Script 3 angepasst. Einige wenige Möglichkeiten, die eigentlich zu scrlayer und daher in Kapitel 14 gehören, sind auch in diesem Kapitel dokumentiert, da sie für die Verwendung von scrlayer-scrpage ebenfalls wichtig sind. Wer bereits mit scrpage2 vertraut ist und dort keine unsauberen Rückgriffe auf interne Anweisungen getätigt hat, kann scrpage2 recht einfach durch scrlayer-scrpage ersetzen.

Neben scrlayer-scrpage ist auch beispielsweise fancyhdr (siehe [vO04]) grundsätzlich geeignet, um Kopf und Fuß der Seiten zu konfigurieren. Allerdings unterstützt dieses Paket diverse Möglichkeiten von KOMA-Script, angefangen von Änderungen der Schrift über das Elemente-Modell (siehe `\setkomafont`, `\addtokomafont` und `\usekomafont` in Abschnitt 5.3, ab Seite 270) bis hin zum konfigurierbaren Format der Gliederungsnummern in Kolumnentiteln (siehe Option `numbers` und beispielsweise Anweisung `\chaptermarkformat` in Abschnitt 3.16, Seite 105 und Seite 120), nicht. Daher wird für die Verwendung mit den KOMA-Script-Klassen scrlayer-scrpage empfohlen. Natürlich ist scrlayer-scrpage auch mit anderen Klassen, beispielsweise den Standardklassen, eine gute Wahl.

Über die in diesem Kapitel erklärten Möglichkeiten hinaus bietet das Paket scrlayer-scrpage weiteres, das jedoch nur für einige wenige Anwender von Interesse sein dürfte und daher in Kapitel 15 von Teil II ab Seite 451 ausgeführt wird. Dennoch: Falls die hier in Teil I dokumentierten Möglichkeiten für Sie nicht ausreichen, sei Ihnen auch jenes Kapitel nahegelegt.

## 5.1. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde. Falls Sie also Abschnitt 2.4 bereits gelesen und verstanden haben, können Sie auf Seite 269 mit Abschnitt 5.2 fortfahren.

```
\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`

v3.00 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 [Tea23] oder jeder LaTeX-Einführung, beispielsweise [DGS+12], beschrieben.

Gegenüber der nachfolgend vorgestellten Schnittstelle zu Einstellungen von Optionen hat \documentclass einen Nachteil, der unbedingt zu beachten ist: Anweisungen, Längen, Zähler und ähnliches können darin leicht zerbrechen. So führt die Verwendung einer LaTeX-Länge im Wert einer Option bei dieser Anweisung bei vielen Nicht-KOMA-Script-Klassen zu einer Fehlermeldung, noch bevor der Wert an ein KOMA-Script-Paket übergeben wird, es also die Kontrolle darüber übernehmen könnte. Wertzuweisungen mit LaTeX-Längen oder LaTeX-Zählern sollten daher nie per \documentclass, sondern mit den nachfolgend dokumentierten Anweisungen \KOMAoptions oder \KOMAoption vorgenommen werden.

```
\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. Mit der Anweisung \KOMAoptions kann man wie bei \documentclass oder \usepackage 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 \KOMAoption der einen *Option* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *Werteliste* durch Komma voneinander getrennt.

Soll ein *Wert* ein Gleichheitszeichen oder ein Komma enthalten, 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«. Siehe dazu Teil II, Abschnitt 10.2, ab Seite 329.

Mit \KOMAoptions oder \KOMAoption gesetzte Optionen erreichen sowohl die KOMA-Script-Klasse als auch alle bereits geladenen KOMA-Script-Pakete, die diese Optionen kennen. Ist eine Option oder ein Wert insgesamt unbekannt, so wird die Option einschließlich des Wertes von scrbase als fehlerhaft gemeldet.

## 5.2. Höhe von Kopf und Fuß

Vermutlich, weil der Fuß bei den Standardklassen kaum besetzt und immer einzeilig ist, gibt es bei LaTeX keine definierte Höhe des Fußes. Zwar ist der Abstand von der letzten Grundlinie des Textblocks zur Grundlinie des Fußes mit `\footskip` definiert. Wenn allerdings der Fuß höher als eine Zeile wird, ist nicht hinreichend festgelegt, wie sich diese Höhe niederschlägt bzw. ob `\footskip` den Abstand zur obersten oder untersten Grundlinie des Fußes darstellt.

Obwohl auch der Kopf bei den Seitenstilen der Standardklassen in einer horizontalen Box ausgegeben wird und damit immer einzeilig ist, hat LaTeX für die Kopfhöhe selbst eine Länge zur Einstellung ihrer Höhe vorgesehen. Dies erklärt sich vermutlich daraus, dass diese Höhe zur Bestimmung des Anfangs des Textbereichs benötigt wird.

`\footheight`
`\headheight`
`autoenlargeheadfoot=`*`Ein-Aus-Wert`*

Das Paket scrlayer führt als neue Länge `\footheight` analog zur Höhe `\headheight` ein. Gleichzeitig interpretiert scrlayer-scrpage `\footskip` so, dass es den Abstand der letzten Grundlinie des Textbereichs von der ersten Standard-Grundlinie des Fußes darstellt. Das Paket typearea betrachtet dies in gleicher Weise, so dass die dortigen Optionen zum Setzen der Höhe des Fußes (siehe die Optionen `footheight` und `footlines` in Abschnitt 2.6, Seite 45) und zur Berücksichtigung des Fußes bei der Berechnung des Satzspiegels (siehe Option `footinclude` in demselben Abschnitt, Seite 42) sehr gut zum Setzen der Werte für scrlayer verwendet werden können und auch das gewünschte Ergebnis liefern.

Wird das Paket typearea nicht verwendet, so sollte man gegebenenfalls die Höhe von Kopf und Fuß über entsprechende Werte für die Längen einstellen. Zumindest für den Kopf bietet aber beispielsweise auch das Paket geometry Einstellmöglichkeiten.

Wurde der Kopf oder der Fuß für den tatsächlich verwendeten Inhalt zu klein gewählt, so versucht scrlayer-scrpage in der Voreinstellung die Längen selbst entsprechend anzupassen. Gleichzeitig wird eine entsprechende Warnung ausgegeben, die auch Ratschläge für passende Einstellungen enthält. Die automatischen Änderungen haben dann ab dem Zeitpunkt, an dem ihre Notwendigkeit erkannt wurde, Gültigkeit und werden nicht automatisch aufgehoben, wenn beispielsweise der Inhalt von Kopf oder Fuß wieder kleiner wird. Über Option `autoenlargeheadfoot` kann dieses Verhalten jedoch geändert werden. Die Option versteht die Werte für einfache Schalter aus Tabelle 2.5, Seite 41. In der Voreinstellung ist die Option aktiviert. Wird sie deaktiviert, so werden Kopf und Fuß nicht mehr automatisch vergrößert, sondern nur noch eine Warnung mit Hinweisen für passende Einstellungen ausgegeben.

v3.25

## 5.3. Textauszeichnungen

Es gilt sinngemäß, was in Abschnitt 3.6 geschrieben wurde. Falls Sie also Abschnitt 3.6 bereits gelesen und verstanden haben, können Sie sich auf Tabelle 5.1, Seite 271 beschränken und ansonsten auf Seite 273 mit Abschnitt 5.4 fortfahren.

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

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 Schriftattribute umgeschaltet werden. In der Regel werden dies Befehle wie `\rmfamily`, `\sffamily`, `\ttfamily`, `\upshape`, `\itshape`, `\slshape`, `\scshape`, `\mdseries`, `\bfseries`, `\normalfont` oder einer der Befehle `\Huge`, `\huge`, `\LARGE`, `\Large`, `\large`, `\normalsize`, `\small`, `\footnotesize`, `\scriptsize` und `\tiny` sein. Die Erklärung zu diesen Befehlen entnehmen Sie bitte [DGS+12], [Tea23] oder [Tea05]. Auch Farbumschaltungen wie `\normalcolor` sind möglich (siehe [Car17] 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 und deren Voreinstellungen sind in Tabelle 5.1 aufgelistet. Die angegebenen Voreinstellungen gelten nur, wenn das jeweilige Element beim Laden von scrlayer-scrpage nicht bereits definiert ist. Beispielsweise definieren die KOMA-Script-Klassen `pageheadfoot` und es gilt dann die von scrlayer-scrpage vorgefundene Einstellung.

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

Tabelle 5.1.: Elemente, deren Schrift bei scrlayer-scrpage mit `\setkomafont` und `\addtokomafont` verändert werden kann, einschließlich der jeweiligen Voreinstellung, falls die Elemente beim Laden von scrlayer-scrpage nicht bereits definiert sind

`footbotline`
: Linie unter dem Fuß eines mit scrlayer-scrpage definierten Seitenstils (siehe Abschnitt 5.5, Seite 293). Das Element wird nach `\normalfont` und nach den Elementen `pageheadfoot` und `pagefoot` angewandt. Es wird empfohlen, dieses Element lediglich für Farbänderungen zu verwenden.
Voreinstellung: *leer*

`footsepline`
: Linie über dem Fuß eines mit scrlayer-scrpage definierten Seitenstils (siehe Abschnitt 5.5, Seite 293). Das Element wird nach `\normalfont` und nach den Elementen `pageheadfoot` und `pagefoot` angewandt. Es wird empfohlen, dieses Element lediglich für Farbänderungen zu verwenden.
Voreinstellung: *leer*

`headsepline`
: Linie unter dem Kopf eines mit scrlayer-scrpage definierten Seitenstils (siehe Abschnitt 5.5, Seite 293). Das Element wird nach `\normalfont` und nach den Elementen `pageheadfoot` und `pagehead` angewandt. Es wird empfohlen, dieses Element lediglich für Farbänderungen zu verwenden.
Voreinstellung: *leer*

`headtopline`
: Linie über dem Kopf eines mit scrlayer-scrpage definierten Seitenstils (siehe Abschnitt 5.5, Seite 293). Das Element wird nach `\normalfont` und nach den Elementen `pageheadfoot` und `pagehead` angewandt. Es wird empfohlen, dieses Element lediglich für Farbänderungen zu verwenden.
Voreinstellung: *leer*

`pagefoot`
: Inhalt des Fußes eines mit scrlayer-scrpage definierten Seitenstils (siehe Abschnitt 5.4, Seite 278). Das Element wird nach `\normalfont` und nach dem Element `pageheadfoot` angewandt.
Voreinstellung: *leer*

...

Tabelle 5.1.: Elemente, deren Schrift verändert werden kann (*Fortsetzung*)

| | |
|---|---|
| pagehead | Inhalt des Kopfes eines mit scrlayer-scrpage definierten Seitenstils (siehe Abschnitt 5.4, Seite 274). Das Element wird nach `\normalfont` und nach Element pageheadfoot angewandt. Voreinstellung: *leer* |
| pageheadfoot | Inhalt des Kopfes oder Fußes eines mit scrlayer-scrpage definierten Seitenstils (siehe Abschnitt 5.4, Seite 274). Das Element wird nach `\normalfont` angewandt. Voreinstellung: `\normalcolor\slshape` |
| pagenumber | Die mit `\pagemark` gesetzte Paginierung (siehe Abschnitt 5.4, Seite 289). Bei einer etwaigen Umdefinierung von `\pagemark` ist dafür zu sorgen, dass die Umdefinierung auch ein `\usekomafont{pagenumber}` enthält! Voreinstellung: `\normalfont` |

```
\usefontofkomafont{Element}
\useencodingofkomafont{Element}
\usesizeofkomafont{Element}
\usefamilyofkomafont{Element}
\useseriesofkomafont{Element}
\useshapeofkomafont{Element}
```

v3.12

Manchmal werden in der Schrifteinstellung eines Elements auch Dinge vorgenommen, die mit der Schrift eigentlich gar nichts zu tun haben, obwohl dies ausdrücklich nicht empfohlen wird. Soll dann nur die Schrifteinstellung, aber keine dieser zusätzlichen Einstellungen ausgeführt werden, so kann statt `\usekomafont` die Anweisung `\usefontofkomafont` verwendet werden. Diese Anweisung übernimmt nur die Schriftgröße und den Grundlinienabstand, die Codierung (engl. *encoding*), die Familie (engl. *family*), die Strichstärke oder Ausprägung (engl. *font series*) und die Form oder Ausrichtung (engl. *font shape*).

Mit den übrigen Anweisungen können auch einzelne Schriftattribute übernommen werden. Dabei übernimmt `\usesizeofkomafont` sowohl die Schriftgröße als auch den Grundlinienabstand.

Vor dem Missbrauch der Schrifteinstellungen wird dennoch dringend gewarnt (siehe Abschnitt 18.5, Seite 485)!

## 5.4. Verwendung vordefinierter Seitenstile

Die einfachste Möglichkeit, mit scrlayer-scrpage zu seinem Wunschdesign für Kopf und Fuß der Seite zu gelangen, ist die Verwendung eines vorgefertigten Seitenstils. In diesem Abschnitt werden diese vom Paket scrlayer-scrpage bereits beim Laden definierten Seitenstile vorgestellt. Darüber hinaus werden diejenigen Befehle erklärt, mit denen grundlegende Einstellungen bei diesen Seitenstilen vorgenommen werden können. Weitere Optionen, Befehle und Hintergründe sind in den nachfolgenden Abschnitten, aber auch in Abschnitt 15.2 in Teil II zu finden.

```
\pagestyle{scrheadings}
\pagestyle{plain.scrheadings}
```

Das Paket scrlayer-scrpage stellt zwei Seitenstile zur Verfügung, die nach eigenen Wünschen umgestaltet werden können. Als erstes wäre der voreingestellte Seitenstil `scrheadings` zu nennen. Dieser ist als Seitenstil mit Kolumnentitel vorgesehen. Er ähnelt in der Voreinstellung dem Seitenstil `headings` der Standard- oder der KOMA-Script-Klassen. Was genau im Kopf und Fuß ausgegeben wird, ist über die nachfolgend beschriebenen Befehle und Optionen einstellbar.

Als zweites ist der Seitenstil `plain.scrheadings` zu nennen. Dieser ist als Seitenstil ohne Kolumnentitel vorgesehen. Er ähnelt in der Voreinstellung dem Seitenstil `plain` der Standard- oder der KOMA-Script-Klassen. Was genau im Kopf und Fuß ausgegeben wird, ist auch hier über die nachfolgend beschriebenen Befehle und Optionen einstellbar.

Natürlich kann auch `scrheadings` als Seitenstil ohne Kolumnentitel und `plain.scrheadings` als Seitenstil mit Kolumnentitel konfiguriert werden. Es ist jedoch zweckmäßig, sich an die vorgenannte Konvention zu halten. Die beiden Seitenstile beeinflussen sich nämlich in gewisser Weise gegenseitig. Sobald einer der Seitenstile einmal ausgewählt wurde, ist `scrheadings` auch unter dem Namen `headings` und der Seitenstil `plain.scrheadings` auch unter dem Namen `plain` aktivierbar. Das hat den Vorteil, dass bei Klassen, die automatisch zwischen `headings` und `plain` umschalten, durch einmalige Auswahl von `scrheadings` oder `plain.scrheadings` nun zwischen diesen beiden Stilen umgeschaltet wird. Direkte Anpassungen der entsprechenden Klassen sind nicht erforderlich. Die beiden Seitenstile stellen also quasi ein Paar dar, das als Ersatz für `headings` und `plain` verwendet werden kann. Sollten weitere solche Paare benötigt werden, so sei auf Abschnitt 15.2 in Teil II verwiesen.

```
\lehead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cehead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\rehead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\lohead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cohead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\rohead[Inhalt plain.scrheadings]{Inhalt scrheadings}
```

Was bei scrheadings und plain.scrheadings in den Kopf der Seite geschrieben wird, ist mit Hilfe dieser Befehle einstellbar. Dabei setzt das optionale Argument jeweils den Inhalt eines Elements in plain.scrheadings, während das obligatorische Argument jeweils einen Inhalt in scrheadings setzt.

Die Inhalte für gerade, also linke Seiten werden mit den Befehlen \lehead, \cehead und \rehead gesetzt. Das »e« an zweiter Stelle des Befehlsnamens steht für »*even*« (engl. für »gerade«).

Die Inhalte für ungerade, also rechte Seiten werden mit den Befehlen \lohead, \cohead und \rohead gesetzt. Das »o« an zweiter Stelle des Befehlsnamens steht für »*odd*« (engl. für »ungerade«).

Es sei an dieser Stelle noch einmal darauf hingewiesen, dass im einseitigen Satz nur rechte Seiten existieren und diese von LaTeX unabhängig von ihrer Nummer als ungerade Seiten bezeichnet werden.

Jeder Kopf eines Seitenstils besitzt ein linksbündiges Element, das mit \lehead respektive \lohead gesetzt werden kann. Das »l« am Anfang des Befehlsnamens steht hier für »*left aligned*« (engl. für »linksbündig«).

Ebenso besitzt jeder Kopf eines Seitenstils ein zentriert gesetztes Element, das mit \cehead respektive \cohead gesetzt werden kann. Das »c« am Anfang des Befehlsnamens steht hier für »*centered*« (engl. für »zentriert«).

Entsprechend besitzt jeder Kopf eines Seitenstils auch ein rechtsbündiges Element, das mit \rehead respektive \rohead gesetzt werden kann. Das »r« am Anfang des Befehlsnamens steht hier für »*right aligned*« (engl. für »rechtsbündig«).

Die drei vorgenannten Elemente nehmen im Kopf denselben Raum ein. Sie sind lediglich unterschiedlich ausgerichtet. Werden also mehrere dieser Elemente verwendet, so muss der Anwender selbst Sorge tragen, dass sie sich nicht gegenseitig überdecken.

Diese Elemente besitzen auch nicht jedes für sich eine Schriftzuordnung mit Hilfe der Befehle \setkomafont und \addtokomafont (siehe Abschnitt 5.3, Seite 270), sondern alle zusammen über das Element pagehead. Vor diesem wird außerdem noch das Element pageheadfoot angewandt. Die Voreinstellungen für diese beiden Elemente sind Tabelle 5.1, Seite 271 zu entnehmen.

In Abbildung 5.1 ist die Bedeutung der einzelnen Befehle für den Kopf der Seiten im doppelseitigen Modus noch einmal skizziert.

**Beispiel:** Angenommen, Sie verfassen einen kurzen Artikel und wollen, dass im Kopf der Seiten

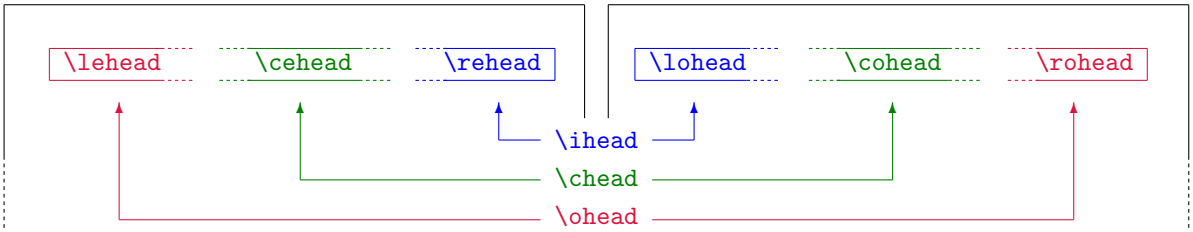



Abbildung 5.1.: Die Bedeutung der Befehle zum Setzen der Inhalte des Kopfes eines Seitenstils für die Seitenköpfe einer schematischen Doppelseite

links der Name des Autors und rechts der Titel des Artikels steht. Sie schreiben daher beispielsweise:

```
\documentclass{scrartcl}
\usepackage{scrlayer-scrpage}
\lohead{Peter Musterheinzel}
\rohead{Seitenstile mit \KOMAScript}
\pagestyle{scrheadings}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\end{document}
```

Doch, was ist das? Auf der ersten Seite erscheint nur eine Seitenzahl im Fuß, der Kopf hingegen bleibt leer!

Die Erklärung dafür ist einfach: Die Klasse scrartcl schaltet wie auch die Standardklasse article für die Seite mit dem Titelkopf in der Voreinstellung auf den Seitenstil `plain`. Nach der Anweisung `\pagestyle{scrheadings}` in der Präambel unseres Beispiels führt dies tatsächlich zur Verwendung des Seitenstils `plain.scrheadings` für die Seite mit dem Titelkopf. Dieser Seitenstil ist bei Verwendung einer KOMA-Script-Klasse mit leerem Kopf und Seitenzahl im Fuß vorkonfiguriert. Da im Beispiel das optionale Argument von `\lohead` und `\rohead` gar nicht verwendet wird, bleibt der Seitenstil `plain.scrheadings` unverändert. Das Ergebnis ist für die erste Seite also tatsächlich korrekt.

Die explizite Verwendung von `\pagestyle{scrheadings}` ist nicht einmal notwendig. Das Paket führt diese Anweisung bereits selbst beim Laden aus, stellt den Seitenstil also automatisch auf `scrheadings` um. Dabei wird auch nicht nur der Seitenstil `headings` automatisch in `scrheadings` geändert, sondern auch `plain` in `plain.scrheadings`.

Fügen Sie jetzt im Beispiel nach `\maketitle` so viel Text ein, dass eine zweite

Seite ausgegeben wird. Sie können dazu auch einfach `\usepackage{lipsum}` in der Dokumentpräambel und `\lipsum` nach `\maketitle` ergänzen. Wie Sie sehen werden, enthält der Kopf der zweiten Seite nun, genau wie gewünscht, den Namen des Autors und den Titel des Dokuments.

Zum Vergleich sollten Sie zusätzlich das optionale Argument der Anweisungen `\lohead` und `\rohead` mit einem Inhalt versehen. Ändern Sie das Beispiel dazu wie folgt ab:

```
\documentclass{scrartcl}
\usepackage{scrlayer-scrpage}
\lohead[Peter Musterheinzel]
       {Peter Musterheinzel}
\rohead[Seitenstile mit \KOMAScript]
       {Seitenstile mit \KOMAScript}
\usepackage{lipsum}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\lipsum
\end{document}
```

Jetzt haben Sie den Kopf auch auf der ersten Seite direkt über dem Titelkopf selbst. Das kommt daher, dass Sie mit den beiden optionalen Argumenten den Seitenstil `plain.scrheadings` nun ebenfalls entsprechend konfiguriert haben. Wie Sie am Ergebnis vermutlich auch erkennen, ist es jedoch besser, diesen Seitenstil unverändert zu lassen, da der Kolumnentitel über dem Titelkopf doch eher störend ist.

Alternativ zur Konfigurierung von `plain.scrheadings` hätte man bei Verwendung einer KOMA-Script-Klasse übrigens auch den Seitenstil für Seiten mit Titelkopf ändern können. Siehe dazu `\titlepagestyle` in Abschnitt 3.12, Seite 87.

Sie sollten niemals die Überschrift oder die Nummer einer Gliederungsebene mit Hilfe einer dieser Anweisungen als Kolumnentitel in den Kopf der Seite setzen. Aufgrund der Asynchronizität von Seitenaufbau und Seitenausgabe kann sonst die falsche Nummer oder die falsche Überschrift im Kolumnentitel ausgegeben werden. Stattdessen ist der Mark-Mechanismus, idealer Weise in Verbindung mit den Automatismen aus dem nächsten Abschnitt, zu verwenden.

```
\lehead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cehead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\rehead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\lohead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cohead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\rohead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
```

v3.14

Die Sternvarianten der zuvor erklärten Befehle unterscheiden sich von der Form ohne Stern lediglich bei Weglassen des optionalen Arguments `[Inhalt plain.scrheadings]`. Während die Form ohne Stern in diesem Fall den Inhalt von plain.scrheadings unangetastet lässt, wird bei der Sternvariante dann das obligatorische Argument *Inhalt scrheadings* auch für plain.scrheadings verwendet. Sollen also beide Argumente gleich sein, so kann man einfach die Sternvariante mit nur einem Argument verwenden.

**Beispiel:** Mit der Sternform von \lohead und \rohead lässt sich das Beispiel aus der vorherigen Erklärung etwas verkürzen:

```
\documentclass{scrartcl}
\usepackage{scrlayer-scrpage}
\lohead*{Peter Musterheinzel}
\rohead*{Seitenstile mit \KOMAScript}
\usepackage{lipsum}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\lipsum
\end{document}
```

```
\lefoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cefoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
\refoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
\lofoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cofoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
\rofoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
```

Was bei scrheadings und plain.scrheadings in den Fuß der Seite geschrieben wird, ist mit Hilfe dieser Befehle einstellbar. Dabei setzt das optionale Argument jeweils den Inhalt eines Elements in plain.scrheadings, während das obligatorische Argument jeweils einen Inhalt in scrheadings setzt.

Die Inhalte für gerade, also linke Seiten werden mit den Befehlen `\lefoot`, `\cefoot` und `\refoot` gesetzt. Das »`e`« an zweiter Stelle des Befehlsnamens steht für »*even*« (engl. für »gerade«).

Die Inhalte für ungerade, also rechte Seiten werden mit den Befehlen `\lofoot`, `\cofoot` und `\rofoot` gesetzt. Das »`o`« an zweiter Stelle des Befehlsnamens steht für »*odd*« (engl. für »ungerade«).

Es sei an dieser Stelle noch einmal darauf hingewiesen, dass im einseitigen Satz nur rechte Seiten existieren und diese von LaTeX unabhängig von ihrer Nummer als ungerade Seiten bezeichnet werden.

Jeder Fuß eines Seitenstils besitzt ein linksbündiges Element, das mit `\lefoot` respektive `\lofoot` gesetzt werden kann. Das »`l`« am Anfang des Befehlsnamens steht hier für »*left aligned*« (engl. für »linksbündig«).

Ebenso besitzt jeder Fuß eines Seitenstils ein zentriert gesetztes Element, das mit `\cefoot` respektive `\cofoot` gesetzt werden kann. Das »`c`« am Anfang des Befehlsnamens steht hier für »*centered*« (engl. für »zentriert«).

Entsprechend besitzt jeder Fuß eines Seitenstil auch ein rechtsbündiges Element, das mit `\refoot` respektive `\rofoot` gesetzt werden kann. Das »`r`« am Anfang des Befehlsnamens steht hier für »*right aligned*« (engl. für »rechtsbündig«).

Die drei vorgenannten Elemente nehmen im Fuß denselben Raum ein. Sie sind lediglich unterschiedlich ausgerichtet. Werden also mehrere dieser Elemente verwendet, so muss der Anwender selbst Sorge tragen, dass sie sich nicht gegenseitig überdecken.

Diese Elemente besitzen auch nicht jedes für sich eine Schriftzuordnung mit Hilfe der Befehle `\setkomafont` und `\addtokomafont` (siehe Abschnitt 5.3, Seite 270), sondern alle zusammen über das Element `pagefoot`. Vor diesem wird außerdem noch das Element `pageheadfoot` angewandt. Die Voreinstellungen für diese beiden Elemente sind Tabelle 5.1, Seite 271 zu entnehmen.

In Abbildung 5.2 ist die Bedeutung der einzelnen Befehle für den Fuß der Seiten im doppelseitigen Modus noch einmal skizziert.

**Beispiel:** Kommen wir zu dem Beispiel des kurzen Artikels zurück. Angenommen Sie wollen nun links im Fuß zusätzlich den Verlag angegeben haben. Daher ergänzen Sie das Beispiel von Seite 275 zu:

```
\documentclass{scrartcl}
```

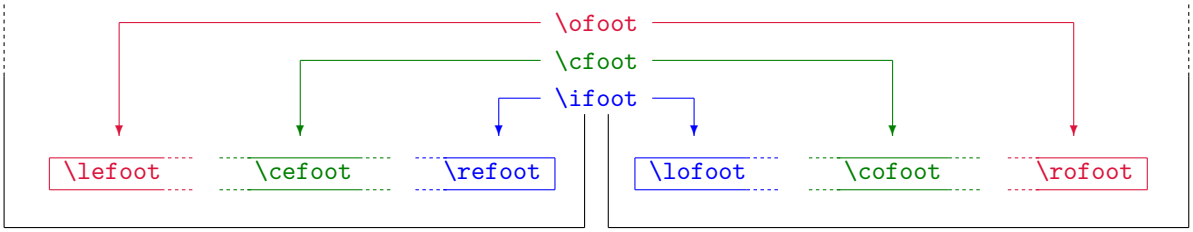



Abbildung 5.2.: Die Bedeutung der Befehle zum Setzen der Inhalte des Fußes eines Seitenstils für die Seitenfüße einer schematischen Doppelseite.

```
\usepackage{scrlayer-scrpage}
\lohead{Peter Musterheinzel}
\rohead{Seitenstile mit \KOMAScript}
\lofoot{Verlag Naseblau, Irgendwo}
\usepackage{lipsum}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\lipsum
\end{document}
```

Der Verlag wird dabei aber nicht auf der Seite mit dem Titelkopf ausgegeben. Die Begründung ist dieselbe wie beim Beispiel zu `\lohead`. Ebenso ist die Lösung, um den Verlag auch auf diese Seite zu bekommen, entsprechend:

```
\lofoot[Verlag Naseblau, Irgendwo]
       {Verlag Naseblau, Irgendwo}
```

Jetzt soll statt der schrägen Schrift in Kopf und Fuß eine aufrechte, aber kleinere Schrift verwendet werden:

```
\setkomafont{pageheadfoot}{\small}
```

Darüber hinaus soll nur der Kopf fett gesetzt werden:

```
\setkomafont{pagehead}{\bfseries}
```

Bei dieser Anweisung ist wichtig, dass sie nach dem Laden von scrpage-scrlayer erfolgt, weil davor `pagehead` und `pageheadfoot` dasselbe Element bezeichnen. Erst durch Laden von scrpage-scrlayer werden daraus zwei unabhängige Elemente.

Ergänzen Sie nun das Beispiel einmal durch ein weiteres `\lipsum` und fügen Sie gleichzeitig Option `twoside` beim Laden von scrartcl hinzu. Zum einen wandert die Seitenzahl im Fuß nun von der Mitte nach außen. Das liegt an der geänderten Voreinstellung für `scrheadings` und `plain.scrheadings` für doppelseitige Dokumente mit einer KOMA-Script-Klasse.

Gleichzeitig verschwinden aber auch Autor, Dokumenttitel und Verlag von Seite 2. Diese finden sich erst auf Seite 3 wieder. Das liegt daran, dass wir bisher nur Befehle für ungerade Seiten verwendet haben. Zu erkennen ist das am »`o`« für *odd* an der zweiten Stelle der Befehlsnamen.

Nun könnten wir die Befehle einfach kopieren und in der Kopie dieses »`o`« durch ein »`e`« für *even* ersetzen. Allerdings ist es bei doppelseitigen Dokumenten meist sinnvoller, wenn die Elemente spiegelverkehrt verwendet werden, dass also Elemente, die auf linken Seiten links stehen, auf rechten Seiten rechts platziert werden und umgekehrt. Daher vertauschen wir auch noch beim ersten Buchstaben »`l`« mit »`r`« und umgekehrt:

```
\documentclass[twoside]{scrartcl}
\usepackage{scrlayer-scrpage}
\lohead{Peter Musterheinzel}
\rohead{Seitenstile mit \KOMAScript}
\lofoot[Verlag Naseblau, Irgendwo]
       {Verlag Naseblau, Irgendwo}
\rehead{Peter Musterheinzel}
\lehead{Seitenstile mit \KOMAScript}
\refoot[Verlag Naseblau, Irgendwo]
       {Verlag Naseblau, Irgendwo}
\setkomafont{pageheadfoot}{\small}
\setkomafont{pagehead}{\bfseries}
\usepackage{lipsum}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\lipsum\lipsum
\end{document}
```

Da es etwas umständlich ist, die Angaben bei doppelseitigen Dokumenten wie im letzten Beispiel immer getrennt für linke und rechte Seiten zu machen, wird später noch eine schönere Lösung für diesen Standardfall eingeführt.

Sie sollten niemals die Überschrift oder die Nummer einer Gliederungsebene mit Hilfe einer dieser Anweisungen als Kolumnentitel in den Fuß der Seite setzen. Aufgrund der Asynchronizität von Seitenaufbau und Seitenausgabe kann sonst die falsche Nummer oder die falsche Überschrift im Kolumnentitel ausgegeben werden. Stattdessen ist der Mark-Mechanismus, idealer Weise in Verbindung mit den Automatismen aus dem nächsten Abschnitt, zu verwenden.

```
\lefoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cefoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\refoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\lofoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cofoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\rofoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
```

v3.14 Die Sternvarianten der zuvor erklärten Befehle unterscheiden sich von der Form ohne Stern lediglich bei Weglassen des optionalen Arguments `[Inhalt plain.scrheadings]`. Während die Form ohne Stern in diesem Fall den Inhalt von `plain.scrheadings` unangetastet lässt, wird bei der Sternvariante dann das obligatorische Argument *Inhalt scrheadings* auch für `plain.scrheadings` verwendet. Sollen also beide Argumente gleich sein, so kann man einfach die Sternvariante mit nur einem Argument verwenden.

**Beispiel:** Mit der Sternform von \lofoot und \rofoot lässt sich das Beispiel aus der vorherigen Erklärung etwas verkürzen:

```
\documentclass[twoside]{scrartcl}
\usepackage{scrlayer-scrpage}
\lohead{Peter Musterheinzel}
\rohead{Seitenstile mit \KOMAScript}
\lofoot*{Verlag Naseblau, Irgendwo}
\rehead{Peter Musterheinzel}
\lehead{Seitenstile mit \KOMAScript}
\refoot*{Verlag Naseblau, Irgendwo}
\setkomafont{pageheadfoot}{\small}
\setkomafont{pagehead}{\bfseries}
\usepackage{lipsum}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\lipsum\lipsum
\end{document}
```

```
\ohead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\chead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\ihead[Inhalt plain.scrheadings]{Inhalt scrheadings}
\ofoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cfoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
\ifoot[Inhalt plain.scrheadings]{Inhalt scrheadings}
```

Um Kopf und Fuß im doppelseitigen Layout zu konfigurieren, müsste man mit den zuvor erklärten Befehlen die linken und die rechten Seiten getrennt voneinander konfigurieren. Meist ist es jedoch so, dass linke und rechte Seite mehr oder weniger symmetrisch zueinander sind. Ein Element das auf linken Seiten links steht, steht auf rechten Seiten rechts und umgekehrt. Mittig bleibt mittig.

Zur Vereinfachung der Einstellungen für diesen Standardfall gibt es bei scrlayer-scrpage sozusagen Abkürzungen. Der Befehl \ohead entspricht einem Aufruf sowohl von \lehead als auch \rohead. Der Befehl \chead entspricht sowohl \cehead als auch \cohead. Und Befehl \ihead entspricht \rehead und \lohead. Gleiches gilt für die Anweisungen des Fußes. In Abbildung 5.1 auf Seite 275 und Abbildung 5.2 auf Seite 278 werden auch diese Beziehungen skizziert.

**Beispiel:** Mit Hilfe der neuen Befehle, lässt sich das letzte Beispiel wie folgt vereinfachen:

```
\documentclass[twoside]{scrartcl}
\usepackage{scrlayer-scrpage}
\ihead{Peter Musterheinzel}
```

```
\ohead{Seitenstile mit \KOMAScript}
\ifoot[Verlag Naseblau, Irgendwo]
      {Verlag Naseblau, Irgendwo}
\setkomafont{pageheadfoot}{\small}
\setkomafont{pagehead}{\bfseries}
\usepackage{lipsum}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\lipsum\lipsum
\end{document}
```

Da im einseitigen Layout alle Seiten als ungerade oder rechte Seiten behandelt werden, können im einseitigen Layout diese Befehle auch als Synonyme für die entsprechenden Befehle für rechte Seiten verwendet werden.

Erlauben Sie mir noch einmal einen wichtigen Hinweis: Sie sollten niemals die Überschrift oder die Nummer einer Gliederungsebene mit Hilfe einer dieser Anweisungen als Kolumnentitel in den Kopf oder Fuß der Seite setzen. Aufgrund der Asynchronizität von Seitenaufbau und Seitenausgabe kann sonst die falsche Nummer oder die falsche Überschrift im Kolumnentitel ausgegeben werden. Stattdessen ist der Mark-Mechanismus, idealer Weise in Verbindung mit den Automatismen aus dem nächsten Abschnitt, zu verwenden.

```
\ohead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\chead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\ihead*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\ofoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\cfoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
\ifoot*[Inhalt plain.scrheadings]{Inhalt scrheadings}
```

v3.14

Die Sternvarianten der zuvor erklärten Befehle unterscheiden sich von der Form ohne Stern lediglich bei Weglassen des optionalen Arguments `[Inhalt plain.scrheadings]`. Während die Form ohne Stern in diesem Fall den Inhalt von `plain.scrheadings` unangetastet lässt, wird bei der Sternvariante dann das obligatorische Argument *Inhalt scrheadings* auch für `plain.scrheadings` verwendet. Sollen also beide Argumente gleich sein, so kann man einfach die Sternvariante mit nur einem Argument verwenden.

**Beispiel:** Mit der Sternform von `\ifoot` lässt sich das Beispiel aus der vorherigen Erklärung weiter verkürzen:

```
\documentclass[twoside]{scrartcl}
\usepackage{scrlayer-scrpage}
\ihead{Peter Musterheinzel}
\ohead{Seitenstile mit \KOMAScript}
```

```
\ifoot*{Verlag Naseblau, Irgendwo}
\setkomafont{pageheadfoot}{\small}
\setkomafont{pagehead}{\bfseries}
\usepackage{lipsum}
\begin{document}
\title{Seitenstile mit \KOMAScript}
\author{Peter Musterheinzel}
\maketitle
\lipsum\lipsum
\end{document}
```

pagestyleset=*Einstellung*

In den zurückliegenden Beispielen wurde bereits mehrfach auf die Voreinstellung der Seitenstile scrheadings und plain.scrheadings hingewiesen. Tatsächlich unterstützt scrlayer-scrpage derzeit zwei unterschiedliche Voreinstellungen. Diese sind von Hand über Option pagestyleset auswählbar.

Mit der *Einstellung* KOMA-Script wird die Voreinstellung gewählt, die auch automatisch eingestellt wird, falls die Option nicht angegeben ist und eine KOMA-Script-Klasse erkannt wurde. Dabei werden für scrheadings im doppelseitigen Satz Kolumnentitel außen im Kopf und Seitenzahlen außen im Fuß gesetzt. Im einseitigen Satz wird der Kolumnentitel stattdessen mittig im Kopf und die Seitenzahl mittig im Fuß platziert. Für die automatischen Kolumnentitel werden Groß- und Kleinbuchstaben wie vorgefunden verwendet. Dies entspricht Option markcase=used. Für plain.scrheadings entfallen die Kolumnentitel. Die Seitenzahlen werden jedoch identisch platziert.

Wird allerdings scrlttr2 als Klasse erkannt, so werden die Voreinstellungen an die Seitenstile jener Klasse angelehnt. Siehe dazu Abschnitt 4.13, ab Seite 240.

Mit der *Einstellung* standard wird die Voreinstellung gewählt, die den Standard-Klassen entspricht. Diese wird auch automatisch eingestellt, falls die Option nicht angegeben ist und keine KOMA-Script-Klasse erkannt wurde. Dabei wird für scrheadings im doppelseitigen Satz der Kolumnentitel im Kopf innen und die Seitenzahl – ebenfalls im Kopf – außen ausgerichtet platziert. Im einseitigen Satz werden dieselben Einstellungen verwendet. Da hierbei nur rechte Seiten existieren, steht der Kolumnentitel dann immer linksbündig im Kopf, die Seitenzahl entsprechend rechtsbündig. Die automatischen Kolumnentitel werden – trotz erheblicher typografischer Bedenken – entsprechend markcase=upper in Großbuchstaben umgewandelt. Von scrheadings deutlich abweichend hat plain.scrheadings die Seitenzahl im einseitigen Satz mittig im Fuß. Im Unterschied zum Seitenstil plain der Standardklassen entfällt die Seitenzahl im doppelseitigen Modus. Die Standardklassen setzen die Seitenzahl stattdessen mittig im Fuß, was im doppelseitigen Satz nicht zum übrigen Stil der Seiten passt. Wer die Seitenzahl zurück haben will, kann dies mit

```
\cfoot[\pagemark]{}
```

erreichen. Der Kolumnentitel entfällt bei plain.scrheadings.

Es ist zu beachten, dass mit der Verwendung dieser Option gleichzeitig der Seitenstil scrheadings aktiviert wird. Dies gilt auch, wenn die Option innerhalb eines Dokuments verwendet wird.

## 5.5. Beeinflussung von Seitenstilen

In Abschnitt 5.4 wurde erklärt, wie die Seitenstile scrheadings und plain.scrheadings grundlegend vordefiniert sind und wie diese Vorbelegung grundsätzlich geändert werden kann. Es fehlen jedoch noch Informationen, wie beispielsweise die Kolumnentitel zustande kommen, wie man die Breite des Kopfes und Fußes verändern kann und wie man Linien über oder unter Kopf oder Fuß setzen kann. Obwohl dies eigentlich Fähigkeiten des Pakets scrlayer sind, werden sie nachfolgend erläutert, da diese grundlegenden Eigenschaften von scrlayer einen wichtigen Teil der Möglichkeiten von scrlayer-scrpage ausmachen.

```
\automark[Gliederungsebene der rechten Marke]{Gliederungsebene der linken Marke}
\automark*[Gliederungsebene der rechten Marke]{Gliederungsebene der linken Marke}
\manualmark
```

Bei den Standardklassen und auch bei den KOMA-Script-Klassen fällt die Entscheidung, ob mit lebenden oder statischen Kolumnentiteln gearbeitet werden soll, über die Wahl des entsprechenden Seitenstils. Wie bereits in Abschnitt 3.12 erklärt, erhält man bei Wahl von headings lebende Kolumnentitel. Unter lebenden Kolumnentiteln versteht man die Wiederholung eines für die Seite oder die *Kolumne* markanten Textes meist im Kopf, seltener im Fuß der Seite.

article, scrartcl

Bei den Artikel-Klassen article oder scrartcl wird für den lebenden Kolumnentitel im einseitigen Modus die Abschnittsüberschrift, also das obligatorische oder das optionale Argument von \section verwendet. Diese wird als *rechte Marke* behandelt. Im doppelseitigen Satz wird dieselbe Überschrift als *linke Marke* verwendet und gleichzeitig die Unterabschnittsüberschrift als *rechte Marke*. Ausgegeben wird die linke Marke, wie der Name schon sagt, auf der linken Seite, während die rechte Marke auf rechten Seiten – im einseitigen Modus also auf allen Seiten – ausgegeben wird. Beim Setzen der linken Marke für den Abschnitt werden von den Klassen in der Voreinstellung außerdem auch immer die rechten Marken gelöscht.

report, scrreprt, book, scrbook

Bei den Bericht- und Buch-Klassen wird eine Ebene höher begonnen. Im einseitigen Modus wird also die Kapitelüberschrift als rechte Marke gesetzt. Im doppelseitigen Satz wird die Kapitelüberschrift als linke Marke und die Abschnittsüberschrift als rechte Marke gesetzt.

Verwendet man hingegen als Seitenstil myheadings, so existieren zwar die Marken im Kopf genauso und auch die Seitenzahlen werden gleich platziert, allerdings werden die Marken nicht automatisch durch die Überschriften gesetzt. Man kann sie dann nur manuell über die später in diesem Abschnitt dokumentierten Anweisungen \markright und \markboth befüllen.

Genau diese Unterscheidung wurde bei scrlayer aufgehoben. Statt die Unterscheidung zwischen automatischen und manuellen Kolumnentiteln über den Seitenstil vorzunehmen, gibt es die beiden Anweisungen \automark und \manualmark.

Mit \manualmark wird dabei auf manuelle Marken umgeschaltet. Es deaktiviert also das automatische Setzen der Marken. Demgegenüber kann mit \automark und \automark* festgelegt werden, welche Gliederungsebenen für das automatische Setzen der Marken verwendet werden sollen. Das optionale Argument gibt dabei die *Gliederungsebene der rechten Marke* an, während das obligatorische Argument die *Gliederungsebene der linken Marke* ist. Als Argument werden jeweils die Namen der Gliederungsebenen angegeben, also part, chapter, section, subsection, subsubsection, paragraph oder subparagraph.

Normalerweise sollte die höhere Ebene die linke Marke setzen, während die tiefere Ebene für die rechte Marke zu verwenden ist. Diese übliche Konvention ist jedoch keine Pflicht, sondern lediglich sinnvoll.

Es ist zu beachten, dass nicht alle Klassen Kolumnentitel für alle Ebenen ermöglichen. So setzen die Standardklassen beispielsweise nie Kolumnentitel für \part. Die KOMA-Script-Klassen unterstützen hingegen alle Ebenen.

Der Unterschied zwischen \automark und \automark* liegt darin, dass \automark alle vorherigen Befehle zum automatischen Setzen der Marken aufhebt, während die Stern-Version \automark* lediglich die Aktionen für die angegebenen Gliederungsebenen ändert.

**Beispiel:** Angenommen Sie wollen, dass wie üblich auf den linken Seiten eines Buches die Kapitelüberschriften als automatische Kolumnentitel verwendet werden und auf den rechten Seiten die Abschnittsüberschriften. Allerdings soll auf rechte Seiten so lange ebenfalls die Kapitelüberschrift verwendet werden, bis der erste Abschnitt auftaucht. Dazu wird zuerst scrlayer-scrpage geladen. Dadurch ist auch bereits automatisch der Seitenstil scrheadings aktiviert. Das Dokument beginnt also mit:

```
\documentclass{scrbook}
\usepackage{scrlayer-scrpage}
```

Als nächstes wird dafür gesorgt, dass die Kapitelüberschriften sowohl die linke als auch die rechte Marke setzen:

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

Dann sollen die Abschnittsüberschriften zusätzlich die rechten Marken setzen:

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

Hier findet die Stern-Version Anwendung, da die vorherige \automark-Anweisung weiterhin wirksam bleiben soll. Außerdem bleibt das Argument für die *Gliederungsebene der linken Marke* leer, weil diese Marke unverändert bleiben soll.

Alles, was jetzt noch fehlt, ist ein wenig Dokumentinhalt:

```
\usepackage{lipsum}
\begin{document}
\chapter{Kapitel}
\lipsum[1-20]
```

```
\section{Abschnitt}
\lipsum[21-40]
\end{document}
```

Wenn Sie dieses Beispiel einmal testen, werden Sie sehen, dass die Kapitelanfangsseite wie üblich ohne Kolumnentitel ist, da sie automatisch im `plain`-Seitenstil `plain.scrheadings` gesetzt wird (siehe dazu Anweisung `\chapterpagestyle` auf Seite 87). Seite 2 bis 4 tragen als Kolumnentitel die Kapitelüberschrift. Nachdem auf Seite 4 eine Abschnittsüberschrift ausgegeben wurde, ändert sich der Kolumnentitel auf Seite 5 in die Abschnittsüberschrift. Ab dann werden die beiden Überschriften im Kopf wechselweise ausgegeben, auf linken Seiten die Kapitelüberschrift, auf rechten Seiten die Abschnittsüberschrift.

`automark`
`autooneside=`*`Ein-Aus-Wert`*
`manualmark`

Außer mit den zuvor erklärten Befehlen kann auch direkt mit den beiden Optionen `manualmark` und `automark` zwischen automatischen und manuellen Kolumnentiteln hin- und hergeschaltet werden. Dabei verwendet `automark` bei Klassen mit `\chapter`-Anweisung immer die Voreinstellung `\automark[section]{chapter}` und bei anderen Klassen `\automark[subsection]{section}`.

Im einseitigen Modus will man in der Regel, dass nur die höheren Ebenen den Kolumnentitel vorgeben. Diese Voreinstellung entspricht einer aktiven Option `autooneside`. Die Option versteht die Werte für einfache Schalter, die in Tabelle 2.5 auf Seite 41 angegeben sind. Wird die Option deaktiviert, so wirken sich im einseitigen Satz sowohl das optionale als auch das obligatorische Argument auf den Kolumnentitel aus.

**Beispiel:** Angenommen, Sie wollen im einseitigen Modus eines Berichts so lange die Kapitelüberschriften für den Kolumnentitel verwenden, bis ein Abschnitt gesetzt wird. Ab dann soll nur noch die Abschnittsüberschrift verwendet werden. Dazu wird das vorherige Beispiel wie folgt abgewandelt:

```
\documentclass{scrreprt}
\usepackage[autooneside=false]{scrlayer-scrpage}
\automark[section]{chapter}
\usepackage{lipsum}
\begin{document}
\chapter{Kapitel}
\lipsum[1-20]
\section{Abschnitt}
\lipsum[21-40]
\end{document}
```

In diesem Fall wird keine \automark*-Anweisung benötigt. Sie sollten zum Vergleich die Option autooneside einmal entfernen. Ein Unterschied ist dann ab Seite 4 zu sehen.

Das Laden des Pakets selbst hat übrigens noch keine Auswirkung darauf, ob mit automatischen Kolumnentiteln gearbeitet wird oder nicht. Erst die explizite Verwendung einer der Optionen automark oder manualmark oder einer der beiden Anweisungen \automark oder \manualmark schafft hier klare Verhältnisse.

draft=*Ein-Aus-Wert*

Die KOMA-Script-Option versteht die Standardwerte für einfache Schalter, die in Tabelle 2.5 auf Seite 41 angegeben sind. Ist die Option aktiviert, so werden alle Elemente der Seitenstile zusätzlich mit Maßlinien versehen. Falls diese Option global gesetzt wurde, die Maßlinien aber nicht gewünscht sind, kann die Option auch nur für das Paket deaktiviert werden, indem man draft=false beim Laden des Pakets angibt.

\MakeMarkcase{*Text*}
markcase=*Wert*

Die automatischen, nicht jedoch die manuellen Kolumnentitel verwenden \MakeMarkcase für ihre Ausgabe. Ist die Anweisung beim Laden von scrlayer nicht definiert, so wird sie in der Voreinstellung derart definiert, dass sie ihr Argument *Text* unverändert ausgibt. Diese Voreinstellung kann jedoch durch Umdefinierung von \MakeMarkcase geändert werden. Die Umdefinierung kann auch automatisch durch Verwendung von Option markcase mit einem der Werte aus Tabelle 5.2 erfolgen.

Leider führt die von LaTeX für Versalsatz vorgesehene Anweisung \MakeUppercase zu keinem guten Ergebnis, da weder gesperrt noch ausgeglichen wird. Dies liegt teilweise sicher daran, dass für typografisch korrekten Versalsatz eine Glyphenanalyse notwendig ist, um die konkrete Form der Buchstaben und ihrer Kombinationen in den Ausgleich der Sperrung einfließen zu lassen. Der KOMA-Script-Autor empfiehlt daher, auf Versalsatz für die Kolumnentitel zu verzichten. Dies ist normalerweise mit markcase=used möglich. Allerdings fügen einige Klassen selbst beispielsweise bei den Kolumnentitel für Verzeichnisse ein \MakeUppercase oder sogar die TeX-Anweisung \uppercase ein. Für diese Fälle gibt es auch noch die Einstellung markcase=noupper, mit deren Hilfe \MakeUppercase und \uppercase für die Kolumnentitel lokal deaktiviert werden können.

Tabelle 5.2.: Mögliche Werte für Option `markcase` zur Wahl von Groß-/Kleinschreibung in automatischen Kolumnentiteln

| | |
|---|---|
| `lower` | definiert `\MakeMarkcase` so um, dass automatische Kolumnentitel mit Hilfe von `\MakeLowercase` in Kleinbuchstaben gewandelt werden (Minuskelsatz). |
| `upper` | definiert `\MakeMarkcase` so um, dass automatische Kolumnentitel mit Hilfe von `\MakeUppercase` in Großbuchstaben gewandelt werden (Versalsatz). |
| `title` v3.41 | definiert `\MakeMarkcase` so um, dass automatische Kolumnentitel mit Hilfe von `\MakeTitlecase` gewandelt werden (*Englischer Titelsatz*). Dabei werden Nummer und Titel getrennt behandelt. Ist `\MakeTitlecase` wegen eines zu alten LaTeX Kerns nicht definiert, wird eine Warnung ausgegeben und die Option ignoriert. |
| `used` | definiert `\MakeMarkcase` so um, dass für automatische Kolumnentitel keine automatische Veränderung der Groß-/Kleinschreibung durchgeführt wird. |
| `ignoreuppercase`, `nouppercase`, `ignoreupper`, `noupper` | definiert nicht nur `\MakeMarkcase` so um, dass für automatische Kolumnentitel keine automatische Veränderung der Groß-/Kleinschreibung durchgeführt wird, sondern deaktiviert zusätzlich lokal für alle Ebenen aller Seitenstile `\MakeUppercase` und `\uppercase`. |

```
\leftmark
\rightmark
\headmark
\pagemark
```

Will man von den vordefinierten Seitenstilen abweichen, so muss man in der Regel auch selbst entscheiden können, wo die Marken gesetzt werden sollen. Mit `\leftmark` platziert man die linke Marke. Diese wird dann bei der Ausgabe der Seite durch den entsprechenden Inhalt ersetzt.

Dementsprechend kann man mit `\rightmark` die rechte Marke platzieren, die dann bei der Ausgabe der Seite durch den entsprechenden Inhalt ersetzt wird. Für einige Feinheiten dabei sei auch auf die weiterführenden Erklärungen zu `\rightmark` in Abschnitt 18.1, Seite 481 verwiesen.

Mit `\headmark` kann man sich das Leben erleichtern. Diese Erweiterung von scrlayer entspricht je nachdem, ob die aktuelle Seite eine linke oder rechte ist, `\leftmark` oder `\rightmark`.

Die Anweisung `\pagemark` hat genau genommen nichts mit den Marken von TeX zu tun.

Sie dient dazu, eine formatierte Seitenzahl zu platzieren. Bei ihrer Ausgabe wird dann auch die Schrifteinstellung für das Element `pagenumber` verwendet. Diese kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` verändert werden (siehe auch Abschnitt 3.6, Seite 60).

**Beispiel:** Angenommen, Sie wollen, dass auch im einseitigen Modus der Kolumnentitel immer am linken Rand und die Seitenzahl immer am rechten Rand ausgerichtet wird. Beide sollen im Kopf platziert werden. Das folgende, vollständige Minimalbeispiel liefert genau dies:

```
\documentclass{scrreprt}
\usepackage{blindtext}
\usepackage[automark]{scrlayer-scrpage}
\ihead{\headmark}
\ohead*{\pagemark}
\chead{}
\cfoot[]{}
\begin{document}
\blinddocument
\end{document}
```

Das Paket blindtext mit seiner Anweisung `\blinddocument` wird hier für die komfortable Erzeugung eines Beispieldokumentinhalts verwendet.

Mit den Anweisungen `\ihead` und `\ohead*` werden die gewünschten Marken platziert. Dabei wird die Seitenzahl durch die Sternform `\ohead*` auch für den auf Kapitelanfangsseiten verwendeten Seitenstil `plain.scrheadings` konfiguriert.

Da die Seitenstile bereits mit Marken in der Mitte von Kopf oder Fuß vordefiniert sind, werden diese beiden Elemente mit `\chead` und `\cfoot` gelöscht. Hierzu werden leere Argumente verwendet. Alternativ dazu hätte man auch `\clearpairofpagestyles` *vor* `\ihead` verwenden können. Diese Anweisung wird jedoch erst in Abschnitt 15.2 auf Seite 456 erklärt werden.

Bitte beachten Sie, dass das leere optionale Argument bei `\cfoot` im Beispiel nicht gleichbedeutend mit dem Weglassen dieses optionalen Arguments ist. Sie sollten das einmal selbst ausprobieren und dabei den Fuß der ersten Seite beobachten.

Fortgeschrittene Anwender finden ab Seite 451 weitere Marken-Anweisungen.

```
\partmarkformat
\chaptermarkformat
\sectionmarkformat
\subsectionmarkformat
\subsubsectionmarkformat
\paragraphmarkformat
\subparagraphmarkformat
```

Diese Anweisungen werden von den KOMA-Script-Klassen und auch von scrlayer intern üblicherweise verwendet, um die Gliederungsnummern der automatischen Kolumnentitel zu formatieren. Dabei wird auch der \autodot-Mechanismus der KOMA-Script-Klassen unterstützt. Bei Bedarf können diese Anweisungen umdefiniert werden, um eine andere Formatierung der Nummern zu erreichen.

**Beispiel:** Angenommen, Sie wollen, dass Abschnittsüberschriften im Kolumnentitel ohne Gliederungsnummer gesetzt werden, so ist das ganz einfach mit

```
\renewcommand*{\sectionmarkformat}{}
```

zu erreichen.

```
\partmark{Text}
\chaptermark{Text}
\sectionmark{Text}
\subsectionmark{Text}
\subsubsectionmark{Text}
\paragraphmark{Text}
\subparagraphmark{Text}
```

Diese Anweisungen werden intern von den meisten Klassen verwendet, um die Marken entsprechend der Gliederungsbefehle zu setzen. Dabei wird als Argument lediglich der Text, nicht jedoch die Nummer erwartet. Die Nummer wird stattdessen automatisch über den aktuellen Zählerstand ermittelt, falls mit nummerierten Überschriften gearbeitet wird.

Allerdings verwenden nicht alle Klassen in allen Gliederungsebenen eine solche Anweisung. Beispielsweise rufen die Standardklassen \partmark bei \part nicht auf.

Falls diese Anweisungen vom Anwender umdefiniert werden, sollte er unbedingt darauf achten, vor dem Setzen der Nummer ebenfalls über secnumdepth zu prüfen, ob die Nummern auszugeben sind. Dies gilt auch, wenn der Anwender secnumdepth selbst nicht verändert, weil Pakete und Klassen sich eventuell auf die Wirkung von secnumdepth verlassen!

Das Paket scrlayer definiert diese Anweisungen außerdem bei jedem Aufruf von \automark oder \manualmark oder den entsprechenden Optionen teilweise neu, um so die gewünschten automatischen oder manuellen Kolumnentitel zu erreichen.

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

Unabhängig davon, ob gerade mit manuellen oder automatischen Kolumnentiteln gearbeitet wird, kann man jederzeit die *linke Marke* oder *rechte Marke* mit einer dieser Anweisungen setzen. Dabei ist zu beachten, dass die resultierende linke Marke in `\leftmark` die letzte auf der entsprechenden Seite gesetzte Marke ist, während die resultierende rechte Marke in `\rightmark` die erste auf der entsprechenden Seite gesetzte Marke ausgibt. Näheres dazu ist den weiterführenden Erklärungen zu `\rightmark` in Abschnitt 18.1, Seite 481 oder zu `\rightfirstmark` in Abschnitt 14.6, Seite 446 zu entnehmen.

Wird mit manuellen Kolumnentiteln gearbeitet, so bleiben die Marken gültig, bis sie durch erneute Verwendung der entsprechenden Anweisung explizit ersetzt werden. Bei automatischen Kolumnentiteln können Marken hingegen je nach Konfigurierung des Automatismus ihre Gültigkeit mit einer der nächsten Gliederungsüberschriften verlieren.

Auch im Zusammenhang mit den Sternvarianten der Gliederungsbefehle können diese Anweisungen nützlich sein.

**Beispiel:** Angenommen, Sie schreiben noch vor dem Inhaltsverzeichnis ein Vorwort über mehrere Seiten, das jedoch im Inhaltsverzeichnis nicht auftauchen soll. Da Sie aber Trennlinien im Kopf verwenden, soll der Kolumnentitel das Vorwort dennoch zeigen:

```
\documentclass[headsepline]{book}
\usepackage[automark]{scrlayer-scrpage}
\usepackage{blindtext}
\begin{document}
\chapter*{Vorwort}
\markboth{Vorwort}{Vorwort}
\blindtext[20]
\tableofcontents
\blinddocument
\end{document}
```

Zunächst erscheint das Ergebnis wunschgemäß. Vielleicht erst beim zweiten Blick fällt aber auf, dass der Kolumnentitel »`Vorwort`« im Gegensatz zu den übrigen Kolumnentiteln nicht im Versalsatz erscheint. Das ist jedoch leicht zu ändern:

```
\documentclass[headsepline]{book}
\usepackage[automark]{scrlayer-scrpage}
\usepackage{blindtext}
\begin{document}
\chapter*{Vorwort}
\markboth{\MakeMarkcase{Vorwort}}%
```

```
        {\MakeMarkcase{Vorwort}}
\blindtext[20]
\tableofcontents
\blinddocument
\end{document}
```

Wie zu sehen ist, wurde \MakeMarkcase verwendet, um auch den manuell korrigierten Kolumnentitel des Vorworts entsprechend der automatischen Kolumnentitel des restlichen Dokuments anzupassen.

Verschieben Sie nun einmal \tableofcontents vor das Vorwort und entfernen Sie die \markboth-Anweisung. Sie werden entdecken, dass das Vorwort als Kolumnentitel nun »CONTENTS« trägt. Das liegt an einer Eigenart von \chapter* (siehe auch in Abschnitt 3.16 auf Seite 112). Soll hier stattdessen kein Kolumnentitel erscheinen, so ist dies sehr einfach mit \markboth mit zwei leeren Argumenten zu erreichen:

```
\documentclass[headsepline]{book}
\usepackage[automark]{scrlayer-scrpage}
\usepackage{blindtext}
\begin{document}
\tableofcontents
\chapter*{Vorwort}
\markboth{}{}
\blindtext[20]
\blinddocument
\end{document}
```

v3.28 Die Anweisung \markdouble setzt sowohl die linke als auch rechte Marke auf denselben Inhalt. Damit ist \markdouble{*Marke*} eine abkürzende Schreibweise für \markboth{*Marke*}{*Marke*} mit zwei identischen Argumenten.

headwidth=*Breite*:*Offset*:*Offset*
footwidth=*Breite*:*Offset*:*Offset*

In der Voreinstellung sind Kopf und Fuß genauso breit wie der Satzspiegel. Mit Hilfe dieser beiden KOMA-Script-Optionen lässt sich das jedoch ändern. Die *Breite* ist dabei die gewünschte Breite des Kopfes beziehungsweise Fußes. Der *Offset* gibt an, wie weit der Kopf respektive Fuß in Richtung des äußeren Randes – im einseitigen Satz entsprechend in Richtung
v3.14 des rechten Randes – verschoben werden soll. Dabei sind alle drei Werte optional, können also auch weggelassen werden. Falls ein Wert weggelassen wird, kann auch ein zugehöriger Doppelpunkt links davon entfallen. Ist nur ein *Offset* angegeben, so wird dieser sowohl für linke als auch für rechte Seiten verwendet. Ansonsten wird im doppelseitigen Satz der erste *Offset* für ungerade, also rechte Seiten und der zweite für gerade, also linke Seiten verwendet. Ist insgesamt nur ein Wert und kein Doppelpunkt angegeben, so handelt es sich um die *Breite*.

Sowohl für *Breite* als auch für *Offset* kann jeder gültige Längenwert aber auch jede LaTeX-Länge oder TeX-Länge oder TeX-Abstand eingesetzt werden. Darüber hinaus sind für beides auch ε-TeX-Längenausdrücke mit den Grundrechenarten +, -, *, / und runden Klammern erlaubt. Näheres zu solchen Längenausdrücken ist [Tea98, Abschnitt 3.5] zu entnehmen. Für *Breite* sind außerdem einige symbolische Werte zulässig. Diese sind Tabelle 5.3 zu entnehmen.

Die Voreinstellung für *Breite* ist die Breite des Textbereichs. Die Voreinstellung für *Offset* hängt von der gewählten *Breite* ab. In der Regel wird im einseitigen Satz die Hälfte des Unterschieds zwischen *Breite* und der Breite des Textbereichs verwendet. Damit wird der Kopf über dem Textbereich zentriert. Im doppelseitigen Satz wird hingegen nur ein Drittel des Unterschieds zwischen *Breite* und der Breite des Textbereichs verwendet. Ist *Breite* jedoch die Breite des Textbereichs zuzüglich der Marginalienspalte, so ist die Voreinstellung von *Offset* immer Null. Falls Ihnen das zu kompliziert ist, sollten Sie den gewünschten *Offset* einfach selbst angeben.

```
headtopline=Dicke:Länge
headsepline=Dicke:Länge
footsepline=Dicke:Länge
footbotline=Dicke:Länge
```

Während die KOMA-Script-Klassen nur eine Trennlinie unter dem Kopf und eine weitere über dem Fuß unterstützen und man diese nur wahlweise ein- und ausschalten kann, erlaubt das Paket scrlayer-scrpage auch noch eine Linie über dem Kopf und unter dem Fuß und man kann sowohl die *Länge* als auch die *Dicke* einstellen.

Beide Werte sind optional. Lässt man die *Dicke* weg, so wird 0,4 pt angenommen, also eine Haarlinie produziert. Verzichtet man auf eine Angabe der *Länge*, so wird die Breite des Kopfes respektive Fußes als gewünschter Wert angenommen. Wird beides weggelassen, so kann auch der Doppelpunkt entfallen. Wird nur ein Wert ohne Doppelpunkt angegeben, so ist dies die *Dicke*.

Natürlich darf die *Länge* nicht nur kürzer als die aktuelle Breite des Kopfes respektive Fußes sein. Sie darf auch länger sein. Siehe dazu auch die Optionen ilines, clines und olines, die später in diesem Abschnitt erklärt werden.

Neben der Dicke und der Länge kann man auch die Farben der Linien ändern. Zunächst richtet sich diese natürlich nach der Farbe, die für den Kopf und den Fuß eingestellt ist. Davon unabhängig werden aber auch noch die Einstellungen für die gleich benannten Elemente headtopline, headsepline, footsepline und footbotline angewendet. Diese können mit den Anweisungen \setkomafont und \addtokomafont geändert werden (siehe Abschnitt 5.3, ab Seite 270). In der Voreinstellung sind die Einstellungen für diese Elemente leer, so dass sie zu keiner Änderung der Schrift oder Farbe führen. Änderungen der Schrift sind im Gegensatz zu Farbänderungen an dieser Stelle ohnehin nicht sinnvoll und werden daher nicht empfohlen.

Tabelle 5.3.: Erlaubte symbolische Werte für *Breite* bei den Optionen `headwidth` und `footwidth`

`foot`
: die aktuelle Breite des Fußes

`footbotline`
: die aktuelle Länge der horizontalen Linie unterhalb des Fußes

`footsepline`
: die aktuelle Länge der horizontalen Linie zwischen dem Textbereich und dem Fuß

`head`
: die aktuelle Breite des Kopfes

`headsepline`
: die aktuelle Länge der horizontalen Linie zwischen dem Kopf und dem Textbereich

`headtopline`
: die aktuelle Länge der horizontalen Linie über dem Kopf

`marginpar`
: die Breite der Marginalienspalte einschließlich des Abstandes zwischen dem Textbereich und der Marginalienspalte

`page`
: die Breite der Seite unter Berücksichtigung einer eventuell mit Hilfe des Pakets typearea definierten Bindekorrektur (siehe Option BCOR in Abschnitt 2.6, Seite 33)

`paper`
: die Breite des Papiers ohne Berücksichtigung einer etwaigen Bindekorrektur

`text`
: die Breite des Textbereichs

`textwithmarginpar`
: die Breite des Textbereichs einschließlich der Marginalienspalte und natürlich des Abstandes zwischen den beiden (Achtung: Nur in diesem Fall ist die Voreinstellung für *Offset* Null)

```
plainheadtopline=Ein-Aus-Wert
plainheadsepline=Ein-Aus-Wert
plainfootsepline=Ein-Aus-Wert
plainfootbotline=Ein-Aus-Wert
```

Mit diesen Optionen können die Einstellungen für die Linien auch für den `plain`-Seitenstil übernommen werden. Als *Ein-Aus-Wert* stehen die Standardwerte für einfache Schalter, die in Tabelle 2.5 auf Seite 41 angegeben sind, zur Verfügung. Bei aktivierter Option werden die entsprechenden Linieneinstellungen übernommen. Bei deaktivierter Option wird die entsprechende Linie im `plain`-Seitenstil hingegen nicht angezeigt.

```
ilines
clines
olines
```

Wie bereits zuvor erklärt wurde, können Trennlinien für den Kopf oder Fuß konfiguriert werden, die länger oder kürzer als die Breite des Kopfes beziehungsweise des Fußes sind. Bisher blieb die Frage offen, wie diese Linien dann ausgerichtet werden. In der Voreinstellung sind sie im einseitigen Satz linksbündig und im doppelseitigen Satz bündig mit dem Anfang des inneren Randes. Dies entspricht Option `ilines`. Alternativ können sie jedoch mit Option `clines` auch horizontal bezüglich der Breite des Kopfes beziehungsweise Fußes zentriert werden. Ebenso ist mit Hilfe von Option `olines` eine Ausrichtung am äußeren beziehungsweise rechten Rand möglich.

# Adressdateien mit scraddr erschließen

Das Paket scraddr ist eine kleine Beigabe zur Briefklasse und zum Briefpaket von KOMA-Script. Ziel ist, die Benutzung von Adressdateien zu vereinfachen und ihre Anwendung flexibler zu gestalten.

## 6.1. Befehle

Im Grunde stellt das Paket nur einen Lademechanismus für Adressdateien bereit, die aus \adrentry- und \adrchar- oder den neueren \addrentry- und \addrchar-Einträgen bestehen, wie sie in Kapitel 4 ab Seite 262 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 4.21 ab Seite 262 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.

```
\Name{Kürzel}
\FirstName{Kürzel}
\LastName{Kürzel}
\Address{Kürzel}
\Telephone{Kürzel}
\FreeI{Kürzel}
\FreeII{Kürzel}
\Comment{Kürzel}
\FreeIII{Kürzel}
\FreeIV{Kürzel}
```

Die englischen Namen der Zugriffsbefehle folgen den Bezeichnungen der Argumente von \adrentry und \addrentry. Die Auswahl des Adresseintrags 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.

## 6.2. Anwendung

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, da es in Auenland keine Telefone gibt. Dagegen wäre es nicht erlaubt, ein Argument einfach komplett weg zu lassen.

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, wie bei \begin{letter} gezeigt, 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 findet sich im nächsten Abschnitt.

## 6.3. 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. Bitte beachten Sie, dass dieses Paket die erweiterte Optionenschnittstelle mit `\KOMAoptions` und `\KOMAoption` nicht unterstützt. Die Optionen sind also entweder als globale Optionen bei `\documentclass` oder als lokale Optionen bei `\usepackage` anzugeben.

```
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.

Wird für das Paket keine Option angegeben, so ist `adrFreeIVshow` voreingestellt.

# 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 für Briefe. Diese grundlegenden Möglichkeiten werden von KOMA-Script teilweise auch durch Paket scrextend angeboten. Dieses Paket sollte nicht mit KOMA-Script-Klassen verwendet werden. Es ist ausschließlich zur Verwendung mit anderen Klassen gedacht. Der Versuch, das Paket mit einer KOMA-Script-Klasse zu laden, wird von scrextend erkannt und mit einer Warnung abgebrochen.

Dass scrletter nicht nur mit KOMA-Script-Klassen, sondern auch mit den Standardklassen verwendet werden kann, liegt übrigens teilweise an scrextend. Stellt scrletter nämlich fest, dass es nicht mit einer KOMA-Script-Klasse verwendet wird, so lädt es automatisch scrextend. Damit stehen dann alle von scrletter aktiv genutzten Möglichkeiten der KOMA-Script-Klassen zur Verfügung.

Es gibt natürlich keine Garantie, dass scrextend mit jeder beliebigen Klasse zusammenarbeitet. Das Paket ist primär für die Erweiterung der Standardklassen und davon abgeleiteter Klassen gedacht. In jedem Fall sollten Benutzer zunächst prüfen, ob die verwendete Klasse nicht selbst entsprechende Möglichkeiten bereitstellt.

Einige grundlegende Möglichkeiten, die hauptsächlich für Klassen- und Paketautoren interessant sind, werden bei KOMA-Script von scrbase bereitgestellt. Das Paket wird von allen KOMA-Script-Klassen und den meisten KOMA-Script-Paketen geladen. Dadurch sind dessen in Kapitel 10, ab Seite 323 dokumentierte Möglichkeiten bei Verwendung von scrextend ebenfalls verfügbar.

Auch das Paket scrlfile aus Kapitel 11 ab Seite 354 wird von allen KOMA-Script-Klassen und dem Paket scrextend geladen. Somit 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 366) 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 spricht nichts dagegen, tocbasic zusätzlich zu scrextend zu laden.

## 7.1. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde. Falls Sie also Abschnitt 2.4 bereits gelesen und verstanden haben, können Sie auf Seite 302 mit Abschnitt 7.2 fortfahren.

```
\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 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 [Tea23] oder jeder LaTeX-Einführung, beispielsweise [DGS+12], beschrieben.

v3.00

Gegenüber der nachfolgend vorgestellten Schnittstelle zu Einstellungen von Optionen hat `\documentclass` einen Nachteil, der unbedingt zu beachten ist: Anweisungen, Längen, Zähler und ähnliches können darin leicht zerbrechen. So führt die Verwendung einer LaTeX-Länge im Wert einer Option bei dieser Anweisung bei vielen Nicht-KOMA-Script-Klassen zu einer Fehlermeldung, noch bevor der Wert an ein KOMA-Script-Paket übergeben wird, es also die Kontrolle darüber übernehmen könnte. Wertzuweisungen mit LaTeX-Längen oder LaTeX-Zählern sollten daher nie per `\documentclass`, sondern mit den nachfolgend dokumentierten Anweisungen `\KOMAoptions` oder `\KOMAoption` vorgenommen werden.

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

v3.00

KOMA-Script bietet bei den meisten Paketoptionen auch die Möglichkeit, den Wert der Optionen noch nach dem Laden des Pakets zu ändern. Mit der Anweisung `\KOMAoptions` kann man wie bei `\documentclass` oder `\usepackage` 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 `\KOMAoption` der einen *Option* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *Werteliste* durch Komma voneinander getrennt.

Soll ein *Wert* ein Gleichheitszeichen oder ein Komma enthalten, 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`«. Siehe dazu Teil II, Abschnitt 10.2, ab Seite 329.

Mit `\KOMAoptions` oder `\KOMAoption` gesetzte Optionen erreichen auch alle bereits geladenen KOMA-Script-Pakete, die diese Optionen kennen. Ist eine Option oder ein Wert insgesamt unbekannt, so wird die Option einschließlich des Wertes von scrbase als fehlerhaft gemeldet.

## 7.2. Kompatibilität zu früheren Versionen von KOMA-Script

Es gilt sinngemäß, was in Abschnitt 2.5 geschrieben wurde. Falls Sie also Abschnitt 2.5 bereits gelesen und verstanden haben, können Sie in Abschnitt 7.3 auf Seite 302 fortfahren.

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

v3.06a Bei scrextend besteht die Wahl, ob eine Quelldatei, soweit irgend möglich, auch zukünftig bei einem LaTeX-Lauf zu exakt demselben Ergebnis führen soll oder ob sie 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 rückwirkende Kompatibilität verzichtet. Wird die Option ohne Wertangabe verwendet, v3.01a so wird ebenfalls `last` angenommen. Dies entspricht auch der Voreinstellung.

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 scrextend nicht mehr verändert werden kann. Das Setzen mit `\KOMAoptions` oder `\KOMAoption` ist für diese Option daher nicht vorgesehen.

## 7.3. 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 können. Diese sind beispielsweise deshalb optional, weil sie potentiell in Konflikt mit den Möglichkeiten der Standardklassen oder häufig benutzter Pakete stehen.

Tabelle 7.1.: 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 7.7, ab Seite 306) |

`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 7.1.

## 7.4. Entwurfsmodus

Es gilt sinngemäß, was in Abschnitt 3.3 geschrieben wurde. Falls Sie also Abschnitt 3.3 bereits gelesen und verstanden haben, können Sie nach dem Ende dieses Abschnitts auf Seite 303 mit Abschnitt 7.5 fortfahren.

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

Mit Option `draft` 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 41 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 [Sch13].

v3.25 Da Option `draft` bei verschiedenen Paketen zu allerlei unerwünschten Effekten führen kann, bietet KOMA-Script die Möglichkeit, die Markierung für überlange Zeilen auch über Option `overfullrule` zu steuern. Auch hier gilt, dass bei aktivierter Option die Markierung angezeigt wird.

## 7.5. Wahl der Schriftgröße für das Dokument

Es gilt sinngemäß, was in Abschnitt 3.5 geschrieben wurde. Falls Sie also Abschnitt 3.5 bereits gelesen und verstanden haben, können Sie direkt zu Abschnitt 7.6 auf Seite 304 springen.

`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 `\normalsize` und die davon abhängigen Schriftgrößen der Befehle `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\large`, `\Large`, `\LARGE`, `\huge` und `\Huge` 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 einem eventuellen Laden von typearea die Aufteilung zwischen Satzspiegel und Rändern nicht automatisch neu berechnet wird (siehe `\recalctypearea`, Abschnitt 2.6, Seite 40). Wird diese Neuberechnung jedoch vorgenommen, so erfolgt sie auf Basis der jeweils gültigen Grundschriftgröße. Die Auswirkungen des Wechsels der Grundschriftgröße auf zusätzlich geladene Pakete oder die verwendete Klasse sind von diesen Paketen und der Klasse abhängig. Es können also Fehler auftreten, die nicht als Fehler von KOMA-Script angesehen werden.

Diese Option sollte keinesfalls als Ersatz für `\fontsize` (siehe [Tea05]) missverstanden werden. Sie sollte auch nicht anstelle einer der von der Grundschrift abhängigen Schriftgrößenanweisungen, `\tiny` bis `\Huge`, verwendet werden! Die Verwendung innerhalb eines Absatzes ist aus diesem Grund auch explizit verboten!

## 7.6. Textauszeichnungen

Es gilt sinngemäß, was in Abschnitt 3.6 geschrieben wurde. Falls Sie also Abschnitt 3.6 bereits gelesen und verstanden haben, können Sie auf Seite 306 mit Abschnitt 7.7 fortfahren. Es sei jedoch darauf hingewiesen dass aus Tabelle 3.2, Seite 61 nur die Elemente für den Dokumenttitel, den schlauen Spruch, die Fußnoten und die `labeling`-Umgebung unterstützt werden. Das Element `disposition` ist zwar auch vorhanden, wird jedoch nur für den Dokumenttitel verwendet.

`sfdefaults=`*Ein-Aus-Wert*
`\maybesffamily`
`\textmaybesf{`*Text*`}`

v3.39

Die Anweisungen `\maybesffamily` und `\textmaybesf` verhalten sich je nach Einstellung von Option `sfdefaults` unterschiedlich. Als *Ein-Aus-Wert* kann dabei einer der Standardwerte für einfache Schalter aus Tabelle 2.5 verwendet werden. Nur bei aktivierter Option führt `\maybesffamily` zu `\sffamily` und `\textmaybesf` zu `\textsf`. Dies ist auch die Voreinstellung. KOMA-Script selbst verwendet `\maybesffamily` in der Voreinstellung der Elemente

disposition und dictum. \maybesffamily darf also auch als Teil der *Befehle* der nachfolgend erklärten Anweisungen \setkomafont und \addtokomafont verwendet werden.

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

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 Schriftattribute umgeschaltet werden. In der Regel werden dies Befehle wie \rmfamily, \sffamily, \ttfamily, \upshape, \itshape, \slshape, \scshape, \mdseries, \bfseries, \normalfont oder einer der Befehle \Huge, \huge, \LARGE, \Large, \large, \normalsize, \small, \footnotesize, \scriptsize und \tiny sein. Die Erklärung zu diesen Befehlen entnehmen Sie bitte [DGS+12], [Tea23] oder [Tea05]. Auch Farbumschaltungen wie \normalcolor sind möglich (siehe [Car17] 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 61 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.

Mit der Anweisung \usekomafont kann die aktuelle Schriftart auf die für das angegebene *Element* umgeschaltet werden.

**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. Das Beispiel setzt voraus, dass Option extendedfeature=title gesetzt ist (siehe Abschnitt 7.3, Seite 303).

```
\usefontofkomafont{Element}
\useencodingofkomafont{Element}
\usesizeofkomafont{Element}
\usefamilyofkomafont{Element}
\useseriesofkomafont{Element}
\useshapeofkomafont{Element}
```

v3.12 Manchmal werden in der Schrifteinstellung eines Elements auch Dinge vorgenommen, die mit der Schrift eigentlich gar nichts zu tun haben, obwohl dies ausdrücklich nicht empfohlen wird. Soll dann nur die Schrifteinstellung, aber keine dieser zusätzlichen Einstellungen ausgeführt werden, so kann statt `\usekomafont` die Anweisung `\usefontofkomafont` verwendet werden. Diese Anweisung übernimmt nur die Schriftgröße und den Grundlinienabstand, die Codierung (engl. *encoding*), die Familie (engl. *family*), die Strichstärke oder Ausprägung (engl. *font series*) und die Form oder Ausrichtung (engl. *font shape*).

Mit den übrigen Anweisungen können auch einzelne Schriftattribute übernommen werden. Dabei übernimmt `\usesizeofkomafont` sowohl die Schriftgröße als auch den Grundlinienabstand.

## 7.7. Dokumenttitel

Es gilt sinngemäß, was in Abschnitt 3.7 geschrieben wurde. Falls Sie also Abschnitt 3.7 bereits gelesen und verstanden haben, können Sie auf Seite 311 mit Abschnitt 7.8 fortfahren. Die Möglichkeiten von scrextend zum Dokumenttitel gehören jedoch zu den optionalen, erweiterten Möglichkeiten und stehen daher nur zur Verfügung, wenn beim Laden des Pakets `extendedfeature=title` gewählt wurde (siehe Abschnitt 7.3, Seite 303).

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.

titlepage=*Ein-Aus-Wert*
titlepage=firstiscover
\coverpagetopmargin
\coverpageleftmargin
\coverpagerightmargin
\coverpagebottommargin

Mit dieser Option wird ausgewählt, ob für die mit \maketitle (siehe Seite 308) gesetzte Titelei eigene Seiten verwendet werden oder stattdessen die Titelei von \maketitle als Titelkopf am Anfang einer neuen Seite gesetzt wird. Als *Ein-Aus-Wert* kann einer der Standardwerte für einfache Schalter aus Tabelle 2.5, Seite 41 verwendet werden.

Mit titlepage=true wird die Titelei in Form von Titelseiten ausgewählt. Die Anweisung \maketitle verwendet dabei 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. Die zusätzlichen Elemente finden sie auf den nachfolgenden Seiten.

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.

v3.12 Mit der dritten Möglichkeit, titlepage=firstiscover, werden nicht nur Titelseiten aktiviert. Es wird auch dafür gesorgt, dass die erste von \maketitle ausgegebene Titelseite, also entweder der Schmutztitel oder der Haupttitel, als Umschlagseite ausgegeben wird. Jede andere Einstellung für die Option titlepage hebt diese Einstellung wieder auf. Die Ränder dieser Umschlagseite werden über \coverpagetopmargin (oberer Rand), \coverpageleftmargin (linker Rand), \coverpagerightmargin (rechter Rand) und natürlich \coverpagebottommargin (unterer Rand) bestimmt. Die Voreinstellungen sind von den Längen \topmargin und \evensidemargin abhängig und können mit \renewcommand geändert werden.

Die Voreinstellung ist von der verwendeten Klasse abhängig und wird von scrextend kompatibel zu den Standardklassen erkannt. Setzt eine Klasse keine entsprechende Voreinstellung, so ist der Titelkopf voreingestellt.

\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. Ein Beispiel für eine einfache Titelseite mit `titlepage` finden Sie in Abschnitt 3.7, Seite 68.

`\maketitle[`*Seitenzahl*`]`

Während bei den Standardklassen nur maximal eine Titelseite mit den Angaben Titel, Autor und Datum existiert, können bei KOMA-Script mit `\maketitle` bis zu sechs Titelseiten gesetzt werden. Außerdem kennt `\maketitle` bei KOMA-Script noch ein optionales numerisches 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 87) 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 in Abschnitt 3.7 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 [BB13]) auch nicht empfehlenswert, diese Anweisungen in der Dokumentpräambel vor `\begin{document}` zu verwenden. Beispieldokumente finden Sie in Abschnitt 3.7 ab Seite 69.

`\extratitle{`*Schmutztitel*`}`
`\frontispiece{`*Frontispiz*`}`

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 ist das *Frontispiz*. 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`.

Ein Beispiel für eine einfache Titelseite mit Schmutztitel und Haupttitel finden Sie in Abschnitt 3.7, Seite 69.

```
\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. Für die Ausgabe wird die Schrift des gleichnamigen Elements verwendet (siehe Tabelle 3.4, Seite 71).

Die *Typisierung* wird unmittelbar über dem *Titel* in der Schrift des gleichnamigen Elements ausgegeben.

Der *Titel* wird in einer sehr großen Schrift gesetzt. Dabei finden Schriftumschaltungen für das Element `title` Anwendung (siehe Tabelle 3.4, Seite 71).

v2.97c

Der *Untertitel* steht knapp unter dem Titel in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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. Die Ausgabe erfolgt in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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. Die Ausgabe erfolgt in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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. Die Ausgabe erfolgt in der Schrift des gleichnamigen Elements (siehe Tabelle 3.4, Seite 71).

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. Damit die Schrifteinstellung für das Element `footnote` beim Paket scrextend Beachtung findet muss allerdings nicht nur die Titelerweiterung aktiviert sein, es muss auch dafür gesorgt sein, dass die Fußnoten mit diesem Paket gesetzt werden (siehe Einleitung von Abschnitt 7.11, Seite 313). Trifft dies nicht zu, so wird die Schrift verwendet, die von der Klasse oder anderen für die Fußnoten verwendeten Paketen vorgegeben ist.

v3.12 Für die Ausgabe der Titelelemente kann die Schrift mit Hilfe der Befehle `\setkomafont` und `\addtokomafont` (siehe Abschnitt 7.6, Seite 305) eingestellt werden. Die Voreinstellungen sind Tabelle 3.3, Seite 71 zu entnehmen.

Bis auf den *Kopf* und eventuelle Fußnoten werden alle Ausgaben horizontal zentriert. Die Formatierungen der einzelnen Elemente sind noch einmal kurz zusammengefasst in Tabelle 3.4, Seite 71 zu finden.

Ein Beispiel mit allen von KOMA-Script angebotenen Elementen für die Haupttitelseite finden Sie in Abschnitt 3.7 auf Seite 71.

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 sogenannten 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, das man in einem getrennten Dokument erstellt.

Seit KOMA-Script 3.12 kann man die erste von `\maketitle` ausgegebene Titelseite alternativ aber auch als Umschlagseite formatieren lassen. Dabei ändern sich nur die für diese Seite verwendeten Ränder (siehe Option `titlepage=firstiscover` auf Seite 307).

`\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*}

v3.12 KOMA-Script bietet eine eigene Widmungsseite. Diese Widmung wird zentriert und in der Voreinstellung mit etwas größerer Schrift gesetzt. Die genaue Schrifteinstellung für das Element `dedication`, die Tabelle 3.3, Seite 71 zu entnehmen ist, kann über die Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 7.6, Seite 305) verändert werden.

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

Ein Beispiel mit allen von KOMA-Script angebotenen Titelseiten finden Sie in Abschnitt 3.7 auf Seite 73.

## 7.8. Erkennung von rechten und linken Seiten

Es gilt sinngemäß, was in Abschnitt 3.11 geschrieben wurde. Falls Sie also Abschnitt 3.11 bereits gelesen und verstanden haben, können Sie in Abschnitt 7.9 auf Seite 311 fortfahren.

\Ifthispageodd{*Dann-Teil*}{*Sonst-Teil*}

v3.28 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 aktuell auf einer ungeraden Seite befindet. Anderenfalls kommt das Argument *Sonst-Teil* zur Anwendung.

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 18.1, Seite 481 zu finden. Ein Beispiel zur Verwendung von `\Ifthispageodd` ist in Abschnitt 3.11 auf Seite 83 angegeben.

## 7.9. Wahl eines vordefinierten Seitenstils

Das Paket scrextend definiert selbst keine Seitenstile, nutzt aber Seitenstile des LaTeX-Kerns.

\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 7.3, Seite 303). Welcher Seitenstil in diesem Fall für einen Titelkopf verwendet wird, ist im Makro `\titlepagestyle` festgelegt. In der Voreinstellung ist das der Seitenstil `plain`. Dieser Seitenstil wird bereits im LaTeX-Kern vordefiniert und sollte daher immer verfügbar sein.

## 7.10. Vakatseiten

Es gilt sinngemäß, was in Abschnitt 3.13 geschrieben wurde. Falls Sie also Abschnitt 3.13 bereits gelesen und verstanden haben, können Sie auf Seite 313 mit Abschnitt 7.11 fortfahren.

cleardoublepage=*Seitenstil*
cleardoublepage=current

Mit Hilfe dieser Option kann man den *Seitenstil* der Vakatseite bestimmen, die bei Bedarf von den Anweisungen `\cleardoublepage`, `\cleardoubleoddpage` oder `\cleardoubleevenpage` eingefügt wird, um bis zur gewünschten Seite zu umbrechen. Als *Seitenstil* sind dabei alle bereits definierten Seitenstile (siehe Abschnitt 7.9 ab Seite 311 und Kapitel 5 ab Seite 267) 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. Ein Beispiel für die Bestimmung des Seitenstils von Vakatseiten finden Sie in Abschnitt 3.13, Seite 91.

```
\clearpage
\cleardoublepage
\cleardoublepageusingstyle{Seitenstil}
\cleardoubleemptypage
\cleardoubleplainpage
\cleardoublestandardpage
\cleardoubleoddpage
\cleardoubleoddpageusingstyle{Seitenstil}
\cleardoubleoddemptypage
\cleardoubleoddplainpage
\cleardoubleoddstandardpage
\cleardoubleevenpage
\cleardoubleevenpageusingstyle{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 2.6, Seite 40) eine neue rechte Seite begonnen wird. Dazu wird gegebenenfalls eine linke Vakatseite im aktuellen Seitenstil ausgegeben.

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 `\cleardoubleoddpageusingstyle` 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`, `\cleardoubleevenpageusingstyle`.

Die Arbeitsweise der Anweisungen `\cleardoublestandardpage`, `\cleardoubleemptypage`, `\cleardoubleplainpage` und der ein Argument erwartenden Anweisung `\cleardoublepageusingstyle` entspricht beim Paket scrextend ebenso wie die Standard-Anweisung `\cleardoublepage` den entsprechenden, zuvor erklärten Anweisungen für den Umbruch zur nächsten ungeraden Seite.

Im doppelseitigen Satz führt `\cleardoubleoddpage` immer zur nächsten ungeraden Seite, `\cleardoubleevenpage` zur nächsten geraden Seite. Eine gegebenenfalls einzufügende Vakatseite wird mit dem über Option `cleardoublepage` festgelegten Seitenstil ausgegeben.

Ein Beispiel für die Verwendung von `\cleardoubleevenemptypage` finden Sie in Abschnitt 3.13, Seite 93.

## 7.11. Fußnoten

Es gilt sinngemäß, was in Abschnitt 3.14 geschrieben wurde. Falls Sie also Abschnitt 3.14 bereits gelesen und verstanden haben, können Sie auf Seite 316 mit Abschnitt 7.12 fortfahren. In der Voreinstellung wird die Formatierung der Fußnoten jedoch der verwendeten Klasse überlassen. Dies ändert sich, sobald die Anweisung `\deffootnote` (siehe Seite 315) verwendet wird. Die Einstellmöglichkeiten für die Trennlinie über den Fußnoten werden hingegen von scrextend nicht bereitgestellt.

`footnotes=`*Einstellung*
`\multfootsep`

Werden in der Voreinstellung `footnotes=nomultiple` 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 handele.

Mit `footnotes=multiple` werden Fußnoten, die unmittelbar aufeinander folgen, stattdessen mit einem Trennzeichen aneinander gereiht. Das in `\multfootsep` definierte Trennzeichen ist als

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

definiert. Es ist also mit einem Komma vorbelegt. Dieses kann umdefiniert werden.

Der gesamte Mechanismus ist kompatibel zu footmisc, Version 5.3d bis 5.5b (siehe [Fai11]) implementiert. Er wirkt sich sowohl auf Fußnotenmarkierungen aus, die mit `\footnote` gesetzt wurden, als auch auf solche, die direkt mit `\footnotemark` ausgegeben werden. Bei Problemen mit der verwendeten Klasse oder anderen Paketen, die Einfluss auf die Fußnoten nehmen, sollte Option `footnotes` nicht verwendet werden.

Eine Zusammenfassung möglicher Werte für die *Einstellung* von `footnotes` bietet Tabelle 3.11, Seite 94.

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

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`.

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 zuvor erklärten Anweisung `\multfootsep` gespeichert.

Beispiele und ergänzende Hinweise sind Abschnitt 3.14 ab Seite 95 zu entnehmen.

`\footref{`*`Referenz`*`}`

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. 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. Ein passendes Beispiel ist in Abschnitt 3.14 auf Seite 96 zu finden. Ab LaTeX 2021-05-01 wird die Anweisung übrigens von LaTeX selbst bereitgestellt.

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

KOMA-Script setzt 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 auf Seite 97 veranschaulicht die verschiedenen Parameter. 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. Das Paket scrextend überlässt hingegen in der Voreinstellung das Setzen der Fußnoten der verwendeten Klasse. Das Laden des Pakets allein sollte daher noch zu keinerlei Änderungen bei der Formatierung der Fußnoten oder der Fußnotenmarken führen. Zur Übernahme der Voreinstellungen der KOMA-Script-Klassen muss man vielmehr obige Einstellung selbst vornehmen, indem man den gezeigten Code in die Dokumentpräambel übernimmt.

Auf die Fußnote einschließlich der Markierung findet außerdem die für das Element `footnote`

eingestellte Schriftart Anwendung. Die Schriftart der Markierung kann jedoch mit Hilfe der Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 7.6, Seite 305) für das Element footnotelabel davon abweichend eingestellt werden. Siehe hierzu auch Tabelle 3.2, Seite 61. Voreingestellt ist jeweils keine Umschaltung der Schrift. Die Elemente finden bei scrextend jedoch nur dann Anwendung, wenn die Fußnoten mit diesem Paket gesetzt werden, also \deffootnote verwendet wurde. Bitte missbrauchen Sie das Element nicht für andere Zwecke, beispielsweise zur Verwendung von Flattersatz in den Fußnoten (siehe \raggedfootnote, Seite 316).

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}}
```

Dabei findet die Schriftart für das Element footnotereference Anwendung (siehe Tabelle 3.2, Seite 61). Die Markierungen im Text und in der Fußnote selbst sind also identisch. Die Schriftart kann mit den Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 7.6, Seite 305) jedoch geändert werden. Ohne \deffootnote kann sich diese auch auf die Markierung in der Fußnote auswirken.

Beispiele für die Verwendung von \deffootnote finden Sie in Abschnitt 3.14, Seite 97.

```
\raggedfootnote
```

v3.23

In der Voreinstellung werden die Fußnoten bei KOMA-Script genau wie bei den Standardklassen im Blocksatz gesetzt. Bei Verwendung von \deffootnote ist es möglich, die Formatierung abweichend vom restlichen Dokument zu ändern. Dazu ist \raggedfootnote umzudefinieren. Gültige Definitionen wären \raggedright, \raggedleft, \centering, \relax oder entsprechend der Voreinstellung eine leere Definition. Auch die Ausrichtungsbefehle des Pakets ragged2e sind zulässig (siehe [Sch09]). Ein passendes Beispiel ist in Abschnitt 3.14, Seite 98 zu finden.

```
\ExecuteDoHook{footnote/text/begin}
\ExecuteDoHook{footnote/text/end}
```

v3.36

Für Experten gibt es außerdem zwei Haken vom Typ *do-hook* (siehe Abschnitt 10.8 ab Seite 349). Der erste davon wird ganz zu Beginn von \@makefntext noch vor \raggedfootnote ausgeführt. Der zweite am Ende noch bevor der Absatz beendet ist. Derzeit werden beide Haken von KOMA-Script selbst nicht verwendet.

## 7.12. Schlauer Spruch

Es gilt sinngemäß, was in Abschnitt 3.17 geschrieben wurde. Jedoch 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. Falls Sie also Abschnitt 3.17 bereits gelesen und verstanden haben, können Sie nach dem Ende dieses Abschnitts auf Seite 318 mit Abschnitt 7.13 fortfahren.

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

Ein Spruch oder Zitat kann mit Hilfe der Anweisung `\dictum` gesetzt werden. Der Spruch wird hierzu zusammen mit einem optional anzugebenden *Urheber* in einer `\parbox` (siehe [Tea23]) 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 `\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 `dictum` eingestellten Schriftart, die mit den Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 7.6, Seite 305) geändert werden kann. Die Voreinstellung entnehmen Sie bitte Tabelle 3.16, Seite 124.

v3.10

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

```
\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 den *Urheber* als Argument. In der Voreinstellung ist `\dictumauthorformat` mit

```
\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 Elements `dictum` definiert werden. Die Voreinstellung entnehmen Sie bitte Tabelle 3.16, Seite 124. Eine Änderung ist mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 7.6, Seite 305) möglich.

## 7.13. Listen

Es gilt sinngemäß, was in Abschnitt 3.18 geschrieben wurde. Falls Sie also Abschnitt 3.18 bereits gelesen und verstanden haben, können Sie auf Seite 319 mit Abschnitt 7.14 fortfahren. Allerdings werden vom Paket scrextend nur die Umgebungen `labeling`, `addmargin` und `addmargin*` definiert. Alle anderen Listenumgebungen bleiben der Verantwortung der verwendeten Klasse überlassen.

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

Eine andere Form der in vielen Klassen als `description`-Umgebung vorhandenen Stichwortliste ist nur bei KOMA-Script-Klassen und scrextend vorhanden: die `labeling`-Umgebung. Im Unterschied zu `description` kann bei `labeling` ein Muster angegeben werden, dessen Länge die Einrücktiefe aller Stichpunkte ergibt. Zwischen Stichpunkt und Beschreibungstext kann ein optionales *Trennzeichen* festgelegt werden. Die Schriftart, die für die Hervorhebung des Schlüsselworts verwendet wird, kann mit Hilfe von `\setkomafont` und `\addtokomafont` (siehe Abschnitt 7.6, Seite 305) für das Element `labelinglabel` (siehe Tabelle 3.2, Seite 61) geändert werden. Für die davon abweichende Schriftart der Trennzeichen ist das Element `labelingseparator` (siehe ebenfalls Tabelle 3.2, Seite 61) zuständig. Beide sind in der Voreinstellung leer, es findet also keine Schriftumschaltung statt. In Abschnitt 3.18 auf Seite 130 finden Sie ein Beispiel für die Anwendung der Umgebung.

v3.02

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

Wie bei den in den Standardklassen und den KOMA-Script-Klassen vorhandenen Umgebungen `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 soll. 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. Die Umgebung ragt dann entsprechend in den Rand hinein. Im Beispiel in Abschnitt 3.18 auf Seite 135 wird dies gezeigt.

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 7.8, Seite 311) und `\ifthispagewasodd` (Abschnitt 18.1, Seite 481).

Im Zusammenspiel von Listen mit Absätzen ergeben sich für Laien häufiger Fragen. Daher widmet sich die weiterführende Erklärung zu Option `parskip` in Abschnitt 18.1, Seite 481 auch diesem Thema. Ebenfalls im Expertenteil in Abschnitt 18.1, Seite 481 wird die Problematik von mehrseitigen `addmargin*`-Umgebungen behandelt.

## 7.14. Randnotizen

Es gilt sinngemäß, was in Abschnitt 3.21 geschrieben wurde. Falls Sie also Abschnitt 3.21 bereits gelesen und verstanden haben, können Sie auf Seite 320 mit Kapitel 8 fortfahren.

`\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`.

Ein ausführliches Beispiel hierzu finden Sie in Abschnitt 3.21, Seite 155.

Für Experten sind in Abschnitt 18.1, Seite 481 Probleme bei der Verwendung von `\marginpar` dokumentiert. Diese gelten ebenso für `\marginline`. Darüber hinaus wird in Kapitel 16 ein Paket vorgestellt, mit dem sich auch Notizspalten mit eigenem Seitenumbruch realisieren lassen.

# Unterstützung für die Anwaltspraxis durch scrjura

Bis einschließlich Version 3.41 enthielt KOMA-Script offiziell das Paket scrjura zur Unterstützung anwaltlicher Dokumente. Dabei ging es vor allem um Satzungen, Gesetze, Kommentare dazu oder im weitesten Sinn um Verträge alle Art. Im Zuge der Restrukturierung von KOMA-Script wurde das Paket ausgelagert. Da der Vertrag das zentrale Element des Pakets ist, bekam es dabei den neuen Namen contract. Unter diesem Namen ist es nicht nur auf CTAN zu finden (siehe [Koh24]). Es wurde auch bereits in gängige TEX-Distributionen aufgenommen und kann daher über deren Paketmanager installiert werden.

Aus Kompatibilitätsgründen wird es für eine begrenzte Zeit in KOMA-Script weiterhin ein Paket scrjura geben. Dabei handelt es sich aber lediglich um eine andere Verpackung des neuen Pakets contract, bei dem ein Teil der Inkompatibilitäten zwischen dem neuen Paket und dem früheren scrjura bereinigt wurden. Damit sollte es in der Regel möglich sein, bisherige Dokumente auf Basis von scrjura weiterhin zu verarbeiten. Für neue Dokumente wird unbedingt empfohlen, stattdessen zu contract zu wechseln. Bei der Überarbeitung alter Dokumente ist eine Umstellung ebenfalls zu empfehlen. Für die dabei zu berücksichtigenden Änderungen sind die Angaben in der Anleitung zum Paket contract maßgeblich.

`\Clause{`*`Optionen`*`}`
`\SubClause{`*`Optionen`*`}`

Der wichtigste Unterschied bei Verwendung von scrjura gegenüber contract ist, dass die *`Optionen`* zu den beiden Anweisungen `\Clause` und `\SubClause` innerhalb einer `contract`-Umgebung bei contract ein optionales Argument darstellen, also in eckigen Klammern anzugeben sind. Beim Paket scrjura waren die *`Optionen`* hingegen immer ein erforderliches Argument also in geschweiften Klammern anzugeben. Das ist mit scrjura auch weiterhin der Fall.

# Das KOMA-Script-Logo mit Paket scrlogo

Das Paket scrlogo ist das kleinste KOMA-Script-Paket, das derzeit nur eine einzige Anweisung bereitstellt. Dafür wird das Paket von allen anderen KOMA-Script-Paketen und den KOMA-Script-Klassen geladen. Die Anweisunge steht also bei Verwendung aller KOMA-Script-Pakete und KOMA-Script-Klassen zur Verfügung.

`\KOMAScript`

Diese Anweisung gibt die Wortmarke »KOMA-Script« in serifenloser Schrift und mit leichter Sperrung des in Versalien gesetzten Teils aus. Die Definition erfolgt mit `\DeclareRobustCommand`. Da auch Pakete, die nicht zu KOMA-Script gehören, diese Wortmarke definieren können, sollte man die Anweisung jedoch nicht als Indiz für die Verwendung eines KOMA-Script-Pakets verstehen. Als Besonderheit enthält scrlogo außerdem Code, um bei Verwendung von hyperref sicherzustellen, dass im PDF-Kontext, also beispielsweise im Text der Lesezeichen keine Warnungen aufgrund der dann nicht verfügbaren Sperrung auftreten. Dabei spielt es keine Rolle, ob hyperref vor oder nach scrbase geladen wird.

v3.36

# 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. Näheres zu dem Paket is [Car99a] zu entnehmen.

Das Paket keyval erlaubt es, Schlüssel zu definieren und diesen Werte zuzuweisen. Auch die Optionen, die scrbase bereitstellt, verwenden die keyval-Syntax: *Schlüssel=Wert*.

## 10.2. Schlüssel als Eigenschaften von Familien und deren Mitgliedern

Wie bereits in Abschnitt 10.1 erwähnt, setzt scrbase bei Schlüsseln und deren Werten auf das Paket keyval. Allerdings erweitert es dessen Funktionalität. Während bei keyval ein Schlüssel einer Familie gehört, kennt scrbase 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, vorzugsweise einem Punkt, zu beginnen. 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 sind die Familien »KOMA« und »KOMAarg« 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 entscheiden Sie sich für den Familienname »Fleischermeister« und fügen in jedem der Pakete die Zeilen

```
\DefineFamily{Fleischermeister}
\DefineFamilyMember{Fleischermeister}
```

ein. Dadurch wird 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}
\FamilyKeyState
\FamilyKeyStateUnknown
\FamilyKeyStateProcessed
\FamilyKeyStateUnknownValue
\FamilyKeyStateNeedValue
```

Mit dieser Anweisung wird ein *Schlüssel* definiert. Ist ein *Mitglied* angegeben, so ist der *Schlüssel* eine Eigenschaft dieses Mitglieds der angegebenen *Familie*. Ist [*Mitglied*] nicht angegeben, so wird wieder das Mitglied ».\@currname.\@currext« angenommen. Ist das *Mitglied* hingegen leer, so wird ein Familienschlüssel anstelle eines Mitgliederschlüssels definiert.

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]{erweiterte Aktion}
```

wobei `\define@key` im keyval-Paket definiert ist (siehe [Car99a]). Allerdings kommen zu dem Aufruf von `\define@key` noch einige zusätzliche Vorkehrungen und auch die *Aktion* wird automatisch um einige interne Aktionen erweitert.

v3.12 Erfolg oder Misserfolg der Ausführung der *Aktion* werden über `\FamilyKeyState` an scrbase zurückgemeldet, damit dieses je nach Bedarf und Verwendung des Schlüssels weitere Maßnahmen ergreifen kann. Das kann beispielsweise eine Fehlermeldung aber auch nur die Signalisierung einer unbekannten Option sein.

Der Zustand `\FamilyKeyState` ist in der Voreinstellung identisch mit dem Zustand `\FamilyKeyStateUnknown`. Das bedeutet, dass es nicht sicher ist, ob der Schlüssel korrekt verarbeitet werden konnte. Findet scrbase nach Ausführung der *Aktion* noch immer diesen Zustand vor, so wird ein Hinweis in die `log`-Datei geschrieben und im weiteren der Zustand `\FamilyKeyStateProcessed` angenommen.

Der Zustand `\FamilyKeyStateProcessed` signalisiert, dass der Schlüssel und die Wertzuweisung an den Schlüssel vollständig abgeschlossen wurde und alles in Ordnung ist. Auf den Zustand kann einfach durch Aufruf von `\FamilyKeyStateProcessed` selbst umgeschaltet werden.

Der Zustand `\FamilyKeyStateUnknownValue` signalisiert, dass der Schlüssel zwar verarbeitet wurde, ihm jedoch ein Wert zugewiesen werden sollte, der nicht erlaubt ist. Diesen Zustand meldet beispielsweise typearea, wenn versucht wird, an Option `twoside` den Wert `unbekannt` zuzuweisen. Die Umschaltung auf den Zustand erfolgt einfach durch Aufruf von `\FamilyKeyStateUnknownValue` selbst.

Der Zustand `\FamilyKeyStateNeedValue` signalisiert, dass der Schlüssel nicht verarbeitet werden konnte, weil er zwingend einen Wert erwartet, er aber ohne Wertzuweisung aufgerufen wurde. Dieser Zustand wird automatisch gesetzt, wenn ein Schlüssel, der keinen *Säumniswert* besitzt, ohne Wertzuweisung verwendet wird. Theoretisch wäre aber auch eine explizite Umschaltung darauf durch Aufruf von `\FamilyKeyStateNeedValue` selbst möglich.

Des Weiteren können zusätzliche Fehlerzustände definiert werden, indem man `\FamilyKeyState` auf eine kurze Meldung umdefiniert. In der Regel sollten jedoch die vier vordefinierten Zustände verwendet werden.

**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 entspre-

chend 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
  \FamilyKeyStateProcessed
}
```

Als Wert ist daher beim Aufruf `true` oder `false` erlaubt. Ein Test auf unerlaubte Werte existiert in diesem Beispiel nicht. Stattdessen wird immer zurückgemeldet, dass der Schlüssel vollständig und korrekt verarbeitet wurde. Wird der Schlüssel später verwendet, so muss entweder einer der erlaubten Werte zugewiesen oder ein Aufruf ohne Wertzuweisung verwendet werden. In letzterem Fall kommt der Säumniswert *true* zum Einsatz.

Für die anderen beiden Pakete kann das fast identisch definiert werden. Lediglich die Zeichenfolge »`Salami`« ist jeweils zu ersetzen.

`\RelaxFamilyKey[`*Mitglied*`]{`*Familie*`}{`*Schlüssel*`}`

v3.15

Wurde zuvor *Schlüssel* für das *Mitglied* der *Familie* definiert, so wird diese Definition quasi aufgehoben. Der *Schlüssel* ist dann für dieses *Mitglied* der *Familie* nicht mehr definiert. Die Verwendung für einen *Schlüssel*, der für dieses *Mitglied* der *Familie* gar nicht definiert ist, ist ebenfalls zulässig und bleibt wirkungslos.

Ist kein *Mitglied* angegeben, so wird genau wie bei Anweisung `\DefineFamilyKey` wieder das Mitglied »`.\@currname.\@currext`« angenommen. Allerdings wird `\RelaxFamilyKey` selten während des Ladens eines Pakets, sondern häufiger zur Laufzeit verwendet. Daher ist *Mitglied* in der Regel auch explizit anzugeben.

`\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 können. Diese Anweisung stellt daher eine 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 nach der Definition der Schlüssel die Zeile

```
\FamilyProcessOptions{Fleischermeister}
```

ergänzt wird, so kann der Anwender bereits beim Laden der Pakete mit `\usepackage` die Eigenschaft `Aufschnitt` als Option angeben. Wird die Option bei `\documentclass` und damit global angegeben, so wird die Eigenschaft automatisch bei allen drei Paketen gesetzt, wenn alle drei Pakete einzeln 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.

Wie auch `\ProcessOptions` darf `\FamilyProcessOptions` nicht während der Ausführung von Optionen verwendet werden. Es ist also insbesondere auch nicht erlaubt, innerhalb der Ausführung von Optionen Pakete zu laden.

`\BeforeFamilyProcessOptions[`*Mitglied*`]{`*Familie*`}{`*Code*`}`

v3.18

Insbesondere Autoren von Wrapper-Klassen wünschen manchmal, in ein Paket oder eine Klasse noch vor der Verarbeitung der Optionen eines zukünftig geladenen Pakets oder einer zukünftig geladenen Klasse mit `\FamilyProcessOptions` eingreifen zu können. Dies ist mit `\BeforeFamilyProcessOptions` möglich. Das Paket scrbase bietet dafür einen sogenannten Haken (engl. *hook*). Diesem kann man mit `\BeforeFamilyProcessOptions` neuen *Code* hinzufügen. Die Parameter *Mitglied* und *Familie* entsprechen dabei denen von `\FammilyProcessOptions`. Allerdings kann man auch den Haken von Familien-Mitgliedern *Code* hinzufügen, wenn bisher die *Familie* oder das *Mitglied* noch gar nicht definiert ist.

Der Haken eines Familien-Mitglieds wird übrigens nach dessen Ausführung automatisch gelöscht. Verwendet man hingegen ein leeres *Mitglied*, so wird dieser Haken für alle Mitglieder der *Familie* ausgeführt und bleibt auch über die Ausführung hinaus erhalten.

**Beispiel:** Sie schreiben ein Paket Rauchwurst, das selbst Mettwurst lädt. Allerdings wollen Sie nicht, dass für dieses Paket die Option `Aufschnitt` gesetzt werden kann. Daher deaktivieren sie die Option vor dem Laden des Pakets über `\BeforeFamilyProcessOptions`.

```
\RequirePackage{scrbase}
\BeforeFamilyProcessOptions[.Mettwurst.sty]%
                           {Fleischermeister}{%
  \RelaxFamilyKey[.Mettwurst.sty]%
                 {Fleischermeister}{Aufschnitt}%
}
\RequirePackageWithOptions{Mettwurst}
```

Versucht nun jemand Ihr Paket mit Option `Aufschnitt` zu laden, so meldet das Paket Mettwurst, dass diese Option nicht bekannt ist. Wird die Option `Aufschnitt` als globale Option angegeben, so ignoriert Paket Mettwurst diese. Voreinstellungen innerhalb des Pakets Mettwurst, die beispielsweise mit `\FamilyExecuteOptions` noch vor `\FamilyProcessOptions` erfolgen könnten, wären allerdings davon unabhängig. Aber natürlich kann man eigene Voreinstellungen von Rauchwurst aus ebenfalls per `\BeforeFamilyProcessOptions` in Mettwurst einschleusen.

`\FamilyExecuteOptions[`*Mitglied*`]{`*Familie*`}{`*Optionenliste*`}`

Diese Anweisung stellt eine 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 den Aufruf

```
\FamilyExecuteOptions{Fleischermeister}
                     {Aufschnitt}
```

ergänzt werden.

v3.20 Wird `\FamilyExecuteOptions` mit einer nicht definierten Option in der *Optionenliste* aufgerufen, so wird normalerweise ein Fehler ausgegeben. Eine Ausnahme von dieser Regel ist, wenn für das *Mitglied* eine Option namens `@else@` definiert wurde. In diesem Fall wird statt der unbekannten Option eben diese Option `@else@` verwendet. Der an `@else@` übergebene Wert ist dabei die nicht definierte Option mit dem im Aufruf angegebenen Wert. Innerhalb von KOMA-Script wird das beispielsweise genutzt, um die Stil-Option bei der Definition von Gliederungsbefehlen vor allen anderen auszuwerten.

Diese Anweisung darf auch innerhalb der Ausführung von Optionen verwendet werden.

\FamilyOptions{*Familie*}{*Optionenliste*}

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. 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.

Durch die Anweisung 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. Dies geschieht ebenfalls, wenn zwar für einige Mitglieder ein Schlüssel existiert, jedoch jedes dieser Mitglieder über \FamilyKeyState einen Fehler zurückmeldet.

**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]
\RequirePackage{scrbase}
\DefineFamily{Fleischermeister}
\DefineFamilyMember{Fleischermeister}
\FamilyProcessOptions{Fleischermeister}\relax
\FamilyOptions{Fleischermeister}{Aufschnitt}
```

Sollte noch kein Wurstpaket geladen sein, so würde nun eine Fehlermeldung wegen der nicht definierten Option »Aufschnitt« ausgegeben. Das kann vermieden werden, indem vor der letzten Zeile für das Paket selbst ebenfalls ein entsprechender Schlüssel definiert wird:

```
\DefineFamilyKey{Fleischermeister}%
                {Aufschnitt}[true]{}%
```

Allerdings produzieren so Wurstpakete, die nach Paket Wurstsalat geladen werden, keinen Aufschnitt. Dies kann man ebenfalls ändern:

```
\AtBeginDocument{%
  \DefineFamilyKey[.Wurstsalat.sty]%
                  {Fleischermeister}%
                  {Aufschnitt}[true]{}%
}
\DefineFamilyKey{Fleischermeister}%
                {Aufschnitt}[true]{%
  \AtBeginDocument{%
    \FamilyOptions{Fleischermeister}%
```

```
                    {Aufschnitt=#1}%
  }%
}%
```

Somit wird zunächst während `\begin{document}` die Option so definiert, dass sie für das Paket Wurstsalat keine Funktion mehr ausübt. Da nach dem Laden von Wurstsalat, innerhalb von `\begin{document}` die beiden Anweisungen `\@currname` und `\@currext` nicht mehr den Dateinamen des Pakets enthalten, muss zwingend das optionale Argument von `\DefineFamilyKey` verwendet werden.

Bis zu dieser Umdefinierung der Option wird jedoch eine Definition verwendet, die während `\begin{document}` die Option erneut für die Familie und all ihre Mitglieder ausführt und damit auch für andere Wurstpakete setzt. Die Verzögerung der Ausführung von `\FamilyOptions` ist hier entscheidend. Zum einen werden nur so später geladene Wurstpakete erfasst. Zum anderen wird dadurch sichergestellt, dass die eigene Option `Aufschnitt` bereits umdefiniert wurde. Dadurch wird eine endlose Rekursion vermieden.

v3.27 Für die Option Namens `@else@` existiert ähnlich wie bei Anweisung `\FamilyExecuteOptions` eine Sonderbehandlung. Ist eine solche für ein Mitglied definiert, so wird sie immer dann ausgeführt, wenn das Mitglied eine angegebene Option nicht kennt. Ist für die Familie selbst eine Option `@else@` definiert, so wird sie nur aufgerufen, wenn weder die Familie noch eines ihrer Mitglieder eine angegebene Option vollständig verarbeiten konnte und mit `\FamilyKeyStateProcessed` quitiert hat.

`\FamilyOption{`*Familie*`}{`*Option*`}{`*Werteliste*`}`

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 weitere Funktionsweise ist der vorhergehenden Erklärung zu `\FamilyOptions` 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.

v3.27 Intern basiert `\FamilyOption` auf `\FamilyOptions`. Die Verarbeitung einer Option `@else@` findet daher in gleicher Weise statt.

`\AtEndOfFamilyOptions{`*Aktion*`}`
`\AtEndOfFamilyOptions*{`*Aktion*`}`

v3.12 Manchmal ist es vorteilhafter, wenn nicht jede Wertzuweisung an eine Option unmittelbar eine *Aktion* auslöst, sondern dies erst geschieht, wenn alle Wertzuweisungen innerhalb eines Aufrufs von `\FamilyProcessOptions` oder `\FamilyExecuteOptions` respektive `\FamilyOptions` oder `\FamilyOption` abgeschlossen sind. Genau das ist mit Hilfe von `\AtEndOfFamilyOptions` v3.23 und der zugehörigen Sternvariante möglich. Die Rückmeldung von Fehlerzuständen ist über die Anweisungen jedoch ebenso wenig möglich wie die Verwendung der Anweisungen außerhalb der Ausführung von Optionen.

Die beiden Varianten unterscheiden sich im Falle von verschachtelt definierten Optionen, wenn also die Ausführung einer Option den Aufruf einer oder mehrerer anderer Optionen bedingt. In diesem Fall werden alle per `\AtEndOfFamilyOptions` festgelegten Aktionen ausgeführt, sobald der innerste Optionenaufruf endet. Dagegen werden die per `\AtEndOfFamilyOptions*` festgelegten Aktionen erst mit dem Ende des äußersten Optionenaufrufs ausgeführt. Die Reihenfolge der Aktionen ist dabei jedoch ausdrücklich unbestimmt! Weder ist sichergestellt, dass zuerst angeforderte Aktionen auch zuerst ausgeführt werden, noch die umgekehrte Reihenfolge.

```
\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 Wert `true` oder `false` angibt. Es existierte keine Fehlerbehandlung, falls der Anwender einen falschen Wert verwendet. Da solche booleschen 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 324). 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 auf Seite 337 erklärte Anweisung `\FamilyUnknownKeyValue` aufgerufen und so `\FamilyKeyState` (siehe Seite 324) entsprechend gesetzt. Dadurch kann dann gegebenenfalls eine Meldung über eine unbekannte Wertzuweisung ausgegeben werden.

**Beispiel:** Der Schlüssel `Aufschnitt` soll in den Wurstpaketen etwas robuster definiert werden, so dass Fehleingaben erkannt und behandelt 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 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
}{}%
\DefineFamilyKey[]{Fleischermeister}%
                  {Aufschnitt}[true]{%
  \FamilySetBool{Fleischermeister}%
                {Aufschnitt}%
```

```
                {@Aufschnitt}%
                {#1}%
}
```

oder einfacher

```
\FamilyBoolKey[]{Fleischermeister}%
                {Aufschnitt}%
                {@Aufschnitt}
```

unter Ausnutzung des Hinweises bezüglich leerer Mitglieder im Gegensatz zum Weglassen des optionalen Arguments auf Seite 325, der nicht nur für \DefineFamilyKey, sondern entsprechend auch für \FamilyBoolKey gilt.

Da \FamilyKeyState bereits von \FamilySetBool gesetzt wird, kann innerhalb der Definition der Option mit Hilfe von \DefineFamilyKey der Status gegebenenfalls auch abgefragt werden. So könnte man im ersten Fall beispielsweise nach \FamilySetBool einen Test der Art:

```
\ifx\FamilyKeyState\FamilyKeyStateProcessed
    ...
\else
    ...
\fi
```

ergänzen, um zusätzliche Aktionen in Abhängigkeit davon, ob \FamilySetBool erfolgreich war oder nicht, auszuführen. Es ist zu beachten, dass an dieser Stelle unbedingt ein Test mit Hilfe von \ifx vorzunehmen ist. Expandierende Tests wie \Ifstr sind hier zu vermeiden. Sie können abhängig vom aktuellen Status und dem Vergleichszustand zu unterschiedlichen Fehlermeldungen und auch zu falschen Ergebnissen führen.

```
\FamilyInverseBoolKey[Mitglied]{Familie}{Schlüssel}{Schaltername}
\FamilySetInverseBool{Familie}{Schlüssel}{Schaltername}{Wert}
```

v3.27

Diese beiden Anweisungen unterscheiden sich von \FamilyBoolKey beziehungsweise \FamilySetBool nur dadurch, dass die Logik invertiert wird. Das heißt, dass die Werte `true`, `yes` und `on` den durch *Schaltername* angegebenen Schalter auf \iffalse setzen und damit deaktivieren, während `false`, `no` und `off` ihn zu \iftrue werden lassen, also aktivieren.

```
\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 entweder links oder nicht links, sondern auch links, mittig oder rechts sein. Intern unterscheidet man solche Einstellungen dann gerne mit Hilfe von `\ifcase`. Diese TEX-Anweisung erwartet wiederum einen numerischen 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 *Schlüssel* 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 zwar 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, dem Anwender wie in diesem Beispiel 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` Anweisung `\FamilyUnknownKeyValue` (siehe Seite 337) aufgerufen. Dies führt zu einer Fehlersignalisierung, indem `\FamilyKeyState` den Status `\FamilyKeyStateUnknownValue` annimmt (siehe Seite 324). Daraus resultiert beispielsweise bei der Verwendung als lokale Option dann eine Fehlermeldung. Ebenso wird beim Aufruf von `\FamilySetNumerical` auch der Erfolg via `\FamilyKeyStateProcessed` in `\FamilyKeyState` signalisiert.

`\FamilyCounterKey[`*Mitglied*`]{`*Familie*`}{`*Schlüssel*`}[`*Säumniswert*`]{`*LaTeX-Zähler*`}`
`\FamilySetCounter{`*Familie*`}{`*Schlüssel*`}{`*LaTeX-Zähler*`}{`*Wert*`}`

v3.12 Während bei `\FamilyNumericalKey` ein Makro aufgrund eines symbolischen Wertes auf einen korrespondierenden numerischen Wert gesetzt wurde, gibt es natürlich auch Fälle, in denen ein *Schlüssel* direkt einen *LaTeX-Zähler* repräsentiert, dem unmittelbar ein numerischer *Wert* zugewiesen werden soll. Dazu dient die Anweisung `\FamilyCounterKey`, von der intern dann `\FamilySetCounter` aufgerufen wird. Dabei finden einige grundlegende Prüfungen des *Wert*-Arguments statt um festzustellen, ob dieses Argument für eine Zuweisung an einen Zähler in Frage kommt. Die Zuweisung findet nur statt, wenn diese Prüfungen gelingen. Allerdings können hier nicht alle Fehler erkannt werden, so dass eine falsche Zuweisung auch zu einer Fehlermeldung von TEX selbst führen kann. Erkannte Fehler werden hingegen über `\FamilyKeyStateUnknownValue` signalisiert.

v3.15 Wurde kein Wert übergeben, so wird stattdessen der *Säumniswert* eingesetzt. Falls kein *Säumniswert* angegeben wird, kann später der *Schlüssel* nur mit Wertübergabe verwendet werden.

```
\FamilyCounterMacroKey[Mitglied]{Familie}{Schlüssel}[Säumniswert]{Makro}
\FamilySetCounterMacro{Familie}{Schlüssel}{Makro}{Wert}
```

v3.12 Diese beiden Anweisungen unterscheiden sich von den zuvor erklärten \FamilyCounterKey und \FamilySetCounter nur dadurch, dass nicht ein LaTeX-Zähler auf einen Wert gesetzt wird, sondern ein *Makro* mit diesem Wert definiert wird. Auch dabei wird *Wert* lokal einem Zähler zugewiesen und dann dessen expandierter Wert verwendet. Es gilt daher der Wert zum Zeitpunkt des Aufrufs der Option.

```
\FamilyLengthKey[Mitglied]{Familie}{Schlüssel}[Säumniswert]{Länge}
\FamilySetLength{Familie}{Schlüssel}{Länge}{Wert}
\FamilyLengthMacroKey[Mitglied]{Familie}{Schlüssel}[Säumniswert]{Makro}
\FamilySetLengthMacro{Familie}{Schlüssel}{Makro}{Wert}
\FamilyUseLengthMacroKey[Mitglied]{Familie}{Schlüssel}[Säumniswert]{Makro}
\FamilySetUseLengthMacro{Familie}{Schlüssel}{Makro}{Wert}
```

v3.12 Über \FamilyLengthKey kann ein *Schlüssel* definiert werden, der eine *Länge* repräsentiert. Dabei spielt es keine Rolle, ob eine LaTeX-Länge, ein TeX-Abstand oder eine TeX-Ausdehnung als *Länge* verwendet wird. Intern wird die *Länge* über \FamilySetLength gesetzt. Dabei finden einige grundlegende Prüfungen des *Wert*-Arguments statt um festzustellen, ob dieses Argument für eine Zuweisung an eine *Länge* in Frage kommt. Die Zuweisung findet nur statt, wenn diese Prüfungen gelingen. Allerdings können hier nicht alle Fehler erkannt werden, so dass eine falsche Zuweisung auch zu einer Fehlermeldung von TeX selbst führen kann. Erkannte Fehler werden hingegen über \FamilyKeyStateUnknownValue signalisiert.

v3.15 Wurde kein Wert übergeben, so wird stattdessen der *Säumniswert* eingesetzt. Falls kein *Säumniswert* angegeben wird, kann später der *Schlüssel* nur mit Wertübergabe verwendet werden.

\FamilyLengthMacroKey und \FamilySetLengthMacroKey sowie \FamilyUseLengthMacro und \FamilySetUseLengthMacro verwenden für die Speicherung des Wertes keine *Länge*, sondern ein *Makro*. Bei \FamilyLengthMacroKey und \FamilySetLengthMacro wird wie im Falle einer *Länge* der aktuelle *Wert* zum Zeitpunkt der Verwendung des Schlüssels beziehungsweise der Anweisung in *Makro* gespeichert. Dagegen wird bei \FamilyUseLengthMacroKey und \FamilySetUseLengthMacro *Wert* selbst gespeichert. Damit wird *Wert* dann bei jeder Verwendung von *Makro* neu ausgewertet.

v3.20

```
\FamilyStringKey[Mitglied]{Familie}{Schlüssel}[Säumniswert]{Makro}
\FamilyCSKey[Mitglied]{Familie}{Schlüssel}[Säumniswert]{Makroname}
```

v3.08 Hier wird nun ein Schlüssel definiert, der jeden beliebigen Wert annehmen kann. Der Wert wird in dem angegebenen *Makro* gespeichert. Wird das optionale Argument für den *Säumniswert* weggelassen, so entspricht \FamilyStringKey:

```
\DefineFamilyKey[Mitglied]{Familie}{Schlüssel}
                {\defMakro{#1}}
```

Existiert das optionale Argument für den *Säumniswert* so entspricht die Anweisung:

```
\DefineFamilyKey[Mitglied]{Familie}{Schlüssel}
                [Säumniswert]
                {\defMakro{#1}\FamilyKeyStateProcessed}
```

**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 `Salatgewicht` heißen:

```
\newcommand*{\Salatgewicht}{250g}
\FamilyStringKey{Fleischermeister}%
                {Salatgewicht}[250g]%
                {\Salatgewicht}
```

Soll nach einer Änderung wie

```
\FamilyOptions{Fleichermeister}{Salatgewicht=100g}
```

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.

Da alle Werte unmittelbar für eine Makrodefinition verwendet werden, gibt es keine unbekannten Werte. In der Wertzuweisung an den *Schlüssel* dürfen jedoch keine Absätze enthalten sein.

v3.25 Im Unterschied zu `\FamilyStringKey` erwartet `\FamilyCSKey` kein Makro als letztes Argument, sondern lediglich den Namen eines Makros, also beispielsweise nicht `{\Salatgewicht}`, sondern `{Salatgewicht}`.

`\FamilyUnknownKeyValue{Familie}{Schlüssel}{Wert}{Werteliste}`

Diese Anweisung signalisiert über `\FamilyKeyState` einen Fehler aufgrund eines unbekannten oder unerlaubten Wertes. Dabei wird als *Werteliste* eine durch Komma separierte Liste von
v3.12 erlaubten Werten der Form: '*Wert*', '*Wert*' ... erwartet. Allerdings wird die *Werteliste* derzeit nicht ausgewertet.

**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]{%
  \FamilySetBool{Fleischermeister}{Aufschnitt}%
                                  {@Aufschnitt}%
                                  {#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed
    \@Feinschnittfalse
  \else
    \Ifstr{#1}{fein}{%
      \@Aufschnitttrue
      \@Feinschnittrue
      \FamilyKeyStateProcessed
    }{%
      \FamilyUnknownKeyValue{Fleichermeister}%
                            {Aufschnitt}%
                            {#1}{%
                              'true', 'on', 'yes',
                              'false', 'off', 'no',
                              'fein'%
                            }%
    }%
  \fi
}%
```

Zunächst wird versucht, den booleschen Schalter für Aufschnitt über \FamilySetBool zu setzen. Gelingt dies, wurde also \FamilyKeyState zu \FamilyKeyStateProcessed definiert, wird der Feinschnitt abgeschaltet. Anderenfalls wird überprüft, ob anstelle eines gültigen Wertes für einen booleschen Schalter `fein` übergeben wurde. In diesem Fall wird sowohl der Feinschnitt, als auch Aufschnitt aktiviert und mit Hilfe von \FamilyKeyStateProcessed der Erfolgs-Status gesetzt. Ist auch das nicht der Fall, wird der von \FamilySetBool signalisierte Fehler-Zustand neu auf \FamilyKeyStateUnknownValue gesetzt. Die Liste der üblichen erlaubten Werte von \FamilySetBool wird dabei um `fein` ergänzt. Da diese Liste jedoch inzwischen nicht mehr verwendet wird, hätte man auf den Aufruf von `\FamilyUnknownKeyValue` im Beispiel auch einfach verzichten und damit den Fehlerstatus von \FamilySetBool übernehmen können.

Die bei den Tests verwendete Anweisung \Ifstr ist auf Seite 339 in Abschnitt 10.3 erklärt.

## 10.3. Verzweigungen

Das Paket scrbase stellt diverse Verzweigungsanweisungen zur Verfügung. Dabei wird überwiegend die LaTeX-Syntax mit Argumenten eingesetzt, die man von Anweisungen wie `\IfFileExists` kennt.

`\Ifundefinedorrelax{`*Name*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

v3.28 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 auch bei älteren LaTeX-Versionen weder Hash-Speicher belegt noch `\`*Name* zu `\relax`, wenn `\`*Name* zuvor undefiniert war.

`\Ifnotundefined{`*Name*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

v3.28 Ist die Anweisung mit dem angegebenen Namen bereits definiert wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Da es bereits ein Paket mit einer inkompatiblen Anweisung `\Ifdefined` gibt, musste leider diese etwas unhandliche Bezeichnung gewählt werden.

`\Ifstr{`*Zeichenfolge*`}{`*Zeichenfolge*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

v3.28 Die beiden Argumente *Zeichenfolge* werden expandiert und dann verglichen. Sind sie gleich, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*.

`\Ifstrstart{`*Zeichenfolge*`}{`*Zeichenfolge*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

v3.28 Die beiden Argumente *Zeichenfolge* werden expandiert und dann verglichen. Beginnt die erste Zeichenfolge, von Leerzeichen abgesehen, mit der zweiten Zeichenfolge, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*.

`\IfArgIsEmpty{`*Argument*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

v3.19 Ist das *Argument* leer, so wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Bereits ein Leerzeichen gilt als nicht leer. Da das *Argument* nicht expandiert wird, gelten Makros, die zu einem leeren Ergebnis expandieren, ebenfalls als nicht leeres *Argument*. Dasselbe gilt für `\relax` oder nicht definierte Makros.

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

v3.28 Wird eine PDF-Datei erzeugt, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Dabei ist es unerheblich, ob die PDF-Datei mit Hilfe von luaTeX, pdfTeX, VTeX oder XeTeX erzeugt wird.

\Ifpsoutput{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wird eine PostScript-Datei erzeugt, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. VTEX kann PostScript direkt erzeugen, was hier 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.

\Ifdvioutput{*Dann-Teil*}{*Sonst-Teil*}

v3.28 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.

\Ifisdimen{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in einem \dimen, also einem TEX-Längenregister, resultiert, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Die Anweisung ist nicht voll expandierbar.

\Ifisdimension{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in etwas resultiert, das syntaktisch dem Wert einer Länge entspricht, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Es ist zu beachten, dass derzeit unbekannte oder fehlende Einheiten zu einer Fehlermeldung führen. Die Anweisung ist nicht voll expandierbar.

\Ifdimen{*Zeichenfolge*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Der *Dann-Teil* wird ausgeführt, wenn die einfache Expansion der *Zeichenfolge* eine gültige Länge mit einer gültigen Längeneinheit ergibt. Anderenfalls wird der *Sonst-Teil* verwendet.

\Ifisdimexpr{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in einer \dimexpr, also einem $\varepsilon$-TEX-Längenausdruck resultiert, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Es ist zu beachten, dass fehlerhafte Ausdrücke zu Fehlermeldungen führen. Die Anweisung ist nicht voll expandierbar.

\Ifisskip{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in einem \skip, also einem TEX-Abstand, resultiert, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Die Anweisung ist nicht voll expandierbar.

\Ifisglue{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in etwas resultiert, das syntaktisch dem Wert eines Abstandes entspricht, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Es ist zu beachten, dass derzeit unbekannte oder fehlende Einheiten zu einer Fehlermeldung führen. Die Anweisung ist nicht voll expandierbar.

\Ifisglueexpr{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in einer \glueexpr, also einem $\varepsilon$-TeX-Abstandsausdruck resultiert, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Es ist zu beachten, dass fehlerhafte Ausdrücke zu Fehlermeldungen führen. Die Anweisung ist nicht voll expandierbar.

\Ifiscounter{*Zähler*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn *Zähler* ein definierter LaTeX-Zähler ist, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Die Anweisung ist nicht voll expandierbar.

\Ifiscount{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in einem \count, also einem TeX-Zähler, resultiert, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Die Anweisung ist nicht voll expandierbar. Für einen Test auf einen LaTeX-Zähler siehe \Ifiscounter.

\Ifisinteger{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in etwas resultiert, das syntaktisch dem Wert eines Zählers entspricht, also eine negative oder positive ganze Zahl ist, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Die Anweisung ist nicht voll expandierbar.

\Ifnumber{*Zeichenfolge*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Der *Dann-Teil* wird ausgeführt, wenn die einfache Expansion der *Zeichenfolge* nur aus Ziffern besteht. In allen anderen Fällen wird der *Sonst-Teil* verwendet.

\Ifisnumexpr{*Code*}{*Dann-Teil*}{*Sonst-Teil*}

v3.28 Wenn die Expansion von *Code* in einer \numexpr, also einem $\varepsilon$-TeX-Zahlenausdruck resultiert, wird der *Dann-Teil* ausgeführt, anderenfalls der *Sonst-Teil*. Es ist zu beachten, dass fehlerhafte Ausdrücke zu Fehlermeldungen führen. Die Anweisung ist nicht voll expandierbar.

\IfActiveMkBoth{*Dann-Teil*}{*Sonst-Teil*}

v3.27

Bei der Umschaltung zwischen automatischen und manuellen Kolumnentiteln, verwendet der LaTeX-Kern die Anweisung \@mkboth. Im Falle automatischer Kolumnentitel setzt diese normalerweise sowohl den linken als auch rechten Markenanteil. Im Falle manueller Kolumnentiteln setzt sie keine Marke. Will man wissen, ob \@mkboth Marken setzt oder nicht, so verwenden viele Pakete einen Vergleich entweder mit \markboth oder \@gobbletwo. Dies deckt aber nicht alle Fälle von möglichen Umdefinierungen von \@mkboth ab. Daher testet \IfActiveMkBoth, ob \@mkboth tatsächlich zum Setzen einer Marke führen würde und deckt dabei sogar die Verwendung von \marks für die Definition von \@mkboth ab. Wird ein solches aktives \@mkboth entdeckt, so wird der *Dann-Teil* ausgeführt. In allen anderen Fällen wird der *Sonst-Teil* ausgeführt.

**Beispiel:** Angenommen, Sie wollen in einem Paket im Fall der Verwendung automatischer Kolumnentitel, wie sie beispielsweise der Seitenstil `headings` aktiviert, nur die rechte Marke setzen und die linke unverändert lassen, wohingegen bei Verwendung manueller Kolumentitel die Marken unverändert bleiben sollen. In einem ersten Ansatz verwenden Sie dafür:

```
\ifx\@mkboth\markboth \markright{Kolumnentitel}\fi
```

Etwas später entdecken Sie, dass irgendein Paket nicht wie gewohnt

```
\let\@mkboth\markboth
```

sondern

```
\renewcommand{\@mkboth}{\markboth}
```

verwendet hat, um automatische Kolumnentitel zu aktivieren. Daher ruft Ihr Vergleich niemals die \markright-Anweisung auf. Daher ändern Sie den Vergleich oben zu

```
\ifx\@mkboth\@gobbletwo
\else \markright{Kolumnentitel}\fi
```

Leider wird nun \markright auch bei manuellen Kolumentiteln aufgerufen, weil irgend jemand für diesen Fall

```
\renewcommand{\@mkboth}[2]{%
  \typeout{DEBUG: ignoring running head setting}%
}
```

definiert hat.

Beide Probleme sind jedoch mit Hilfe von scrbase einfach zu lösen:

```
\IfActiveMkBoth{\markright{Kolumnentitel}}{}
```

Noch einfacher lässt sich das Problem aus dem Beispiel mit Hilfe von \@mkright aus Paket scrlayer lösen (siehe Abschnitt 14.6, Seite 449).

`\if@atdocument` *Dann-Teil* `\else` *Sonst-Teil* `\fi`

Diese Verzweigung in TEX-Syntax existiert bewusst nur als interne Anweisung. In der Dokumentpräambel ist `\if@atdocument \iffalse`, nach `\begin{document}` entspricht `\if@atdocument` dann `\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. Ab LATEX 2020/10/01 wird `\@atdocumenttrue` von scrbase über den generischen LATEX-Hook `begindocument/before` ausgeführt. Daher ist der Zustand bereits für `\BeginDocument` für Klassen und alle Pakete `\iftrue`. Für Verwendung von `\if@atdocument` mit dem generischen LATEX-Hook `begindocument/before` wird hingegen gegebenenfalls eine zusätzliche Hook-Regel benötigt, um einen definierten Zustand zu erreichen.

v3.37

## 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 Pakets wie babel oder polyglossia diese Anweisungen von den Paketen erneut definiert werden. Bei den genannten Paketen geschieht dies, indem alle zuvor gemachten Änderungen überschrieben werden. Aus diesem Grund ist es sinnvoll, eigene Änderungen bis nach dem Laden entsprechender Pakete oder mit Hilfe von `\AtBeginDocument` bis `\begin{document}` zu verzögern.

Darüber hinaus kommt erschwerend hinzu, dass einige Pakete zusätzliche, sprachabhängige Begriffe in `\captions`*Sprache* definieren, während andere dafür `\extras`*Sprache* verwenden. So muss der Anwender sich schon sehr genau auskennen, um die richtige Anweisung auf die richtige Weise zu ergänzen.

Das Paket scrbase bietet dem Anwender daher für die Definition und Änderung selbst einige zusätzliche Anweisungen, die ihn von vielen dieser Überlegungen befreit. Gleichzeitig erlauben die Befehle, die sprachabhängigen Begriffe mehrerer Dialekte oder Ausprägungen einer Sprache gleichzeitig zu definieren oder zu ändern.

```
\defcaptionname{Sprachliste}{Begriff}{Inhalt}
\providecaptionname{Sprachliste}{Begriff}{Inhalt}
\newcaptionname{Sprachliste}{Begriff}{Inhalt}
\renewcaptionname{Sprachliste}{Begriff}{Inhalt}
\defcaptionname*{Sprachliste}{Begriff}{Inhalt}
\providecaptionname*{Sprachliste}{Begriff}{Inhalt}
\newcaptionname*{Sprachliste}{Begriff}{Inhalt}
\renewcaptionname*{Sprachliste}{Begriff}{Inhalt}
```

Mit Hilfe dieser Anweisungen ist es möglich, einem *Begriff* in Abhängigkeit der Sprache einen *Inhalt* zuzuweisen. Mehrere Sprachen können durch Komma voneinander getrennt als *Sprachliste* angegeben werden. Der *Begriff* ist immer ein Makro. Die Arbeitsweise der Anweisungen unterscheidet sich je nachdem, ob eine Sprache und ein *Begriff* innerhalb der Sprache zum Zeitpunkt des Aufrufs bereits definiert ist.

v3.12

Ist eine Sprache nicht definiert, vermerkt `\providecaptionname` das lediglich in `log`-Datei. 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 ebenfalls ein entsprechender Hinweis in die `log`-Datei geschrieben.

Ist dagegen bei `\newcaptionname` eine 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.

Ist bei Anweisung `\renewcaptionname` eine 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.

v3.12

Die Anweisung `\defcaptionname` definiert einen *Begriff* immer, überschreibt also eventuell vorhandene Definitionen. Wie bei `\providecaptionname` braucht eine angegebene Sprache nicht definiert zu sein.

KOMA-Script selbst verwendet `\providecaptionname` um beispielsweise die Begriffe aus Abschnitt 19.4, Seite 520 zu definieren.

**Beispiel:** Möchten Sie »Abb.« statt »Abbildung« in den Abbildungsunterschriften, so erreichen Sie dies mit:

```
\renewcaptionname{ngerman}{\figurename}{Abb.}
```

Soll dieselbe Änderung nicht nur für `ngerman`, sondern auch für die Sprachen `naustrian` und `nswissgerman`, also für Österreichisch und Schweizer Deutsch gelten,

so ist es nicht notwendig zwei weitere Anweisungen:

```
\renewcaptionname{naustrian}{\figurename}{Abb.}
\renewcaptionname{nswissgerman}{\figurename}{Abb.}
```

hinzuzufügen. Stattdessen kann einfach, die *Sprachliste* erweitert werden:

```
\renewcaptionname{ngerman,naustrian,nswissgerman}%
                 {\figurename}{Abb.}
```

In gleicher Weise können auch `german`, `austrian` und `swissgerman`, also Deutsch, Österreichisch und Schweizer Deutsch nach der veralteten Rechtschreibung, hinzugefügt werden.

v3.12 Es ist nicht erforderlich, die Definierung oder Umdefinierung via `\AtBeginDocument` bis `\begin{document}` zu verzögern. Stattdessen erledigt scrbase das selbst, falls die Anweisungen in der Dokumentpräambel aufgerufen werden. Außerdem prüft scrbase auch, ob ein umzudefinierender Begriff statt in `\captions`*Sprache* in `\extras`*Sprache* zu definieren ist. Die Sternvarianten der Befehle verwenden grundsätzlich `\extras`*Sprache*, da dessen Definitionen in der Regel nach `\captions`*Sprache* Anwendung finden. Damit funktioniert auch das Umdefinieren von sprachabhängigen Bezeichnern von Paketen, die dafür `\extras`*Sprache* 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

...

Tabelle 10.1.: Überblick über übliche sprachabhängige Begriffe (*Fortsetzung*)

`\contentsname`
: Überschrift für das Inhaltsverzeichnis

`\enclname`
: Spitzmarke für die Anlagen bei Briefen

`\figurename`
: Spitzmarke in der Abbildungsunterschrift

`\glossaryname`
: Überschrift für das Glossar

`\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

Obwohl – oder gerade weil – scrbase ganz allgemein als Paket für Klassen- und Paketautoren konzipiert ist, wird es natürlich von den KOMA-Script-Klassen und den meisten KOMA-Script-Paketen verwendet. Es enthält daher auch eine Anweisung, die in allen KOMA-Script-Klassen und allen grundlegenden KOMA-Script-Paketen vorhanden ist.

`\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 »2026/10/19 v3.49.3 KOMA-Script« erstellt. Das Vorhandensein der Anweisung ist ein starkes Indiz dafür, dass zumindest ein KOMA-Script-Paket verwendet wird.

## 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 naheliegenden Anweisungen `\ClassInfoNoLine` und `\PackageInfoNoLine`, um auch Informationen ohne Zeilennummer in die Log-Datei zu schreiben, fehlen jedoch. Diese werden von scrbase bereitgestellt.

`\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. Das funktioniert in dieser Form nur für Makros ohne Argumente. Dennoch könnte man die Anweisung in einigen Fällen auch in einer Form benötigen, die lokal zur aktuellen Gruppe arbeitet. Diese wird mit `\l@addto@macro` von scrbase bereitgestellt. Eine Alternative stellt hier die Verwendung des Pakets etoolbox oder xpatch dar, die eine ganze Reihe solcher Anweisungen für unterschiedliche Zwecke bieten (siehe [Leh11] oder [Gre12]).

`\ExpandArgsCS{`*Spezifikation*`}{`*Anweisungsname*`}{`*Argument*$_i$`}...`

v3.49.3

Seit Version 2022-06-01 bietet der LaTeX-Kern eine Anweisung `\ExpandArgs`, mit deren Hilfe man die an eine Anweisung übergebenen Argumente entsprechend einer *Spezifikation* expandieren kann. In der L3-Programmierebene entspricht dies den Funktionen `\exp_args:N...`. Während die L3-Programmierebene auch Funktionen `\exp_args:c...` bereitstellt, mit deren Hilfe man dieselbe Funktionalität auch nutzen kann, wenn die Anweisung noch nicht als Makro, sondern nur als *Anweisungsname* – in der L3-Sprechweise als *csname* bezeichnet – verfügbar ist, fehlt eine Entsprechung auf LaTeX 2$\varepsilon$-Ebene. `\ExpandArgsCS` ist diese Entsprechung. Der Aufruf `\ExpandArgsCS{`*Anweisungsname*`}{`*Spezifikation*`}...` entspricht also `\ExpandArg\`*Anweisungsname*`{`*Spezifikation*`}...`. Näheres zu `\ExpandArgs` ist [Tea26] zu entnehmen. Die Dokumentation von `\exp_args:...` ist in [Pro26] zu finden.

## 10.7. Erweiterungen der mathematischen Fähigkeiten von $\varepsilon$-TeX

Das für LaTeX inzwischen verwendete und von KOMA-Script vorausgesetzte $\varepsilon$-TeX besitzt mit `\numexpr` erweiterte Möglichkeiten zur Berechnung einfacher Ausdrücke mit TeX-Zählern und ganzen Zahlen. Als Operationen werden dabei die vier Grundrechenarten und Klammern unterstützt. Bei der Division wird korrekt gerundet.

`\XdivY{`*Dividend*`}{`*Divisor*`}`
`\XmodY{`*Dividend*`}{`*Divisor*`}`

v3.05a

Die Anweisung `\XdivY` liefert den Wert des ganzzahligen Quotienten, die Anweisung `\XmodY` den Wert des Rests der Division mit Rest. Diese Art der Division ist nach der Gleichung

$$\mathit{Dividend} = \mathit{Divisor} \cdot \mathit{Quotient} + \mathit{Rest}$$

definiert, wobei *Dividend*, *Divisor* 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` voranzustellen.

v3.27

## 10.8. Mechanismus für mehrstufige Haken

Der LaTeX-Kern bietet selbst bereits einige wenige Stellen in der Verarbeitung eines Dokuments, an denen die Ausführung zusätzlichen Codes *eingehakt* werden kann. Klassen- und Paketautoren dürften `\AtBeginDocument` und `\AtEndDocument` bestens bekannt sein. KOMA-Script bietet an einigen Stellen vergleichbares, beispielsweise um Code in die Ausführung von Gliederungsbefehlen einzuhaken. Über die Jahre zeigten sich dabei zwei Probleme:

- Es gibt nie genug Haken.
- Es gibt sowohl Code, der nur einmal ausgeführt werden soll, also quasi bei seiner Ausführung wieder vom Haken fällt, als auch Code, der bei jedem Durchlaufen des Hakens auszuführen ist, also dauerhaft auf dem Haken bleibt.

Üblicherweise muss man zur Definition eines einzelnen Hakens eine Anweisung definieren, mit der Code für den jeweiligen Haken aufgesammelt wird. Der Code wird dazu in einem weiteren internen Makro abgelegt, das man an der Stelle einfügen muss, an der der aufgesammelte Code auszuführen ist. Je mehr Haken man einfügt, desto mehr solcher Anweisungen gibt es. Um sowohl Einmalcode als auch dauerhaften Code zu ermöglichen, bedarf es sogar zweier Haken und damit der doppelten Anzahl an zu definierenden Anweisungen.

Das Beispiel der Gliederungsbefehle zeigt, dass sich das erste Problem an einem einzigen Codeausführungspunkt noch verschärfen kann. Der eine Paketautor benötigt nur eine Möglichkeit, für alle Gliederungsbefehle denselben Code auszuführen. Ein anderer Paketautor hätte lieber, dass er bei bestimmten Gliederungsbefehlen unterschiedlichen Code ausführen kann. Es würde also sowohl ein allgemeiner Haken als auch ein Haken je Gliederungsbefehl benötigt. Das ganze dann wieder wegen des zweiten Problems verdoppelt.

KOMA-Script bietet in scrbase daher einen verallgemeinerten Hakenmechanismus, der mehrstufige Haken sowohl für Einmalcode als auch für dauerhaften Code bereitstellt. Diese Haken haben aufgrund der Implementierung den Namen *do-hook* erhalten. Davon leitet sich auch der Name der Anweisungen ab, mit denen sie gesteuert werden.

`\ExecuteDoHook{`*`Spezifikator`*`}`

v3.27

Über diese Anweisung werden Haken implementiert. Dabei bestimmt der *`Spezifikator`* den Namen des oder genauer der Haken. Der *`Spezifikator`* wird zur Analyse immer vollständig expandiert.

Der *`Spezifikator`* besteht allgemein aus durch Querstrichen, »`/`«, voneinander getrennten Zeichenfolgen. Von `\ExecuteDoHook` wird zunächst die Zeichenfolge vor dem ersten Querstrich abgetrennt. Diese ist der Name. Der verbleibende Rest ohne den führenden Querstrich ist das Argument. Dann wird der Code für den Haken mit diesem Namen ausgeführt. Anschließend wird erneut die vordere Zeichenfolge vom Rest abgetrennt und mit einem Querstrich hinten an den Namen angefügt und der Code für den so neu gebildeten Namen ausgeführt. Das geht

so lange, bis der Code für den Haken mit dem Namen *Spezifikator* und leerem Argument ausgeführt wurde.

Im einfachsten Fall besteht der *Spezifikator* aus einem einzelnen Namen. In diesem Fall wird zunächst der dauerhafte Code für genau einen durch diesen Namen festgelegten Haken mit leerem Argument ausgeführt.

In jeder Stufe der Ausführung wird nach dem dauerhaften Code für einen Haken zusätzlich Einmalcode ausgeführt, bevor der Einmalcode global vom Haken genommen wird.

**Beispiel:** Mit `\ExecuteDoHook{heading/begingroup/`*Name*`}` in der Ausführung jedes durch `\DeclareSectionCommand` definierten Gliederungbefehls wurden in den KOMA-Script-Klassen letztlich sechs Haken an diesem Codepunkt eingefügt, die in dieser Reihenfolge ausgeführt werden:

1. `heading` mit Argument `begingroup/`*Name* für dauerhaften Code,
2. `heading` mit Argument `begingroup/`*Name* für Einmalcode,
3. `heading/begingroup` mit Argument *Name* für dauerhaften Code,
4. `heading/begingroup` mit Argument *Name* für Einmalcode,
5. `heading/begingroup/`*Name* mit leerem Argument für dauerhaften Code,
6. `heading/begingroup/`*Name* mit leerem Argument für Einmalcode.

Der Platzhalter *Name* ist dabei identisch mit dem bei `\DeclareSectionCommand`, `\DeclareNewSectionCommand` oder `\ProvideSectionCommand` angegebenen Namen der Gliederungsebene beziehungsweise des dadurch bestimmten Gliederungsbefehls, also beispielsweise `subparagraph` oder `chapter`. Betrachtet man obige Auflistung unter Berücksichtigung der Vielzahl an Gliederungsbefehlen, wird klar, dass die Haken `heading` und `heading/begingroup` mehrfach, nämlich für jeden Gliederungsbefehl existieren.

`\AddtoDoHook{`*Name*`}{`*Befehl*`}`
`\AddtoOneTimeDoHook{`*Name*`}{`*Befehl*`}`

v3.27

Mit `\AddtoDoHook` wird an dem Haken mit dem Namen *Name* dauerhafter Code aufgehängt. Als Code dient dabei der *Befehl*, dem das in der Erklärung zu `\ExecuteDoHook` erwähnte Argument als Parameter angehängt wird.

**Beispiel:** Angenommen, es soll innerhalb von `\section` mitgezählt werden, wie oft diese Anweisung ausgeführt wird. Dies wäre in Fortführung des obigen Beispiels mit

```
\newcounter{sectionZaehler}
\AddtoDoHook{heading/begingroup/section}
            {\stepcounter{sectionZaehler}}
```

sehr einfach möglich. Allerdings würde dann in Wirklichkeit `\stepcounter{sectionZaehler}{}` ausgeführt. Wir erinnern uns daran, dass immer ein Argument als Parameter angehängt wird. Im Falle des Hakens mit dem Namen `heading/begingroup/section` ist dieses Argument leer. Da ein solcher leerer Parameter hier zu einer leeren Gruppe wird, ist es besser, diesen leeren Parameter quasi aufzubrauchen:

```
\newcommand*{\stepcountergobble}[2]{%
  \stepcounter{#1}%
}
\AddtoDoHook{heading/begingroup/section}
            {\stepcountergobble{sectionZaehler}}
```

Somit wird der angehängte, leere Parameter von der neuen Anweisung `\stepcountergobble` gelesen, aber nicht verwendet.

Sollen nun statt `\section` alle Gliederungsbefehle gezählt werden, so muss nur ein anderer Hakenname eingesetzt werden:

```
\AddtoDoHook{heading/begingroup}
            {\stepcountergobble{sectionZaehler}}
```

Hier ist übrigens der angehängte Parameter nicht leer, sondern enthält den Namen der Gliederungsebene oder des Gliederungsbefehls. Wollte man die Gliederungsbefehle einzeln zählen, so könnte man genau dies ausnützen:

```
\newcommand*{\stepZaehler}[1]{%
  \stepcounter{#1Zaehler}%
}
\AddtoDoHook{heading/begingroup}
            {\stepZaehler}
```

Natürlich müssten dann auch die Zähler `partZaehler`, `chapterZaehler` bis hinunter zu `subparagraphZaehler` definiert werden.

Die Anweisung `\AddtoOneTimeDoHook` arbeitet vergleichbar, fügt den *Befehl* aber dem Einmalcode hinzu. Dieser Code wird dann nach der ersten Ausführung global vom Haken genommen.

`\ForDoHook{`*Spezifikator*`}{`*Befehl*`}`

v3.27

Während `\ExecuteDoHook` dazu gedacht ist, die Befehle auszuführen, die zuvor mit `\AddtoDoHook` oder `\AddtoOneTimeDoHook` für die durch den *Spezifikator* bestimmten Haken gespeichert wurden, führt diese Anweisung den unmittelbar angegebenen *Befehl* aus. Dabei werden an *Befehl* sogar zwei Parameter angefügt. Der erste Parameter ist der Name des Haken, der zweite das Argument des Hakens.

Diese Anweisung ist ein Abfallprodukt der Implementierung von `\ExecuteDoHook`. Normalerweise sollten weder Anwender noch Paketautoren diese Anweisung benötigen.

\SplitDoHook{*Spezifikator*}{*Kopf-Makro*}{*Rest-Makro*}

v3.27

Wie aus den vorangegangenen Erklärungen deutlich wird, kann der Parameter eines mit \AddtoDoHook oder \AddtoOneTimeDoHook hinzugefügten *Befehls* ebenfalls ein mehrteiliger *Spezifikator* sein. Mit \SplitDoHook kann so ein *Spezifikator* in das vordere Element und den Rest aufgeteilt werden. Das *Kopf-Makro* wird dabei als das vordere Element definiert. Das *Rest-Makro* wird als der Rest definiert. Im Falle, dass kein Rest bleibt, wird das *Rest-Makro* leer definiert. Falls bereits der *Spezifikator* leer war, wird eine Warnung ausgegeben und sowohl *Kopf-Makro* als auch *Rest-Makro* leer definiert.

**Beispiel:** Wollte man am Anfang der Gruppe, in der die Überschrift ausgegeben wird, einen Zähler erhöhen, diesen am Ende aber wieder verringern, so könnte man dies über zwei Haken tun:

```
\AddtoDoHook{heading/begingroup}
            {\stepZaehler}
\newcommand*{\restepZaehler}[1]{%
  \addtocounter{#1Zaehler}{-1}%
}
\AddtoDoHook{heading/endgroup}
            {\restepcounter}
```

Man könnte aber auch mit einem einzigen Haken arbeiten und dessen Parameter zerlegen:

```
\newcommand*{\changeZaehler}[1]{%
  \SplitDoHook{#1}{\Gruppe}{\Ebene}%
  \Ifstr{\Gruppe}{begingroup}{%
    \stepcounter{\Ebene Zaehler}%
  }{%
    \Ifstr{\Gruppe}{endgroup}{%
      \addtocounter{\Ebene Zaehler}{-1}%
    }{}%
  }%
}
\AddtoDoHook{heading}
            {\changeZaehler}
```

Wie zu sehen ist, ist die erste Lösung um einiges einfacher. Dazu kommt, dass man die zweite Fallunterscheidung in der zweiten Lösung gerne vergisst. Das wäre allerdings fatal, da es weitere Haken mit Namen `heading` aber abweichendem Argument geben kann.

Genau genommen ist diese Anweisung ein Abfallprodukt der Implementierung von \ForDoHook.

## 10.9. Obsolete Optionen und Anweisungen

Frühere Versionen von scrbase verfügten über Optionen und Anweisungen, die inzwischen entfernt oder ersetzt wurden. Diese sind hier nur aus Gründen der Vollständigkeit aufgeführt und sollten nicht mehr verwendet werden.

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

# 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.

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. Teilweise ist es auch notwendig, in irgendeiner Form auf das Laden eines anderen Pakets zu reagieren.

Nehmen wir als einfaches Beispiel das Laden des longtable-Pakets 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.

Ursprünglich bestand die einzige Lösungsmöglichkeit 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 Pakets gerne die Bedeutung eines Makros in einem Hilfsmakro retten und nach dem Laden des Pakets wieder restaurieren will. Auch das geht mit scrlfile.

Die Anwendung von scrlfile ist nicht auf die Abhängigkeit von Paketen beschränkt. Auch Abhängigkeiten von anderen Dateien, die mit `\input` oder `\InputIfFileExists` geladen werden, können berücksichtigt werden.

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.

`withdeprecated`

v3.32 Mit LaTeX 2020-10-01 steht ein komplett neuer *Hook*-Mechanismus zur Verfügung. Statt Makros des LaTeX-Kerns umzudefinieren, macht scrlfile ab Version 3.32 davon auch Gebrauch.
v3.35 Dazu lädt es das interne Paket scrlfile-hook. Bei älteren LaTeX-Versionen wird dagegen je nach Kernel-Version das interne Paket scrlfile-hook-3.34 beziehungsweise scrlfile-patcholdkernel verwendet. Da mit dem neuen Mechanismus in LaTeX jedoch keine identische Funktionalität zu erreichen war, wurden einige Anweisungen von scrlfile als veraltet markiert. Darüber hinaus wurde die Gelegenheit genutzt, den Wildwuchs an Befehlsvarianten zu ordnen. Sollte ein Anwender eine Klasse oder Pakete verwenden, die noch auf den dadurch nicht mehr unterstützten Befehlen basieren, so kann vor scrlfile vor dem Laden der Klasse mit

```
\RequirePackage[withdeprecated]{scrlfile}
```

geladen werden. scrlfile emuliert dann zusätzlich einige der veralteten Befehle in einer Weise, die in den meisten Fällen ausreichend sein sollte. In jedem Fall sollte jedoch der Autor der entsprechenden Klasse oder des entsprechenden Pakets verständigt werden, damit dieser eine Anpassung an die aktuelle Version von scrlfile oder direkt an LaTeX ab Version 2021-11-15 vornimmt.

## 11.1. 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`*`}[`*`Label`*`]{`*`Anweisungen`*`}`
`\AfterFile{`*`Datei`*`}[`*`Label`*`]{`*`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` als Teil des Dateinamens anzugeben.

Um die Funktionalität für LaTeX vor Version 2020-10-01 bereitstellen zu können, bedient sich scrlfile-patcholdlatex der bekannten LaTeX-Anweisung `\InputIfFileExists`. Diese wird hierzu umdefiniert. Falls die Anweisung nicht die erwartete Definition hat, gibt scrlfile-patcholdlatex eine Warnung aus. Dies geschieht für den Fall, dass die Anweisung bereits von einem inkompatiblen Paket umdefiniert wurde.

v3.32 Bei LaTeX ab Version 2020-10-01 wird von scrlfile-hook hingegen `\AddToHook{file/before/`*`Datei`*`}[`*`Label`*`]{`*`Anweisungen`*`}` beziehungsweise `\AddToHook{file/after/`*`Datei`*`}[`*`Label`*`]{`*`Anweisungen`*`}` verwendet. Näheres zur Bedeutung des optionalen Arguments *`Label`* ist der zugehörigen Anleitung zur LaTeX-Kern-Anweisung

\AddToHook zu entnehmen. Empfohlen wird in der Regel der Verzicht auf das optionale Argument. Bei älteren LaTeX-Versionen wird dieses ohnehin ignoriert.

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}[Label]{Anweisungen}
\BeforePackage{Paket}[Label]{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.

Es ist zu beachten, dass hier von scrlfile-hook ebenfalls file und nicht class respektive package für den Hook verwendet wird. Nur so ist sichergestellt, dass die Ausführung der *Anweisungen* bereits im Kontext der Klasse beziehungsweise des Pakets erfolgt. Zur Bedeutung des optionalen Arguments *label* sei wiederum auf die Anleitung der LaTeX-Kernanweisung \AddToHook verwiesen. Empfohlen wird in der Regel der Verzicht auf das optionale Argument. Bei älteren LaTeX-Versionen wird dieses ohnehin ignoriert.

```
\AfterClass{Klasse}[Label]{Anweisungen}
\AfterClass*{Klasse}[Label]{Anweisungen}
\AfterAtEndOfClass{Klasse}[Label]{Anweisungen}
\AfterAtEndOfClass*{Klasse}[Label]{Anweisungen}
\AfterPackage{Paket}[Label]{Anweisungen}
\AfterPackage*{Paket}[Label]{Anweisungen}
\AfterAtEndOfPackage{Paket}[Label]{Anweisungen}
\AfterAtEndOfPackage*{Paket}[Label]{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 \AfterClass* und \AfterPackage* 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 Code, dessen Ausführung per `\AtEndOfClass` oder `\AtEndOfPackage` innerhalb der Klasse respektive des Pakets verzögert wird, wird allerdings erst danach ausgeführt. Mit `\AfterAtEndOfClass` beziehungsweise `\AfterAtEndOfPackage` wird dagegen sichergestellt, dass *Anweisungen* erst nach solchem Code ausgeführt wird.

v3.32 Auch hierzu gibt es Sternvarianten, die dafür sorgen, dass *Anweisungen* sofort ausgeführt werden, falls die Klasse beziehungsweise das Paket bereits vollständig geladen ist. Wird eine Klasse oder ein Paket gerade geladen, so wird die Ausführung wie bei der Version ohne Stern verzögert.

Es ist zu beachten, dass scrlfile-hook für `\AfterClass`, `\AfterPackage` und deren Sternvarianten einen `file/after`-Hook verwendet. Dagegen setzen `\AfterAtEndOfClass` und dessen Sternvariante einen `class/after`-Hook und `\AfterAtEndOfPackage` und dessen Sternvariante entsprechend einen `package/after`-Hook. Zur Bedeutung des optionalen Arguments *label* sei wiederum auf die Anleitung der LaTeX-Kernanweisung `\AddToHook` verwiesen. Empfohlen wird in der Regel der Verzicht auf das optionale Argument. Bei älteren LaTeX-Versionen wird dieses ohnehin ignoriert.

**Beispiel:** Als Beispiel für Paket- oder Klassenautoren will ich zunächst erklären, wie KOMA-Script selbst Gebrauch von den neuen Anweisungen macht. Dazu findet sich beispielsweise in scrbook Folgendes:

```
\AfterPackage{hyperref}{%
  \@ifpackagelater{hyperref}{2001/02/19}{}{%
    \ClassWarningNoLine{scrbook}{%
      You are using an old version of the hyperref
      package!\MessageBreak%
      This version has a buggy hack in many
      drivers,\MessageBreak%
      causing \string\addchap\space to behave
      strangely.\MessageBreak%
      Please update hyperref to at least version
      6.71b}%
  }%
}
```

Alte Versionen von hyperref definierten ein Makro von scrbook in einer Weise um, die 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 Pakets überprüft wird, ob es sich um eine verträgliche Version handelt. Falls dies nicht der Fall ist, wird eine Warnung ausgegeben.

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 von hyperref automatisch getroffen.

Alles weitere kann nun `\AfterFile` überlassen werden:

```
\documentclass{article}
\usepackage[ngerman]{babel}
\usepackage{scrlfile}
\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 mehrere Zeile
    umfasst 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` zur Anwendung kommt, wird die dann nützliche Einstellung `linktocpage` verwendet. Wenn Sie jedoch mit pdfLaTeX eine PDF-Datei erstellen, wird darauf verzichtet, da dann der hyperref-Treiber `hpdftex.def` verwendet wird. Das bedeutet, dass weder die Treiberdatei `hdvips.def` noch `hypertex.def` geladen wird.

Übrigens kann scrlfile auch bereits vor `\documentclass` geladen werden. In diesem Fall ist allerdings `\RequirePackage` anstelle von `\usepackage` zu verwenden (siehe [Tea06]).

\BeforeClosingMainAux{*Anweisungen*}
\AfterReadingMainAux{*Anweisungen*}

Paketautoren haben des Öfteren das Problem, dass sie Anweisungen in die aux-Datei schreiben wollen, nachdem die letzte Seite des Dokuments ausgegeben wurde. Dazu wird – in Unkenntnis der dadurch verursachten Probleme – häufig Code wie der folgende eingesetzt:

```
\AtEndDocument{%
  \if@filesw
    \write\@auxout{\protect\writethistoaux}%
  \fi
}
```

Dies ist jedoch keine wirkliche Lösung. Wurde die letzte Seite vor \end{document} bereits ausgegeben, so führt obiges zu keiner Ausgabe in die aux-Datei. 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 auch 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 die aux-Datei geschrieben.

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
}
```

Das ist auch dann erfolgreich, wenn das finale \clearpage innerhalb von \end{document} tatsächlich zu keiner Ausgabe einer Seite mehr führt oder wenn – sei es korrekt verwendet oder in Unkenntnis der oben erläuterten Probleme – \clearpage innerhalb einer \AtEndDocument-Anweisung zum Einsatz kam.

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 unvorhersehbar 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.

## 11.2. 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.

v3.32 Bei Verwendung eines LaTeX-Version bis 2020-04-01 muss dass Paket scrlfile-patcholdlatex dazu die Anweisung \InputIfFileExists umdefinieren. Bei Verwendung von LaTeX ab Version 2020-10-01 wird hingegen von scrlfile-hook die interne LaTeX-Anweisung \declare@file@substitution verwendet. Das LaTeX-Team bittet darum, eine derartige Dateiersetzung nur vorzunehmen, wenn es keine andere Möglichkeit gibt, zu dem gewünschten Ergebnis zu gelangen, beispielsweise wenn eine solche Ersetzung zum Erhalt der Kompatibilität unabdingbar ist und dabei die *Ersatzdatei* die gleiche Funktionalität bereitstellt.

**Beispiel:** Sie wollen, dass anstelle der Datei \jobname.lot, die Datei \jobname.tol geladen wird. Dazu verwenden Sie:

```
\ReplaceInput{\jobname.lot}{\jobname.tol}
```

Wenn Sie nun zusätzlich \jobname.tol auch noch durch \jobname.tlo ersetzen:

```
\ReplaceInput{\jobname.tol}{\jobname.tlo}
```

dann wird `\jobname.lot` am Ende durch `\jobname.tlo` ersetzt. Es wird also die komplette Ersetzungskette abgearbeitet.

Einer Ersetzung im Kreis:

```
\ReplaceInput{\jobname.lot}{\jobname.tol}
\ReplaceInput{\jobname.tol}{\jobname.lot}
```

würde jedoch zu einem Fehler 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. Es wird jedoch empfohlen, für das Ersetzen eines Pakets oder einer Klasse die nachfolgenden Anweisungen zu verwenden. Bei älteren LaTeX-Versionen ist dies sogar zwingend.

```
\ReplaceClass{Klasse}{Ersatzklasse}
\ReplacePackage{Paket}{Ersatzpaket}
```

v2.96 Eine Klasse oder ein Paket sollte niemals mit Hilfe der oben erklärten Anweisung `\ReplaceInput` ersetzt werden. Stattdessen sollte für Paketersetzungen `\ReplacePackage` und für Klassenersetzungen `\ReplaceClass` verwendet werden. Es ist zu beachten, dass 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, die eine geforderte Option nicht unterstützt, würde das zu den üblichen Warnungen und Fehlern führen.

Damit Klassen mit `\ReplaceClass` ersetzt werden können, ist es natürlich erforderlich scrlfile vor der Klasse zu laden. Dazu ist `\RequirePackage` anstelle von `\usepackage` zu verwenden (siehe [Tea06]).

v3.32 Bei Verwendung eines LaTeX-Version bis 2020-04-01 muss dass Paket scrlfile-patcholdlatex die internen Anweisung `\@onefilewithoptions` und `\@loadwithoptions` umdefinieren. Bei Verwendung von LaTeX ab Version 2020-10-01 wird hingegen von scrlfile-hook die interne LaTeX-Anweisung `\declare@file@substitution` verwendet. Das LaTeX-Team bittet darum, eine derartige Ersetzung nur vorzunehmen, wenn es keine andere Möglichkeit gibt, zu dem gewünschten Ergebnis zu gelangen, beispielsweise wenn eine solche Ersetzung zum Erhalt der

Kompatibilität unabdingbar ist und dabei die *Ersatzpaket* beziehungsweise *Ersatzklasse* die gleiche Funktionalität bereitstellt.

```
\UnReplaceInput{Datei}
\UnReplacePackage{Paket}
\UnReplaceClass{Klasse}
```

v3.12 Eine Ersetzung kann auch wieder aufgehoben werden. Dabei sollten Ersetzungen von Dateien immer mit `\UnReplaceInput`, Ersetzungen von Paketen mit `\UnReplacePackage` und Ersetzungen von Klassen mit `\UnReplaceClass` aufgehoben werden. Nach der Aufhebung der Ersetzung führen Ladebefehle für die entsprechende *Datei*, das entsprechende *Paket* oder die entsprechende *Klasse* dann wieder dazu, dass die *Datei*, das *Paket* oder die *Klasse* selbst anstelle der Ersatzdatei, des Ersatzpakets oder der Ersatzklasse geladen wird.

## 11.3. 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 oft verwenden. Wenn es dann mit einem dieser automatisch geladenen Pakete zu einem Problem kommt, muss man irgendwie das Laden des problematischen Pakets verhindern. Auch hier bietet scrlfile eine einfache Lösung.

```
\PreventPackageFromLoading[Stattdessencode]{Paketliste}
\PreventPackageFromLoading*[Stattdessencode]{Paketliste}
```

v3.08 Wird diese Anweisung vor dem Laden eines Pakets 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 beim hierbei empfohlenen Paket fontspec 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 (siehe [Tea06]).

Wird eine leere *Paketliste* angegeben oder wird ein Paket angegeben, das bereits geladen ist, gibt `\PreventPackageFromLoading` eine Warnung aus, während
v3.12 `\PreventPackageFromLoading*` lediglich einen entsprechenden Hinweis in die Log-Datei schreibt.

v3.12 Das optionale Argument kann verwendet werden, wenn anstelle des Ladens des Pakets etwas anderes getan werden soll. Innerhalb des *Stattdessencode*s dürfen jedoch keine anderen Pakete und keine Dateien geladen werden. Zum Laden eines anderen Pakets siehe `\ReplacePackage` in Abschnitt 11.2 auf Seite 361. Beachten Sie bitte auch, dass der *Stattdessencode* mehrfach ausgeführt wird, falls Sie versuchen, das Paket mehrfach zu laden!

`\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 dem 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.

Beachten Sie bitte, dass `\ResetPreventPackageFromLoading` den *Stattdessencode* für ein Paket nicht löscht, sondern nur vorübergehend dessen Ausführung nicht mehr erfolgt.

`\UnPreventPackageFromLoading{`*Paketliste*`}`
`\UnPreventPackageFromLoading*{`*Paketliste*`}`

v3.12 Statt die Liste der Pakete, für die das Laden verhindert werden soll, komplett zurückzusetzen, kann man auch einzelne oder mehrere Pakete gezielt von dieser Liste entfernen. Die Sternvariante des Befehls löscht außerdem den *Stattdessencode*, der für das Paket gespeichert ist. Falls die Verhinderungsliste beispielsweise aus einer gespeicherten Liste wiederhergestellt wird, wird dann der *Stattdessencode* trotzdem nicht mehr ausgeführt.

**Beispiel:** Angenommen, Sie wollen zwar verhindern, dass ein Paket foo geladen wird, wollen aber nicht, dass ein eventuell bereits gespeicherter *Stattdessencode* ausgeführt wird. Stattdessen soll nur Ihr neuer *Stattdessencode* ausgeführt werden. Dies ist wie folgt möglich:

```
\UnPreventPackageFromLoading*{foo}
\PreventPackageFromLoading[%
  \typeout{Stattdessencode}%
]{foo}
```

Für die Anweisung `\UnPreventPackageFromLoading*` ist es unerheblich, ob das Paket zuvor überhaupt vom Laden ausgenommen war.

Natürlich können Sie die Anweisung indirekt auch nutzen, um den *Stattdessencode* aller Pakete zu löschen:

```
\StorePreventPackageFromLoading\TheWholePreventList
\UnPreventPackageFromLoading*{\TheWholePreventList}
\PreventPackageFromLoading{\TheWholePreventList}
```

Die Pakete werden dann zwar noch immer nicht geladen, ihr *Stattdessencode* existiert aber nicht mehr und wird nicht mehr ausgeführt.

# Dateien mit scrwfile sparen und ersetzen

Von Version 3.10 bis 3.38 war das Paket scrwfile Bestandteil von KOMA-Script. Danach wurde es als eigenständiges Paket aus KOMA-Script herausgelöst. Das erschien und erscheint noch immer sinnvoll, weil das Paket immer experimentell und ohne offizielle Unterstützung war. Darüber hinaus lautet schon seit einiger Zeit die Empfehlung, stattdessen LuaLaTeX zu verwenden, da mit LuaLaTeX das Grundproblem:

```
! No room for a new \write .
\ch@ck ...\else \errmessage {No room for a new #3}
                                                \fi
```

nicht mehr existiert.

Sollten Sie das Paket dennoch verwendet haben und weiterhin benötigen, so kann es getrennt installiert wird. Verwenden Sie dazu vorzugsweise den Paketmanager Ihrer TeX-Distribution. Näheres zu dem Paket finden Sie im CTAN-Katalogeintrag [Koh23c].

# 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 [BB13]) 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.

Als kleiner Nebeneffekt kann das Paket auch verwendet werden, um neue Gleitumgebungen oder den Gleitumgebungen ähnliche nicht gleitende Umgebungen für Konsultationsobjekte zu definieren. Näheres dazu wird nach der Erklärung der grundlegenden Anweisungen in den folgenden vier Abschnitten durch ein Beispiel in Abschnitt 13.5 verdeutlicht, das in kompakter Form noch einmal in Abschnitt 13.6 aufgegriffen wird.

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 `\addxcontentsline` vorgenommen. Darüber hinaus gibt es Anweisungen, mit denen Aktionen 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 die Bezeichnung einer Kategorie sein, die der Autor der Klasse oder des Pakets, das tocbasic verwendet, frei gewählt hat. KOMA-Script selbst verwendet beispielsweise die Kategorie `float` für die Dateierweiterungen `lof` und `lot`, die für das Abbildungs- und das Tabellenverzeichnis stehen. Für das Inhaltsverzeichnis verwendet KOMA-Script als Besitzer den Dateinamen der Klasse.

`\Ifattoclist{`*`Dateierweiterung`*`}{`*`Dann-Teil`*`}{`*`Sonst-Teil`*`}`

v3.28

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 eine 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* der Liste der bekannten Dateierweiterungen hinzu. 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 weggelassen, 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 `\addtotoclist` erst später aufgrund der Verwendung einer Anweisung durch den Benutzer aufgerufen wird. In diesem Fall wird ein leerer *Besitzer* eingetragen.

Beachten Sie, dass ein leeres Argument *Besitzer* nicht immer das Gleiche ist wie das Weglassen des kompletten optionalen Arguments einschließlich der eckigen Klammern. Ein leeres Argument würde immer einen leeren *Besitzer* ergeben.

**Beispiel:** Angenommen, Sie wollen die Dateierweiterung »`foo`« 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`« der Liste der bekannten Dateierweiterung hinzu, wenn diese Erweiterung nicht bereits in der Liste ist. Wenn die verwendete Klasse oder ein anderes Paket diese Dateierweiterung schon 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 Eingabe-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 die Kategorie »float« anstelle 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. Das liegt daran, dass KOMA-Script die Dateierweiterungen »lof« für das Abbildungsverzeichnis und »lot« für das Tabellenverzeichnis mit der Kategorie »float« als *Besitzer* anmeldet und die Optionen für diesen Besitzer setzt.

Das Paket scrhackscrhack enthält übrigens Patches für mehrere Pakete wie float oder floatrow, die eigene Verzeichnisse bereitstellen. Bei Verwendung von scrhack wird unter anderem die jeweilige Dateierweiterung der Liste der bekannten Dateierweiterungen hinzugefügt. Dabei wird als *Besitzer* »float« verwendet. Dies ist sozusagen der grundlegende Baustein, um die Möglichkeiten von tocbasic und der KOMA-Script-Klassen auch für diese Verzeichnisse automatisch nutzen zu können. Siehe [Koh23a] für weitere Informationen.

\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 Arguments 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 13.4, Seite 394.

`\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. Das gilt auch für den leeren *Besitzer*. Wird dagegen gar kein `[`*Besitzer*`]` angegeben, entfallen also auch die eckigen Klammern, findet kein Besitzertest statt, sondern die *Dateierweiterung* wird unabhängig vom Besitzer entfernt.

`\doforeachtocfile[`*Besitzer*`]{`*Anweisungen*`}`

Bisher haben Sie nur Anweisungen kennengelernt, 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 mit 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. Sollen hingegen nur die Dateierweiterungen des Besitzers »`foo`« ausgegeben werden, geht das einfach mit:

```
\doforeachtocfile[foo]{\typeout{\@currext}}
```

Die KOMA-Script-Klassen scrbook und scrreprt verwenden die Anweisung, um für Verzeichnisse, für die Eigenschaft `chapteratlist` gesetzt ist, optional einen vertikalen Abstand oder die Kapitelüberschrift in das Verzeichnis einzutragen. Wie Sie diese Eigenschaft setzen können, ist in Abschnitt 13.2 ab Seite 375 zu finden.

`\tocbasicautomode`

Diese Anweisung definiert das vom LaTeX-Kern für Klassen- und Paketautoren bereitgestellte `\@starttoc` 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` anstelle von `\@starttoc` verwendet. Näheres zu `\tocbasic@starttoc` und `\@starttoc` ist Abschnitt 13.4, Seite 394 zu entnehmen.

Mit Hilfe von `\tocbasicautomode` wird also jedes Verzeichnis, das mit Hilfe von `\@starttoc` erstellt wird, automatisch 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 verwaltet werden kann und wie automatisch Anweisungen beim Hinzufügen von Dateierweiterungen zu dieser Liste ausgeführt werden können. Des Weiteren haben Sie eine Anweisung kennengelernt, mit der man für jede einzelne bekannte Dateierweiterung oder einen spezifischen Teil davon Anweisungen ausführen kann. In diesem Abschnitt werden Sie Anweisungen kennenlernen, 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 mit dem 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, sind wie bei \addtocontents mit \protect zu schützen.

\addxcontentsline{*Dateierweiterung*}{*Ebene*}[*Gliederungsnummer*]{*Inhalt*}

v3.12

Diese Anweisung ähnelt sehr der Anweisung \addcontentsline aus dem LaTeX-Kern. Allerdings besitzt sie ein zusätzliches optionales Argument für die *Gliederungsnummer* des Eintrags, während diese bei \addcontentsline im Argument *Inhalt* mit angegeben wird. Sie wird verwendet, um nummerierte oder nicht nummerierte Einträge in das über die *Dateierweiterung* spezifizierte Verzeichnis aufzunehmen. Dabei ist *Ebene* der symbolische Name der Gliederungsebene und *Inhalt* der entsprechende Eintrag. Die Seitenzahl wird automatisch bestimmt.

Im Unterschied zu \addcontentsline testet \addxcontentsline zunächst, ob Anweisung \add*Ebene*_*Dateierweiterung*entry definiert ist. In diesem Fall wird sie für den Eintrag verwendet, wobei *Gliederungsnummer* als optionales Argument und *Inhalt* als obligatorisches Argument übergeben wird. Ein Beispiel für eine solche Anweisung, die von den KOMA-Script-Klassen bereitgestellt wird, wäre \addparttocentry (siehe Abschnitt 18.4, Seite 484). Ist die entsprechende Anweisung nicht definiert, wird stattdessen die interne Anweisung \tocbasic@addxcontentsline verwendet. Diese erhält alle vier Argumente als obligatorische Argumente und verwendet dann ihrerseits \addcontentsline, um den gewünschten Eintrag vorzunehmen. Näheres zu \tocbasic@addxcontentsline ist Abschnitt 13.4, Seite 396 zu entnehmen.

Ein Vorteil der Verwendung von \addxcontentsline gegenüber \addcontentsline ist zum einen, dass die Eigenschaft numberline (siehe Seite 375) beachtet wird. Zum anderen kann die Form der Einträge über die Definition entsprechender, für die *Ebene* und *Dateierweiterung* spezifischer Anweisungen konfiguriert werden.

```
\addxcontentslinetoeachtocfile[Besitzer]{Ebene}[Gliederungsnummer]{Inhalt}
\addcontentslinetoeachtocfile[Besitzer]{Ebene}{Inhalt}
```

Diese beiden Anweisungen stehen in direkter Beziehung zu dem oben erklärten v3.12 \addxcontentsline beziehungsweise zum im LaTeX-Kern definierten \addcontentsline. Der Unterschied besteht darin, dass diese Anweisungen *Inhalt* nicht nur in eine einzelne Datei, sondern in alle Dateien eines angegebenen *Besitzers* und bei Verzicht auf das erste 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.

```
\addxcontentslinetoeachtocfile{chapter}%
                              [\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.

v3.12 Die Anweisung \addxcontentslinetoeachtocfile ist gegenüber \addcontentslinetoeachtocfile möglichst vorzuziehen, da die Erweiterungen von \addxcontentsline nur damit Anwendung finden. Näheres zu diesen Erweiterungen und Vorteilen ist in der vorausgehenden Erklärung von \addxcontentsline zu finden.

```
\listoftoc[Titel]{Dateierweiterung}
\listoftoc*{Dateierweiterung}
\listofeachtoc[Besitzer]
\listofDateierweiterungname
```

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 das Inhaltsverzeichnis und den 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 mit dem angegebenen Besitzer oder alle Verzeichnisse aller bekannten Dateinamenerweiterungen aus. Damit dabei der korrekte Titel ausgegeben werden kann, sollte \listof*Dateierweiterung*name passend definiert sein. Da eventuell auch der Anwender selbst \listoftoc ohne optionales Argument oder \listofeachtoc verwenden könnte, wird dies ohnehin empfohlen.

**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}
```

Im 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 375 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 Laden von Paket scrwfile [Koh23c] oder die Verwendung von LuaLaTeX bieten.

Das Paket scrhack enthält übrigens Patches für mehrere Pakete wie float oder floatrow, damit deren Verzeichnisbefehle \listoftoc verwenden. Dadurch stehen viele Möglichkeiten

von tocbasic und den KOMA-Script-Klassen auch für deren Verzeichnisse zur Verfügung. Für weitere Informationen siehe [Koh23a].

\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 eine solche Ausführung 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.

\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, 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 nicht bereits definiert ist. Sie kann also in einer Klasse in der gewünschten Weise vorbelegt werden und wird dann von tocbasic nicht umdefiniert, sondern wie vorgefunden verwendet.

**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. Zum anderen 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 absolute Experten ist und fast immer Handarbeit bedeutet, wird Laien empfohlen, darauf zu verzichten oder ihn nur vorsichtig und nicht an so exponierter Stelle wie dem 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 für dieses Argument der Titel des Verzeichnisses übergeben.

Die *`Definition`* ist dann selbstverständlich auch für die Auswertung weiterer Eigenschaften, die sich auf die Überschrift beziehen, verantwortlich. Das gilt insbesondere für die nachfolgend erklärten Eigenschaften `leveldown`, `numbered` und `totoc`.

`\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 und gelöscht werden. Die *`Liste von Eigenschaften`* ist dabei eine durch Komma getrennte Folge 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 nächsttieferen 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 mit dem Besitzer `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 vom Paket scrwfile [Koh23c] für Klonziele verwendet, da die Erweiterungen dort bereits durch das Klonen selbst aus der Klonquelle übernommen werden.

Es ist zu beachten, dass die Eigenschaft bereits vor dem Hinzufügen der Dateierweiterung zu der Liste der bekannten Dateierweiterungen gesetzt sein muss, damit sie eine Wirkung hat.

v3.27 `noindent` veranlasst alle von KOMA-Script bereitgestellten Verzeichniseintragsstile ihre Eigenschaft `indent` (siehe Tabelle 13.1, Seite 385) zu ignorieren und stattdessen den Einzug zu deaktivieren.

v3.17 `noparskipfake` verhindert, dass vor dem Abschalten des Absatzabstandes für die Verzeichnisse ein letztes Mal ein expliziter Absatzabstand eingefügt wird. Dies führt in der Regel dazu, dass bei Dokumenten mit Absatzabstand der Abstand zwischen Überschrift und erstem Verzeichniseintrag kleiner wird als zwischen Überschriften und normalem Text. Ohne diese Eigenschaft wirkt die Formatierung daher meist einheitlicher.

v3.10 `noprotrusion` verhindert das Abschalten des optischen Randausgleichs in den Verzeichnissen. Optischer Randausgleich wird standardmäßig abgeschaltet, wenn das Paket microtype oder ein anderes Paket, das die Anweisung `\microtypesetup` bereitstellt, geladen ist. Wenn also optischer Randausgleich in den Verzeichnissen gewünscht wird, dann muss diese Eigenschaft aktiviert werden. Es ist jedoch zu beachten, dass der optische Randausgleich in Verzeichnissen häufig zu einem falschen Ergebnis führt. Dies ist ein bekanntes Problem des optischen Randausgleichs.

`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 mit dem Besitzer float.

v3.12 `numberline` bedeutet, dass all diejenigen Einträge, die mit Hilfe der Anweisung `\addxcontentsline` oder der Anweisung `\addxcontentslinetoeachtocfile` vorgenommen werden, wobei das optionale Argument für die Nummer fehlt oder leer ist, mit einer leeren `\numberline`-Anweisung versehen werden. Das führt in der Regel dazu, dass diese Einträge nicht linksbündig mit der Nummer, sondern mit dem Text der

v3.20 nummerierten Einträge gleicher Ebene gesetzt werden. Bei Verwendung des Verzeichniseintragsstils `tocline` kann die Eigenschaft weitere Auswirkungen haben. Siehe dazu die Stil-Eigenschaften `breakafternumber` und `entrynumberformat` in Tabelle 13.1 ab Seite 383.

Die KOMA-Script-Klassen setzen diese Eigenschaft bei Verwendung der Option `listof=numberline` für die Dateierweiterungen mit dem Besitzer `float` und bei Verwendung der Option `toc=numberline` für die Dateierweiterung `toc`. Entsprechend wird die Eigenschaft bei Verwendung von Option `listof=nonumberline` oder `toc=nonumberline` wieder zurückgesetzt.

v3.01 `onecolumn` bedeutet, dass für dieses Verzeichnis automatisch der LaTeX-interne Einspaltenmodus mit `\onecolumn` verwendet wird. Das 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 die Eigenschaft per `\AtAddToTocList` (siehe Seite 368) 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 mit dem Besitzer 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 151 zu entnehmen.

**Beispiel:** Wollen Sie, 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 zusätzlich

```
\Ifundefinedorrelax{chapter}{}{%
  \setuptoc{loa}{onecolumn}%
}
```

Die Verwendung von \Ifundefinedorrelax setzt das Paket scrbase voraus (siehe Abschnitt 10.3, Seite 339).

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 bedeuten würden und deshalb in vielen Paketen leider fehlen.

`\Iftocfeature{`*Dateierweiterung*`}{`*Eigenschaft*`}{`*Dann-Teil*`}{`*Sonst-Teil*`}`

v3.28 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. Konfiguration von Verzeichniseinträgen

v3.20 Neben den eigentlichen Verzeichnissen und den zugehörigen Hilfsdateien kann man mit dem Paket tocbasic ab Version 3.20 auch Einfluss auf die Verzeichniseinträge nehmen. Dazu können neue Stile definiert werden. Es stehen aber auch mehrere vordefinierte Stile zur Verfügung. Damit soll tocbasic auch das nie offiziell gewordene KOMA-Script-Paket tocstyle ablösen. Die KOMA-Script-Klassen selbst bauen seit Version 3.20 ebenfalls vollständig auf die von tocbasic bereitgestellten Stile für Verzeichniseinträge.

`tocdepth`

Verzeichniseinträge sind normalerweise hierarchisch geordnet. Dazu wird jeder Eintragsebene ein numerischer Wert zugeordnet. Je höher dieser Wert, desto tiefer in der Hierarchie liegt die Ebene. Bei den Standardklassen hat die Ebene für Teile beispielsweise den Wert -1 und die Ebene für Kapitel den Wert 0. Über den LaTeX-Zähler `tocdepth` wird bestimmt, bis zu welcher Ebene Einträge im Verzeichnis ausgegeben werden.

Bei book ist `tocdepth` beispielsweise mit 2 voreingestellt, es werden also die Einträge der Ebenen `part`, `chapter`, `section` und `subsection` ausgeben. Tiefere Ebenen wie `subsubsection`, deren numerischer Wert 3 ist, werden nicht ausgegeben. Trotzdem sind die Einträge in der Hilfsdatei für das Inhaltsverzeichnis vorhanden.

Die von tocbasic definierten Eintragsstile beachten, abgesehen von `gobble` (siehe \DeclareTOCStyleEntry), ebenfalls `tocdepth`.

`\numberline{`*`Gliederungsnummer`*`}`
`\usetocbasicnumberline[`*`Code`*`]`

v3.20 Zwar definiert bereits der LaTeX-Kern eine Anweisung `\numberline`, diese ist allerdings für die Anforderungen von tocbasic nicht ausreichend. Deshalb definiert tocbasic eigene Anweisungen und setzt `\numberline` bei Bedarf mit Hilfe von `\usetocbasicnumberline` für die einzelnen Verzeichniseinträge entsprechend. Eine Umdefinierung von `\numberline` ist daher bei Verwendung von tocbasic oftmals wirkungslos und führt teilweise auch zu Warnungen.

Man kann auch die Definition von tocbasic generell nutzen, indem man bereits in der Dokumentpräambel `\usetocbasicnumberline` aufruft. Die Anweisung versucht zunächst zu ermitteln, ob in der aktuellen Definition wichtige interne Anweisungen von tocbasic verwendet werden. Ist dies nicht der Fall, wird `\numberline` entsprechend umdefiniert und zusätzlich *`Code`* ausgeführt. Ist kein optionales Arguments angegeben, wird stattdessen via `\PackageInfo` eine Meldung über die erfolgte Umdefinierung ausgegeben. Diese Meldung kann man einfach unterdrücken, indem ein leeres optionales Argument angegeben wird.

Es ist zu beachten, dass `\usetocbasicnumberline` den internen Schalter `\if@tempswa` global verändern kann!

`\DeclareTOCStyleEntry[`*`Optionenliste`*`]{`*`Stil`*`}{`*`Eintragsebene`*`}`
`\DeclareTOCStyleEntries[`*`Optionenliste`*`]{`*`Stil`*`}{`*`Liste von Eintragsebenen`*`}`

v3.20 Über diese Anweisungen werden die Verzeichniseinträge für bestimmte *`Eintragsebenen`* deklariert oder konfiguriert. Dabei ist die *`Eintragsebene`* der Name der jeweiligen Eintragsebene, beispielsweise `section` für einen zur gleichnamigen Gliederungsebene gehörenden Eintrag ins Inhaltsverzeichnis oder `figure` für den Eintrag einer Abbildung ins Abbildungsverzeichnis. Jeder *`Eintragsebene`* wird ein bestimmter *`Stil`* zugeordnet, der zum Zeitpunkt der Deklaration bereits definiert sein muss. Über die *`Optionenliste`* können die verschiedenen, meist vom *`Stil`* abhängenden Eigenschaften des Eintrags festgelegt werden.

Derzeit werden von tocbasic die folgenden Eintragsstile definiert:

`default` ist in der Voreinstellung ein Klon von Stil `dottedtocline`. Klassenautoren, die tocbasic verwenden, wird empfohlen, diesen Stil mit Hilfe von \CloneTOCEntryStyle auf den Standardverzeichniseintragsstil der Klasse zu ändern. Bei den KOMA-Script-Klassen wird `default` beispielsweise zu einem Klon von Stil `tocline`.

`dottedtocline` entspricht dem Stil, der von den Standardklassen book und report für die Inhaltsverzeichniseinträge der Ebenen `section` bis `subparagraph` und bei allen Standardklassen für die Einträge in das Abbildungs- und das Tabellenverzeichnis bekannt ist. Er kennt nur drei Eigenschaften. Die Einträge werden um `indent` von links eingezogen in der aktuellen Schrift ausgegeben. \numberline wird nicht umdefiniert. Die

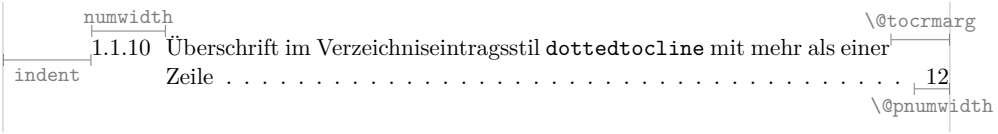



Abbildung 13.1.: Illustration einiger Attribute des Verzeichniseintragsstils `dottedtocline`

Breite der Nummer wird von `numwidth` bestimmt. Bei mehrzeiligen Einträgen wird der Einzug ab der zweiten Zeile um `numwidth` erhöht. Die Seitenzahl wird mit `\normalfont` ausgegeben. Eintragstext und Seitenzahl werden durch eine punktierte Linie verbunden. Abbildung 13.1 illustriert die Eigenschaften des Stils.

`gobble` ist der denkbar einfachste Stil. Einträge in diesem Stil werden unabhängig von allen Einstellungen für `tocdepth` nicht ausgegeben, sondern sozusagen verschluckt. Dennoch verfügt er über die Standardeigenschaft `level`, die jedoch nie ausgewertet wird.

`largetocline` entspricht dem Stil, der von den Standardklassen für die Ebene `part` bekannt ist. Er kennt nur die Eigenschaften *`level`* und *`indent`*. Letzteres ist gleichsam eine Abweichung von den Standardklassen, die selbst keinen Einzug der `part`-Einträge unterstützen.

Vor dem Eintrag wird ein Seitenumbruch erleichtert. Die Einträge werden um `indent` von links eingezogen und mit den Schrifteinstellungen `\large\bfseries` ausgegeben. Sollte `\numberline` verwendet werden, so ist die Nummernbreite fest auf 3 em eingestellt. `\numberline` wird nicht umdefiniert. Die Standardklassen verwenden für *`part`*-Einträge kein `\numberline`. Der Wert hat bei mehrzeiligen Einträgen auch keine Auswirkung auf den Einzug ab der zweiten Zeile.

Abbildung 13.2 illustriert die Eigenschaften des Stils. Dabei wird auch auffällig, dass der Stil einige Ungereimtheiten der Standardklassen übernommen hat, beispielsweise den fehlenden Einzug ab der zweiten Zeile bei mehrzeiligen Einträgen und zwei unterschiedliche Werte für `\@pnumwidth`, die aus einer Abhängigkeit von der Schriftgröße resultieren. Daraus resultiert auch der Umstand, dass im Extremfall der Text der Überschrift der Seitenzahl zu nahe kommen kann. Es ist zu beachten, dass die in der Abbildung gezeigte Breite der Gliederungsnummer nur dann Verwendung findet, wenn auch tatsächlich `\numberline` verwendet wird. Die Standardklassen setzen hingegen einen festen Abstand von 1 em nach der Nummer.

`tocline` ist ein flexibler Stil, der in der Voreinstellung für alle Einträge der KOMA-Script-Klassen verwendet wird. Entsprechend definieren diese Klassen auch die Klone `part`, `chapter` und `section` beziehungsweise `section` und `subsection` mit Hilfe dieses Stils, ändern dann jedoch den *`Initialisierungscode`* der Klone so ab, dass sie unterschiedliche Voreinstellungen besitzen.

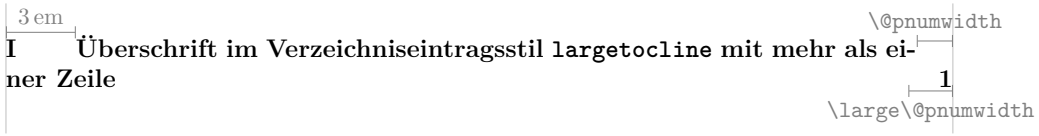



Abbildung 13.2.: Illustration einiger Attribute des Verzeichniseintragsstils `largetocline`

Der Stil kennt neben der Standardeigenschaft `level` noch 20 weitere Eigenschaften. Die Voreinstellungen all dieser Eigenschaften werden abhängig vom Namen der *Eintragsebene* bestimmt und orientieren sich dann an den Ergebnissen der Standardklassen. Es ist daher möglich, nach Laden von tocbasic den Stil der Inhaltsverzeichniseinträge der Standardklassen mit `\DeclareTOCEntryStyle` in `tocline` zu ändern, ohne dass dies unmittelbar zu größeren Veränderungen im Aussehen der Inhaltsverzeichniseinträge führt. So kann man gezielt nur die Eigenschaften ändern, die für erwünschte Änderungen notwendig sind. Dasselbe gilt für Abbildungs- und Tabellenverzeichnis der Standardklassen.

Aufgrund der hohen Flexibilität kann dieser Stil prinzipiell die Stile `dottedtocline`, `undottedtocline` und `largetocline` ersetzen, bedarf dann jedoch teilweise eines höheren Aufwands bei der Konfiguration.

Abbildung 13.3 illustriert einige Längen-Eigenschaften des Stils. Die weiteren sind in Tabelle 13.1, ab Seite 383 erklärt.

v3.27 `toctext` stellt eine Besonderheit dar. Während alle anderen Stile je Eintrag einen Absatz erzeugen, wird hier für alle aufeinanderfolgenden Einträge dieses Stils nur ein einziger Absatz erzeugt. Für die Konfigurierung stehen neben der Standardeigenschaft `level` mit 16 weiteren Eigenschaften fast so viele Möglichkeiten wie bei `tocline` zur Verfügung. Dieser Stil ist aber zwingend darauf angewiesen, dass am Anfang aller anderen Stile

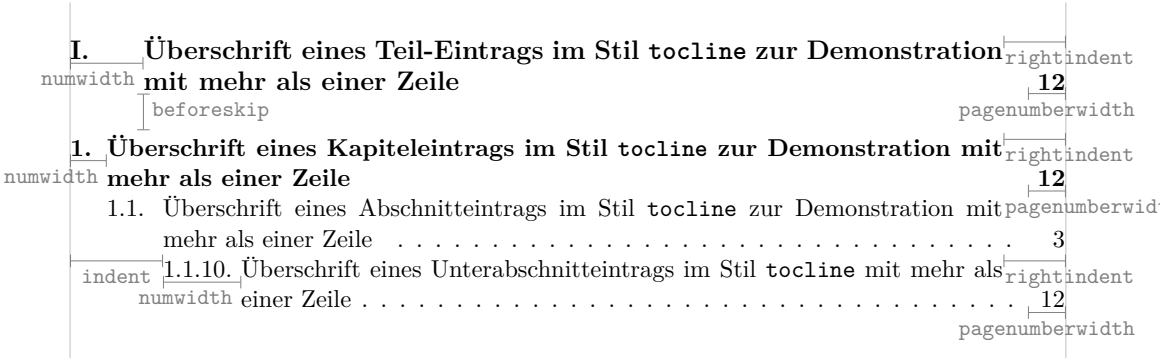



Abbildung 13.3.: Illustration einiger Attribute des Verzeichniseintragsstils `tocline`

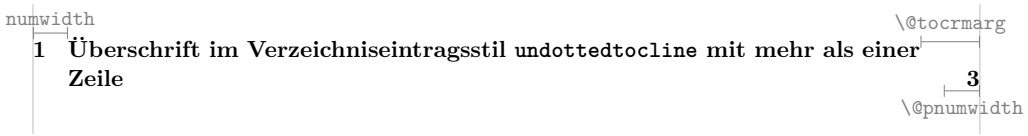



Abbildung 13.4.: Illustration einiger Attribute des Verzeichniseintragsstils `undottedtocline` am Beispiel einer Kapitelüberschrift

und auch am Ende des Verzeichnisses ein noch nicht beendeter Absatz tasächlich beendet wird. Daher sollten Einträge dieses Stils nicht mit Einträgen oder Verzeichnissen kombiniert werden, die an tocbasic vorbei erzeugt werden.

`undottedtocline` entspricht dem Stil, der von den Standardklassen book und report für die Ebene `chapter` und von article für die Ebene `section` bekannt ist. Er kennt nur drei Eigenschaften. Vor dem Eintrag wird ein Seitenumbruch erleichtert und ein vertikaler Abstand eingefügt. Die Einträge werden um `indent` von links eingezogen in `\bfseries` ausgegeben. Dies ist gleichsam eine Abweichung von den Standardklassen, die selbst keinen Einzug für Einträge der genannten Ebenen bieten. `\numberline` wird nicht umdefiniert. Die Breite der Nummer wird von `numwidth` bestimmt. Bei mehrzeiligen Einträgen wird der Einzug ab der zweiten Zeile um `numwidth` erhöht. Abbildung 13.4 illustriert die Eigenschaften des Stils.

Eine Erklärung zu den Eigenschaften der von tocbasic definierten Stile findet sich in Tabelle 13.1. Neben der normalen Zuweisung eines Wertes in der Form *`Schlüssel=Wert`* verstehen beide Befehle für alle Optionen der durch KOMA-Script definierten Stile auch die Zuweisungen in der Form *`Schlüssel:=Eintragsebene`*. In diesem Fall wird die aktuell gültige Einstellung für *`Eintragsebene`* kopiert, soweit diese verfügbar ist. Es kann also beispielsweise mit `indent:=figure` der Einzug für `figure`-Einträge kopiert werden. Für Optionen, die eine Länge oder einen Integer als *`Wert`* erwarten, gibt es außerdem die Möglichkeit mit *`Schlüssel+=Wert`* den *`Wert`* zur aktuell gültigen Einstellung zu addieren. Für eine Substraktion kann ein negativer *`Wert`* verwendet werden. Mit `indent+=1cm` könnte so beispielsweise der Einzug um 1 cm erhöht werden. Für Optionen, die eine Liste als *`Wert`* erwarten, kann mit *`Schlüssel+=Wert`* der neue *`Wert`* an den bereits vorhandenen angehängt werden.

v3.27

v3.31

v3.21 Bei Verwendung der Eigenschaften als Optionen für Anweisung `\DeclareNewTOC` (siehe Seite 400) sind die Namen der Eigenschaften mit dem Präfix `tocentry` zu verwenden, also beispielsweise `tocentrylevel` anstelle von `level`. Die zuvor beschriebene Kopiermöglichkeit mit `:=` ist hierbei ebenfalls verfügbar. Die Addition mit Hilfe von `+=` wird derzeit jedoch nicht unterstützt.

v3.20 Bei Verwendung als Optionen für `\DeclareSectionCommand` (siehe Seite 487) und verwandten Anweisungen sind die Namen der Eigenschaften mit dem Präfix `toc` zu versehen, also beispielsweise `toclevel` anstelle von `level`. Hierbei existiert derzeit weder die Kopiermöglichkeit

mit := noch die Addition mit +=.

Letztlich führt der Aufruf von \DeclareTOCStyleEntry zur Definition der Anweisung \l@*Eintragsebene*.

Während \DeclareTOCStyleEntry nur eine *Eintragsebene* definiert, kann über v3.26 \DeclareTOCStyeEntries auf einen Schlag eine ganze *Liste von Eintragsebenen* definiert werden. Die durch Komma voneinander getrennt angegebenen Eintragsebenen der Liste werden dabei alle mit demselben *Stil* und den über *Optionenliste* angegebenen Einstellungen definiert.

Tabelle 13.1.: Attribute für die vordefinierten Verzeichniseintragsstile von tocbasic

afterpar=*Code*

v3.27 Der angegebene *Code* wird nach dem Ende des Absatzes ausgeführt, in dem ein Eintrag mit dem Stil toctext ausgegeben wird. Verfügen mehrere Einträge über solche Einstellungen, so werden diese in der Reihenfolge der Einträge ausgeführt.

beforeskip=*Länge*

Vertikaler Abstand, der vor einem Eintrag dieser Ebene im Stil tocline eingefügt wird (siehe Abbildung 13.3). Der Abstand wird je nach Ebene mit \vskip oder \addvspace eingefügt, so dass diesbezüglich möglichst Kompatibilität zu den Standardklassen und früheren Versionen von KOMA-Script besteht.

Bei der *Eintragsebene* part wird das Attribut mit 2.25em plus 1pt initialisiert, bei chapter mit 1em plus 1pt. Ist noch keine *Eintragsebene* chapter bekannt, wird stattdessen für section 1em plus 1pt verwendet. Ansonsten wird es für section wie für alle anderen Ebenen mit 0pt plus .2pt initialisiert.

v3.31 Im Stil toctext wird der vertikale Abstand vor dem Absatz eingefügt, wenn es sich um den ersten Eintrag im Absatz handelt. Bei allen weiteren Einträgen des Absatzes wird er ignoriert. Findet die Initialisierung über diesen Stil statt, so wird als Voreinstellung 0pt verwendet.

breakafternumber=*Schalter*

*Schalter* ist einer der Werte für einfache Schalter aus Tabelle 2.5, Seite 41. Ist der Schalter beim Stil tocline aktiviert, so wird nach der mit \numberline gesetzten Nummer eine neue Zeile begonnen, die erneut linksbündig mit der Nummer beginnt. In der Voreinstellung ist die Eigenschaft im Stil tocline nicht gesetzt.

Ist für ein Verzeichnis mit \setuptoc die Eigenschaft numberline gesetzt (siehe Abschnitt 13.2, Seite 375), wie das bei den KOMA-Script-Klassen und Verwendung deren Option toc=numberline der Fall ist, führt dies auch dazu, dass bei nicht nummerierten Einträgen dennoch die Zeile mit der dann leeren Nummer in der Formatierung von entrynumberformat gesetzt wird.

...

Tabelle 13.1.: Attribute der Verzeichniseintragsstile (*Fortsetzung*)

`dynindent=`*`Schalter`*

v3.31
v3.49

*`Schalter`* ist einer der Werte für einfache Schalter aus Tabelle 2.5, Seite 41. Ist der Schalter beim Stil `tocline` oder `toctext` aktiviert, gibt die Eigenschaft `indent` nur noch einen Minimalwert an. Der Maximalwert wird durch die Nummernbreite und den Einzug der via `indentfollows` vorgegebenen Ebenen bestimmt.

`dynnumwidth=`*`Schalter`*

*`Schalter`* ist einer der Werte für einfache Schalter aus Tabelle 2.5, Seite 41. Ist der Schalter beim Stil `tocline` aktiviert, gibt die Eigenschaft `numwidth` nur noch einen Minimalwert an. Übertrifft die beim letzten LaTeX-Lauf ermittelte maximale Breite der Eintragsnummern gleicher Ebene zuzüglich des Wertes von `numsep` diesen Minimalwert, so wird stattdessen der ermittelte Wert verwendet.

`entryformat=`*`Befehl`*

Über diese Eigenschaft kann die Formatierung des gesamten Eintrags verändert werden. Der dabei als Wert angegebene *`Befehl`* hat genau ein Argument zu erwarten. Dieses Argument ist nicht zwingend voll expandierbar. Befehle wie `\MakeUppercase`, die ein voll expandierbares Argument erwarten, dürfen an dieser Stelle also nicht verwendet werden. Font-Änderungen über `entryformat` erfolgen ausgehend von `\normalfont\normalsize`. Es wird darauf hingewiesen, dass die Ausgabe von `linefill` und der Seitenzahl unabhängig von `entryformat` ist. Siehe dazu auch die Eigenschaft `pagenumberformat`.

Initialisiert wird die Eigenschaft für die *`Eintragsebene`* `part` mit der Ausgabe des übergebenen Argument in `\large\bfseries` und für `chapter` in `\bfseries`. Falls bei der Initialisierung von `section` noch keine Ebene `chapter` existiert, wird auch für diese Ebene `\bfseries` verwendet. Bei allen anderen Ebenen wird das Argument unverändert ausgegeben.

`entrynumberformat=`*`Befehl`*

Über diese Eigenschaft kann die Formatierung der mit `\numberline` gesetzten Eintragsnummer verändert werden. Der dabei als Wert angegebene *`Befehl`* hat genau ein Argument zu erwarten. Font-Änderungen erfolgen ausgehend von der Eigenschaft `entryformat`.

Initialisiert wird die Eigenschaft mit der Ausgabe des übergebenen Arguments. Die Eintragsnummer wird also unverändert ausgegeben.

Ist für ein Verzeichnis mit `\setuptoc` die Eigenschaft `numberline` gesetzt (siehe Abschnitt 13.2, Seite 375), wie dies bei den KOMA-Script-Klassen und Verwendung deren Option `toc=numberline` der Fall ist, führt dies auch dazu, dass bei nicht nummerierten Einträgen *`Befehl`* dennoch ausgeführt wird.

...

Tabelle 13.1.: Attribute der Verzeichniseintragsstile (*Fortsetzung*)

`indent=`*`Länge`*

v3.27

Beim Stil `toctext` ist *`Länge`* der horizontale Abstand des Absatzes vom linken Rand. Haben die unterschiedlichen Einträge des Absatzes unterschiedliche Einstellungen, so gewinnt der letzte Eintrag. Bei den übrigen Stilen ist `Länge` entsprechend der horizontale Abstand des Eintrags vom linken Rand (siehe Abbildung 13.1 und Abbildung 13.3).

Bei den Stilen `tocline` und `toctext` wird für alle Eintragsebenen, deren Name mit »`sub`« beginnt, eine Initialisierung mit `indent`+`numwidth` der gleichnamigen Eintragsebene ohne diesen Präfix vorgenommen, falls eine solche Ebene mit entsprechenden Eigenschaften existiert. Bei den Stilen `dottedtocline`, `undottedtocline` und `tocline` findet für die Eintragsebenen `part` bis `subparagraph` sowie `figure`, `table` und `lstlisting` eine Initialisierung mit Werten entsprechend der Standardklassen beziehungsweise des Paket listings statt. Alle anderen Ebenen erhalten keine Initialisierung. Für sie ist eine explizite Angabe daher bei der ersten Verwendung zwingend.

v3.39

Ist für ein Verzeichnis die Eigenschaft `noindent` via `\setuptoc` gesetzt, so ignorieren die Einträge bei allen von KOMA-Script bereitgestellten Stilen diesen Wert und verwenden stattdessen 0 pt. Der Einzug wird also deaktiviert.

`indentfollows=`*`Ebenenliste`*

v3.39

Ist `dynindent` beim Stil `tocline` oder `toctext` gesetzt, so dient die hier angegebene durch Komma separierte Liste an Ebenennamen dazu, den tatsächlichen Einzug zu ermitteln. Dabei findet bei Ebenen, deren Name mit »`sub`« beginnt, eine Initialisierung mit dem Namen ohne diesen Präfix statt. Die KOMA-Script-Klassen setzen außerdem automatisch passende Werte für die Ebenen `section` und `paragraph`.

...

Tabelle 13.1.: Attribute der Verzeichniseintragsstile (*Fortsetzung*)

`level=`*`Integer`*

Numerischer Wert der *Eintragsebene*. Tatsächlich angezeigt werden nur Einträge, deren numerische Ebene nicht größer als Zähler `tocdepth` ist.
Diese Eigenschaft ist für alle Stile zwingend und wird bei der Definition eines Stils automatisch definiert.
Bei den Stilen `tocline` und `toctext` findet für alle Eintragsebenen, deren Name mit »`sub`« beginnt, eine Initialisierung entsprechend dem um eins erhöhten Wert einer gleichnamigen Eintragsebene ohne diesen Präfix statt, falls eine solche Ebene existiert. Bei den Stilen `dottedtocline`, `largetocline`, `tocline`, `toctext` und `undottedtocline` findet für die *Eintragsebene* `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph`, `subparagraph`, `figure`, `table` und `lstlisting` automatisch eine Initialisierung aufgrund des Namens statt. Für andere Ebenen findet eine Initialisierung mit dem Wert der Gliederungsebene statt, falls kompatibel zu den KOMA-Script-Klassen `\`*`Eintragsebene`*`numdepth` definiert ist. Als letzter Ausweg kann auch das interne hyperref-Makro `\toclevel@`*`Eintragsebene`* herangezogen werden.

v3.39

`linefill=`*`Code`*

Beim Stil `tocline` kann zwischen dem Ende des Eintragstextes und der Seitenzahl die Art der Füllung verändert werden. Die Eigenschaft *`linefill`* erhält als Wert direkt den gewünschten *Code*. Für *Eintragsebene* `part` und `chapter` wird die Eigenschaft mit `\hfill` initialisiert. Dadurch rückt die Seitenzahl an den rechten Rand. Ist bisher keine *Eintragsebene* `chapter` definiert, so gilt dies auch für `section`. Alle anderen Ebenen werden mit `\TOCLineLeaderFill` (siehe Seite 394) initialisiert.
Wird *Code* angegeben, der nicht automatisch zu einer Füllung des Abstandes führt, sollte übrigens auch die Eigenschaft `raggedpagenumber` gesetzt werden, damit es nicht zu »`underfull \hbox`«-Meldungen kommt.

`numsep=`*`Länge`*

Der Stil `tocline` versucht sicherzustellen, dass zwischen der Nummer und dem Text eines Eintrags mindestens ein Abstand von *Länge* eingehalten wird. Bei aktiviertem `dynnumwidth` kann die für die Nummer reservierte Breite `numwidth` entsprechend korrigiert werden. Bei nicht aktiviertem `dynnumwidth` wird hingegen lediglich eine Warnung ausgegeben, wenn diese Bedingung nicht eingehalten wird.

v3.27

Der Stil `toctext` fügt dagegen immer einen Abstand dieser *Länge* nach der Nummer des Eintrags ein.
Die Eigenschaft wird mit einem Wert von 0,4 em initialisiert.

...

Tabelle 13.1.: Attribute der Verzeichniseintragsstile (*Fortsetzung*)

`numwidth=`*`Länge`*

Für die Nummer eines Eintrags reservierte Breite (siehe Abbildung 13.1 bis Abbildung 13.4). Dieser Wert wird bei den Stilen `dottedtocline`, `tocline` und `undottedtocline` ab der zweiten Zeile eines Eintrags zum linken Einzug hinzugerechnet.

Beim Stil `tocline` wird für alle Eintragsebenen, deren Name mit »`sub`« beginnt, eine Initialisierung mit dem Wert der gleichnamigen Eintragsebene ohne diesen Präfix zuzüglich 0,9 em vorgenommen, falls eine solche Ebene mit entsprechender Eigenschaft existiert. Bei den Stilen `dottedtocline`, `undottedtocline` und `tocline` findet für die Eintragsebenen `part` bis `subparagraph` sowie `figure`, `table` und `lstlisting` eine Initialisierung mit Werten entsprechend der Standardklassen beziehungsweise des Pakets listings statt. Alle anderen Ebenen erhalten keine Initialisierung. Für sie ist eine explizite Angabe daher bei der ersten Verwendung zwingend.

v3.39

`onendentry=`*`Code`*

v3.27

Führt den angegebenen *`Code`* unmittelbar nach einem Eintrag im Stil `toctext` aus, sofern es nicht der letzte Eintrag im Absatz ist. Der Anwender muss unbedingt sicherstellen, dass *`Code`* auf keinen Fall zum Beenden des Absatzes führt.

Hinweis: Tatsächlich wird der *`Code`* gar nicht am Ende des Eintrags, sondern vor dem nächsten Eintrag im Stil `toctext` ausgeführt.

`onendlastentry=`*`Code`*

v3.27

Führt den angegebenen *`Code`* unmittelbar vor dem Ende des Absatzes mit dem Eintrag im Stil `toctext` aus, sofern es sich um den letzten Eintrag im Absatz handelt. Der Anwender sollte sicherstellen, dass *`Code`* nicht zum Beenden des Absatzes führt.

`onstartentry=`*`Code`*

v3.27

Führt den angegebenen *`Code`* unmittelbar vor dem Eintrag im Stil `toctext` aus, sofern es sich nicht um den ersten Eintrag im Absatz handelt. Der Anwender muss unbedingt sicherstellen, dass *`Code`* auf keinen Fall zum Beenden des Absatzes führt.

`onstartfirstentry=`*`Code`*

v3.27

Führt den angegebenen *`Code`* unmittelbar vor dem Eintrag im Stil `toctext` aus, sofern es sich um den ersten Eintrag im Absatz handelt. Der Anwender muss unbedingt sicherstellen, dass *`Code`* auf keinen Fall zum Beenden des bereits begonnen Absatzes führt.

...

Tabelle 13.1.: Attribute der Verzeichniseintragsstile (*Fortsetzung*)

`onstarthigherlevel=`*`Code`*

Der Stil `tocline` kann zu Beginn eines Eintrags eine Aktion in Abhängigkeit davon ausführen, ob der zuletzt gesetzte Eintrag einen höheren, denselben oder einen niedrigeren *`level`*-Wert hatte. Im Falle, dass der aktuelle Eintrag einen größeren *`level`*-Wert besitzt, in der Hierarchie der Einträge also tiefer steht, wird der über diese Eigenschaft angegebene *`Code`* ausgeführt.
Die Erkennung funktioniert übrigens nur, solange sich `\lastpenalty` seit dem letzten Eintrag nicht geändert hat.
Initialisiert wird die Eigenschaft mit `\LastTOCLevelWasLower` (siehe Seite 393).

`onstartlowerlevel=`*`Code`*

Der Stil `tocline` kann zu Beginn eines Eintrags eine Aktion in Abhängigkeit davon ausführen, ob der zuletzt gesetzte Eintrag einen höheren, denselben oder einen niedrigeren *`level`*-Wert hatte. Im Falle, dass der aktuelle Eintrag einen kleineren *`level`*-Wert besitzt, in der Hierarchie der Einträge also höher steht, wird der über diese Eigenschaft angegebene *`Code`* ausgeführt.
Die Erkennung funktioniert übrigens nur, solange sich `\lastpenalty` seit dem letzten Eintrag nicht geändert hat.
Initialisiert wird die Eigenschaft mit `\LastTOCLevelWasHigher` (siehe Seite 393), was normalerweise dazu führt, dass ein Umbruch vor dem Eintrag begünstigt wird.

`onstartsamelevel=`*`Code`*

Der Stil `tocline` kann zu Beginn eines Eintrags eine Aktion in Abhängigkeit davon ausführen, ob der zuletzt gesetzte Eintrag einen höheren, denselben oder einen niedrigeren *`level`*-Wert hatte. Im Falle, dass der aktuelle Eintrag denselben *`level`*-Wert besitzt, in der Hierarchie der Einträge also gleich gestellt ist, wird der über diese Eigenschaft angegebene *`Code`* ausgeführt.
Die Erkennung funktioniert übrigens nur, solange sich `\lastpenalty` seit dem letzten Eintrag nicht geändert hat.
Initialisiert wird die Eigenschaft mit `\LastTOCLevelWasSame` (siehe Seite 393), was normalerweise dazu führt, dass ein Umbruch vor dem Eintrag begünstigt wird.

`pagenumberbox=`*`Befehl`*

Normalerweise wird die zu einem Eintrag gehörende Seitenzahl rechtsbündig in eine Box der Breite `\@pnumwidth` gesetzt. Beim Stil `tocline` kann der Befehl, der dazu verwendet wird, über diese Eigenschaft konfiguriert werden. Der dabei anzugebende *`Befehl`* hat genau ein Argument zu erwarten.
Initialisiert wird die Eigenschaft mit der bereits erwähnten Box.

...

Tabelle 13.1.: Attribute der Verzeichniseintragsstile (*Fortsetzung*)

`pagenumberformat=`*`Befehl`*

Über diese Eigenschaft kann die Formatierung der Seitenzahl des Eintrags verändert werden. Der dabei als Wert angegebene *`Befehl`* hat genau ein Argument zu erwarten. Font-Änderungen erfolgen ausgehend von `entryformat`, gefolgt von `\normalfont\normalsize`.

Initialisiert wird die Eigenschaft für die *`Eintragsebene`* `part` mit der Ausgabe des übergebenen Arguments in `\large\bfseries`. Für die *`Eintragsebene`* `chapter` wird nur `\bfseries` verwendet. Bei Klassen ohne vordefiniertes `\l@chapter` geschieht dies auch für die *`Eintragsebene`* `section`. Für alle anderen Ebenen erfolgt die Ausgabe in `\normalfont\normalcolor`.

`pagenumberwidth=`*`Länge`*

v3.27

Mit dieser Eigenschaft kann die Breite der Standardbox für die Seitenzahl eines Eintrags im Stil `tocline` von `\@pnumwidth` in die angegebene *`Länge`* geändert werden. Es ist zu beachten, dass bei Änderung des Befehls für die Box über die Eigenschaft `pagenumberbox` die angegebene *`Länge`* nicht mehr automatisch Anwendung findet.

`prepagenumber=`*`Code`*

v3.27

Im Stil `toctext` wird zwischen dem Eintragstext und der Seitenzahl *`Code`* ausgeführt. Dies dient in erster Linie dazu, Abstand oder Trennzeichen zwischen Text und Seitenzahl einzufügen.

Voreingestellt ist mit `\nonbreakspace` ein nicht umbrechbares Leerzeichen.

`raggedentrytext=`*`Schalter`*

v3.21

*`Schalter`* ist einer der Werte für einfache Schalter aus Tabelle 2.5, Seite 41. Ist der Schalter beim Stil `tocline` aktiviert, so wird der Text des Eintrags nicht im Blocksatz, sondern im Flattersatz gesetzt. Dabei werden nur noch Wörter getrennt, die länger als eine Zeile sind.

In der Voreinstellung ist dieser Schalter nicht gesetzt.

...

Tabelle 13.1.: Attribute der Verzeichniseintragsstile (*Fortsetzung*)

`raggedpagenumber=`*`Schalter`*

*`Schalter`* ist einer der Werte für einfache Schalter aus Tabelle 2.5, Seite 41. Ist der Schalter beim Stil `tocline` aktiviert, so wird die Seitenzahl nicht zwingend rechtsbündig gesetzt.
Je nach Wert der Eigenschaft `linefill` kann sich das Setzen dieses Schalters nur im Erscheinen oder Verschwinden einer Warnung oder auch konkret in der Formatierung der Einträge auswirken. Es ist also wichtig, diese beiden Eigenschaften zueinander passend zu setzen.
In der Voreinstellung ist dieser Schalter nicht gesetzt und passt damit zur Initialisierung von `linefill` sowohl mit `\hfill` als auch mit `\TOCLineLeaderFill`.

`raggedright=`*`Schalter`*

v3.27 *`Schalter`* ist einer der Werte für einfache Schalter aus Tabelle 2.5, Seite 41. Ist der Schalter innerhalb eines Absatzes bei irgendeinem Eintrag im Stil `toctext` gesetzt, so wird der komplette Absatz in linksbündigem Flattersatz gesetzt.

`rightindent=`*`Länge`*

v3.27 Mit dieser Eigenschaft kann der rechte Rand für den Text eines Eintrags im Stil `tocline` von `\@tocrmarg` in die angegebene *`Länge`* geändert werden. Beim Stil `toctext` wird entsprechend der rechte Rand für den kompletten Absatz eingestellt.

```
\DeclareTOCEntryStyle{Stil}[Initialisierungscode]{Befehlscode}
\DefineTOCEntryOption{Option}[Säumniswert]{Code}
\DefineTOCEntryBooleanOption{Option}[Säumniswert]{Präfix}{Postfix}{Erklärung}
\DefineTOCEntryCommandOption{Option}[Säumniswert]{Präfix}{Postfix}{Erklärung}
\DefineTOCEntryIfOption{Option}[Säumniswert]{Präfix}{Postfix}{Erklärung}
\DefineTOCEntryLengthOption{Option}[Säumniswert]{Präfix}{Postfix}{Erklärung}
\DefineTOCEntryNumberOption{Option}[Säumniswert]{Präfix}{Postfix}{Erklärung}
```

v3.20 `\DeclareTOCEntryStyle` ist eine der komplexesten Anweisungen in KOMA-Script. Sie richtet sich daher ausdrücklich an LaTeX-Entwickler und nicht an LaTeX-Anwender. Mit ihrer Hilfe ist es möglich, einen neuen *`Stil`* für Verzeichniseinträge zu definieren. Üblicherweise werden Verzeichniseinträge mit `\addcontentsline` oder bei Verwendung von tocbasic vorzugsweise mit `\addxcontentsline` (siehe Abschnitt 13.1, Seite 371) erzeugt. Dabei schreibt LaTeX eine zugehörige Anweisung `\contentsline` in die jeweilige Hilfsdatei. Beim Einlesen dieser Hilfsdatei führt LaTeX dann für jedes `\contentsline` eine Anweisung `\l@`*`Eintragsebene`* aus.

Wird später einer *`Eintragsebene`* über `\DeclareTOCStyleEntry` ein *`Stil`* zugewiesen, so wird zunächst *`Initialisierungscode`* ausgeführt, falls angegeben, und dann *`Befehlscode`* für die Definition von `\l@`*`Eintragsebene`* verwendet. *`Befehlscode`* ist also letztlich der Co-

de, der bei \l@*Eintragsebene* ausgeführt wird. Dabei ist #1 der Name der Eintragsebene, während ##1 und ##2 Platzhalter für die beiden Argumente von \l@*Eintragsebene* sind.

Der *Initialisierungscode* dient einerseits dazu, die Einstellungen eines Stils zu initialisieren. Entwickler sollten darauf achten, dass wirklich alle Einstellungen hier bereits einen Wert erhalten. Nur dann funktioniert \DeclareTOCStyleEntry auch ohne Angabe einer *Optionenliste* fehlerfrei. Darüber hinaus hat der *Initialisierungscode* auch alle Optionen, die der jeweilige Stil versteht, zu definieren. Zwingend vordefiniert wird lediglich level. Der eingestellte Wert für level kann in *Befehlscode* mit \@nameuse{#1tocdepth} abgefragt werden, um ihn beispielsweise mit dem Wert des Zählers tocdepth zu vergleichen.

Zur Definition neuer Optionen für die Eigenschaften einer Eintragsebene existieren nur innerhalb von *Initialisierungscode* die Anweisungen \DefineTOCEntryBooleanOption, \DefineTOCEntryCommandOption, \DefineTOCEntryIfOption, \DefineTOCEntryLengthOption und \DefineTOCEntryNumberOption. Diese Anweisungen definieren jeweils eine *Option*, die bei ihrem Aufruf eine Anweisung *Präfix*Eintragsebene*Postfix* mit dem übergebenen Wert oder bei Fehlen einer Wertzuweisung mit dem *Säumniswert* definieren. Eine Besonderheit stellt \DefineTOCEntryIfOption dar. Diese definiert *Präfix*Eintragsebene*Postfix* immer als Anweisung mit zwei Argumenten. Ist der an die Option übergebene Wert einer der Aktivierungswerte aus Tabelle 2.5, Seite 41, so expandiert die Anweisung zum ersten Argument. Ist der an die Option übergebene Wert hingegen ein Deaktivierungswert, so expandiert die Anweisung zum zweiten Argument.

v3.27 Neben den normalen Optionen der Form *Schlüssel*=*Wert* werden von allen fünf \DefineTOCEntry...Option-Anweisungen automatisch Optionen der Form *Schlüssel*:=*Eintragsebene* definiert. Diese dienen dazu, den Wert einer anderen *Eintragsebene* zu kopieren, sofern der Wert in einem Makro mit gleichem *Präfix* und *Postfix* gespeichert ist. Bei den von tocbasic vordefinierten Stilen ist das für gleichnamige Optionen über Stilgrenzen hinweg der Fall.

Vergleichbar dazu werden von \DefineTOCEntryLengthOption und \DefineTOCEntryNumberOption jeweils zusätzliche Optionen der Form *Schlüssel*+=*Wert* mitdefiniert, die dazu dienen, zu dem in *Präfix*Eintragsebene*Postfix* bereits gespeicherten Wert den neuen *Wert* zu addieren.

Die *Erklärung* sollte ein möglichst kurzer Text sein, der den Sinn der Option mit wenigen Schlagworten beschreibt. Er wird von tocbasic bei Fehlermeldungen, Warnungen und Informationen auf dem Terminal und in der log-Datei ausgegeben.

**Beispiel:** Der einfachste Stil von tocbasic, gobble, wurde mit

```
\DeclareTOCEntryStyle{gobble}{}%
```

definiert. Würde man nun mit

```
\DeclareTOCStyleEntry[level=1]{gobble}{dummy}
```

eine Eintragsebene dummy in diesem Stil definieren, so würde das

```
\def\dummytocdepth{1}
\def\l@dummy#1#2{}
```

entsprechen.

Innerhalb von Stil `tocline` wird beispielsweise

```
\DefineTOCEntryCommandOption{linefill}%
  [\TOCLineLeaderFill]{scr@tso@}{@linefill}%
  {filling between text and page number}%
```

verwendet, um Option `linefill` zu definieren. Durch die Angabe von `\TOCLineLeaderFill` als *Säumniswert* würde ein Aufruf wie

```
\DeclareTOCStyleEntry[linefill]{tocline}{part}
```

unter anderem die Definition

```
\def\scr@tso@part@linefill{\TOCLineLeaderFill}
```

vornehmen.

Wer sich selbst einen Stil definieren möchte, dem sei empfohlen, zunächst die Definition des Stils `dottedtocline` zu studieren. Nachdem dessen Definition verstanden wurde, gibt dann die deutlich komplexere Definition von Stil `tocline` viele Hinweise darauf, wie die Anweisungen sinnvoll zu verwenden sind.

In vielen Fällen wird es jedoch auch ausreichen, einen der vorhandenen Stile mit `\CloneTOCEntryStyle` zu klonen und dann dessen Initialisierungscode mit Anweisung `\TOCEntryStyleInitCode` oder `\TOCEntryStyleStartInitCode` abzuändern.

`\DefineTOCEntryOption` dient eher der Definition der übrigen Anweisungen und sollte in der Regel nicht direkt verwendet werden. Normalerweise besteht dafür auch keine Notwendigkeit. Sie sei hier nur der Vollständigkeit halber erwähnt.

`\CloneTOCEntryStyle{`*`Stil`*`}{`*`neuer Stil`*`}`

v3.20

Mit dieser Anweisung kann ein existierender *`Stil`* geklont werden. Dabei wird ein *`neuer Stil`* mit denselben Eigenschaften und Voreinstellungen wie der existierende *`Stil`* deklariert. Das Paket selbst verwendet `\CloneTOCEntryStyle`, um den Stil `default` als Klon von `dottedtocline` zu deklarieren. Die KOMA-Script-Klassen verwenden die Anweisung um die Stile `part`, `section` und `chapter` oder `subsection` als Klon von `tocline` zu deklarieren und dann mit `\TOCEntryStyleInitCode` und `\TOCEntryStyleStartInitCode` abzuändern. Der Stil `default` wird von scrbook und scrreprt neu als Klon von `section` und von scrartcl als Klon von `subsection` deklariert.

\TOCEntryStyleInitCode{*Stil*}{*Initialisierungscode*}
\TOCEntryStyleStartInitCode{*Stil*}{*Initialisierungscode*}

v3.20 Jeder Verzeichniseintragsstil verfügt über einen Initialisierungscode. Dieser wird immer dann aufgerufen, wenn einer Verzeichnisebene der entsprechende *Stil* mit \DeclareTOCEntryStyle zugewiesen wird. Dieser *Initialisierungscode* sollte keine globalen Seiteneffekte aufweisen, da er auch für lokale Initialisierungen innerhalb anderer Anweisungen wie \DeclareNewTOC verwendet wird. Der *Initialisierungscode* dient einerseits dazu, Eigenschaften für den jeweiligen *Stil* zu definieren. Er setzt aber auch die Standardeinstellungen für diese Eigenschaften.

Mit Hilfe der Anweisungen \TOCEntryStyleStartInitCode und \TOCEntryStyleInitCode kann der für einen *Stil* bereits definierte Initialisierungscode um weiteren *Initialisierungscode* erweitert werden. Dabei fügt \TOCEntryStyleStartInitCode den neuen *Intialisierungscode* vorn an, während \TOCEntryStyleInitCode den *Initialisierungscode* hinten an den vorhandenen Code anfügt. Dies wird beispielsweise von den KOMA-Script-Klassen verwendet, um für den von tocline geklonten Stil part Füllung, Schrift und vertikalen Abstand passend zu initialisieren.

**Beispiel:** Die Klassen scrbook und scrreprt verwenden

```
\CloneTOCEntryStyle{tocline}{section}
\TOCEntryStyleStartInitCode{section}{%
  \expandafter\providecommand%
  \csname scr@tso@#1@linefill\endcsname
  {\TOCLineLeaderFill\relax}%
}
```

um den Stil section als abgewandelten Klon von tocline zu definieren.

\LastTOCLevelWasHigher
\LastTOCLevelWasSame
\LastTOCLevelWasLower

v3.20 Bei Einträgen im Stil tocline wird am Anfang abhängig vom Wert von \lastpenalty eine dieser drei Anweisungen ausgeführt. Dabei fügen \LastTOCLevelWasHigher und \LastTOCLevelWasSame im vertikalen Modus \addpenalty{\@lowpenalty} ein und ermöglichen so einen Umbruch vor Einträgen gleicher oder übergeordneter Ebene. \LastTOCLevelWasLower ist hingegen bisher leer definiert, so dass zwischen einem Eintrag und seinem ersten Untereintrag normalerweise ein Umbruch untersagt ist.

Anwender sollten diese Anweisungen nicht umdefinieren. Stattdessen können und sollten Änderungen bei Zuweisung des Stils an eine Eintragsebene gezielt über die Eigenschaften onstartlowerlevel, onstartsamelevel und onstarthigherlevel vorgenommen werden.

`\TOCLineLeaderFill[`*`Füllzeichen`*`]`

v3.20

Die Anweisung ist dazu gedacht, als Wert für Eigenschaft `linefill` des Verzeichniseintragsstils *`tocline`* verwendet zu werden. Sie erzeugt dann eine Verbindung zwischen dem Ende des Textes eines Eintrags und der zugehörigen Seitenzahl. Das *`Füllzeichen`*, das dazu in regelmäßigem Abstand wiederholt wird, kann als optionales Argument angegeben werden. Voreinstellung ist ein Punkt.

Wie der Name schon vermuten lässt, werden die Füllzeichen mit Hilfe von `\leaders` gesetzt. Als Abstand wird wie bei der LaTeX-Kern-Anweisung `\@dottedtocline` vor und nach dem Füllzeichen `\mkern\@dotsep mu` verwendet.

## 13.4. 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 nicht etwa umdefinieren!

`\tocbasic@extend@babel{`*`Dateierweiterung`*`}`

Das Paket babel (siehe [BB13]) 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 375) kann das unterdrückt werden. Für die Dateinamenerweiterungen `toc`, `lof` und `lot` unterdrückt tocbasic dies bereits selbst, da babel sie von sich aus vornimmt.

`\tocbasic@starttoc{`*`Dateierweiterung`*`}`

Diese Anweisung ist der eigentliche Ersatz der Anweisung `\@starttoc` aus dem LaTeX-Kern. Es ist die Anweisung, die sich hinter `\listoftoc*` (siehe Abschnitt 13.2, Seite 372) 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 kann. Die Erklärungen der Haken finden Sie im Anschluss.

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 Laden des Pakets scrwfile [Koh23c] oder die Verwendung von LuaLaTeX 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` definierten Anweisungen ausgeführt. Es ist erlaubt, diesen Haken mit Hilfe von `\g@addto@macro` zu erweitern.

KOMA-Script nutzt diese Haken, um 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 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, sollten 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 sie. Anderenfalls definiert \tocbasic@listhead diese vor der Verwendung.

\tocbasic@addxcontentsline{*Dateierweiterung*}{*Ebene*}{*Gliederungsnummer*}{*Eintrag*}
\nonumberline

v3.12

Anweisung \tocbasic@addxcontentsline nimmt einen *Eintrag* der angegebenen *Ebene* in das über die *Dateierweiterung* spezifizierte Verzeichnis vor. Ob der Eintrag nummeriert wird oder nicht, hängt davon ab, ob das Argument *Gliederungsnummer* leer ist oder nicht. Im Falle eines leeren Argument wird dem *Eintrag* ein \nonumberline ohne Argument vorangestellt. Anderenfalls wird wie gewohnt \numberline mit *Gliederungsnummer* als Argument verwendet.

Die Anweisung \nonumberline wird innerhalb \listoftoc (siehe Abschnitt 13.2, Seite 372) entsprechend der Eigenschaft numberline (siehe Abschnitt 13.2, Seite 375) umdefiniert. Dadurch wirkt sich das Setzen oder Löschen dieser Eigenschaft bereits beim nächsten LaTeX-Lauf aus.

\tocbasic@DependOnPenaltyAndTOCLevel{*Eintragsebene*}
\tocbasic@SetPenaltyByTOCLevel{*Eintragsebene*}

v3.20

Der Verzeichniseintragsstil tocline (siehe Abschnitt 13.3) setzt am Ende jedes Eintrags über \tocbasic@SetPenaltyByTOCLevel \penalty so, dass nach einem Eintrag kein Seitenumbruch erfolgen darf. Der genaue Wert wird dabei abhängig von der *Eintragsebene* gewählt.

Über \tocbasic@DependOnPenaltyAndTOCLevel wird am Anfang eines Eintrags, abhängig von \lastpenalty und *Eintragsebene*, die über die Eigenschaft onstartlowerlevel im internen Makro \scr@tso@*Eintragsebene*@LastTOCLevelWasHigher, über die Eigenschaft onstartsamelevel im zugehörigen, internen Makro \scr@tso@*Eintragsebene*@LastTOCLevelWasSame oder über die Eigenschaft onstarthigherlevel im zugehörigen, internen Makro \scr@tso@*Eintragsebene*@LastTOCLevelWasLower gespeicherte Aktion ausgeführt. In der Voreinstellung erlauben die ersten beiden einen Umbruch, wenn sie im vertikalen Modus ausgeführt werden.

Entwicklern, die eigene Stile kompatibel mit `tocline` erstellen wollen, sei empfohlen, dieses Verhalten zu kopieren. Zu diesem Zweck dürfen sie auf diese eigentlich internen Makros zurückgreifen.

## 13.5. 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 namens `remarkbox` bereitstellt:

```
\newenvironment{remarkbox}%
  {\@float{remarkbox}}%
  {\end@float}
```

Jede Gleitumgebung hat eine Standardplatzierung. Diese setzt sich aus einer oder mehreren der bekannten Platzierungsoptionen `b`, `h`, `p` und `t` 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 numerischen Gleittyp zwischen 1 und 31. 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. Abbildungen dürfen also Tabellen überholen und umgekehrt.

```
\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 beziehen sich automatisch alle Optionen, die von den KOMA-Script-Klassen für Verzeichnisse von Gleitumgebungen angeboten werden, auch auf das neue Verzeichnis.

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 alle weiteren Sprachen, die man unterstützen will. Siehe dazu Abschnitt 10.4, ab Seite 343.

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 stattdessen auch die Einstellungen selbst kopieren können:

```
\DeclareTOCStyleEntry[level:=figure,%
                      indent:=figure,%
                      numwidth:=figure]%
                     {tocline}{remarkbox}
```

Selbstverständlich wären auch explizite Festlegungen möglich:

```
\DeclareTOCStyleEntry[level=1,%
                      indent=1.5em,%
                      numwidth=2.3em]%
                     {tocline}{remarkbox}
```

Außerdem wollen Sie, dass sich Kapiteleinträge 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 schon. 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.

## 13.6. 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 Abschnitt 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, ab Seite 407) können dann verwendet werden.

*Dateierweiterung* definiert dabei die Dateiendung der Hilfsdatei, die das Verzeichnis repräsentiert, wie dies in Abschnitt 13.1, ab Seite 366 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, Seite 31) bekannt ist. Diese Optionen können jedoch *nicht* mit `\KOMAoptions` gesetzt werden! Eine Übersicht über die möglichen Optionen bietet Tabelle 13.2.

v3.20 Wird Option `tocentrystyle` nicht gesetzt, so wird bei Bedarf der Stil `default` verwendet. Näheres zu diesem Stil ist Abschnitt 13.3 zu entnehmen. Soll kein Befehl für Verzeichniseinträge definiert werden, so kann ein leeres Argument, also wahlweise `tocentrystyle=` oder `tocentrystyle={}` verwendet werden.

v3.21 Abhängig vom Stil der Verzeichniseinträge können auch alle für diesen Stil gültigen Eigenschaften gesetzt werden, indem die entsprechenden in Tabelle 13.1 ab Seite 383 aufgeführten Namen, mit dem Präfix `tocentry` versehen, in der *Optionenliste* angegeben werden. Nachträgliche Änderungen am Stil sind mit `\DeclareTOCStyleEntry` jederzeit möglich. Siehe dazu Abschnitt 13.3, Seite 379.

v3.06 Tabelle 13.2.: Optionen für die Anweisung `\DeclareNewTOC`

v3.09 `atbegin=`*Code*
: Falls eine neue Gleitumgebung oder nicht gleitende Umgebung definiert wird, so wird *Code* jeweils am Anfang dieser Umgebung ausgeführt.

v3.09 `atend=`*Code*
: Falls eine neue Gleitumgebung oder nicht gleitende Umgebung definiert wird, so wird *Code* jeweils am Ende dieser Umgebung ausgeführt.

`category=`*Kategorie*
v3.27 : Diese Option kann als Synonym für `owner=`*Besitzer* verwendet werden.

...

Tabelle 13.2.: Optionen für die Anweisung \DeclareNewTOC (*Fortsetzung*)

`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* abhängig gemacht werden. Eine Einstellung `counterwithin=chapter` wird bei Klassen mit `\chapter` jedoch nur im Hauptteil (siehe `\frontmatter`, `\mainmatter` und `\backmatter` in Abschnitt 3.15, Seite 99) und nur dann beachtet, wenn der Zähler `chapter` bei der Ausgabe größer als Null ist. Bei Klassen ohne `\chapter` gilt dies entsprechend für die Einstellung `counterwithin=section` und Zähler `section`.

v3.35

`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 geändert werden kann. Mit dieser Option wird das *Gleitverhalten* für die optional erstellbare Gleitumgebung (siehe Option `float`) festgelegt. Die Syntax und Semantik sind 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 numerischen 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`.

...

Tabelle 13.2.: Optionen für die Anweisung \DeclareNewTOC (*Fortsetzung*)

`hang=`*Einzug*

v3.20 Diese Option gilt seit KOMA-Script 3.20 als überholt. Die Breite der Nummer des Verzeichniseintrags ist nun stattdessen als Eigenschaft in Abhängigkeit des Verzeichniseintragsstils von Option `tocentrystyle` anzugeben. Bei den Stilen von KOMA-Script wäre das beispielsweise die Eigenschaft `numwidth` und damit Option `tocentrynumwidth`. Besitzt ein Stil diese Eigenschaft, so wird sie von `\DeclareNewTOC` mit 1,5 em voreingestellt. Diese Voreinstellung kann durch explizite Angabe von `tocentrynumwidth=`*Wert* leicht mit einem anderen *Wert* überschrieben werden. Für Abbildungen verwenden die KOMA-Script-Klassen beispielsweise den *Wert* `2.3em`.

`indent=`*Einzug*

v3.20 Diese Option gilt seit KOMA-Script 3.20 als überholt. Der Einzug des Verzeichniseintrags von links ist nun stattdessen als Eigenschaft in Abhängigkeit des Verzeichniseintragsstils von Option `tocentrystyle` anzugeben. Bei den Stilen von KOMA-Script wäre das beispielsweise die Eigenschaft `indent` und damit Option `tocentryindent`. Besitzt ein Stil diese Eigenschaft, so wird sie von `\DeclareNewTOC` mit 1 em voreingestellt. Diese Voreinstellung kann durch explizite Angabe von `tocentryindent=`*Wert* leicht mit einem anderen *Wert* überschrieben werden. Für Abbildungen verwenden die KOMA-Script-Klassen beispielsweise den *Wert* `1.5em`.

`level=`*Gliederungsebene*

v3.20 Diese Option gilt seit KOMA-Script 3.20 als überholt. Der numerische Wert der Ebene des Verzeichniseintrags ist nun stattdessen als Eigenschaft in Abhängigkeit des Verzeichniseintragsstils von Option `tocentrystyle` anzugeben. Nichtsdestotrotz haben alle Stile die Eigenschaft `level` und damit Option `tocentrylevel`. Die Eigenschaft wird von `\DeclareNewTOC` mit 1 voreingestellt. Diese Voreinstellung kann durch explizite Angabe von `tocentrylevel=`*Wert* leicht mit einem anderen *Wert* überschrieben werden.

`listname=`*Verzeichnistitel*

Jedes Verzeichnis hat eine Überschrift, die durch diese Option bestimmt werden kann. Ist die Option nicht angegeben, so wird 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.

...

Tabelle 13.2.: Optionen für die Anweisung \DeclareNewTOC (*Fortsetzung*)

`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 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 Umgebung, in der sie verwendet wird, durchbricht.

`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 als *`Besitzer`* die Kategorie »`float`« verwendet, die auch von den KOMA-Script-Klassen für das Abbildungs- und das Tabellenverzeichnis verwendet wird.

`setup=`*`Liste von Eigenschaften`*

v3.25 Die *`Liste von Eigenschaften`* wird via `\setuptoc` gesetzt. Es wird darauf hingewiesen, dass für die Angabe mehrerer durch Komma getrennter Eigenschaften die *`Liste von Eigenschaften`* in geschweifte Klammern gesetzt werden muss.

`tocentrystyle=`*`Eintragsstil`*

v3.20 *`Eintragsstil`* gibt den Stil an, den Einträge in das entsprechende Verzeichnis haben sollen. Der Name der Eintragsebene wird dabei über Option `type` bestimmt. Zusätzlich zu den Optionen dieser Tabelle können auch alle Eigenschaften des Stils als Optionen angegeben werden, indem die Namen der Eigenschaften mit dem Präfix `tocentry` ergänzt werden. So kann der numerische Wert der Ebene beispielsweise als `tocentrylevel` angegeben werden. Näheres zu den Stilen ist Abschnitt 13.3 ab Seite 378 zu entnehmen.

...

Tabelle 13.2.: Optionen für die Anweisung \DeclareNewTOC (*Fortsetzung*)

`tocentry`*`Stiloption`*`=`*`Wert`*

v3.20 Weitere Optionen in Abhängigkeit vom via `tocentrystyle` gewählten *`Eintragsstil`*. Siehe dazu Abschnitt 13.3 ab Seite 378. Für die von tocbasic vordefinierten Verzeichniseintragsstile finden sich die als *`Stiloption`* verwendbaren Attribute in Tabelle 13.1, ab Seite 383.

`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`* der Wert von `type` mit angehängtem »s« verwendet.

`unset=`*`Liste von Eigenschaften`*

v3.25 Die *`Liste von Eigenschaften`* wird via \unsettoc aufgehoben. Es wird darauf hingewiesen, dass für die Angabe mehrerer durch Komma getrennter Eigenschaften die *`Liste von Eigenschaften`* in geschweifte Klammern gesetzt werden muss.

**Beispiel:** Das Beispiel aus Abschnitt 13.5 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` und einige interne Anweisunen für die Dateiendung `lor` definiert. Soll der Gleitumgebungstyp dem Paket überlassen werden, so kann Option `floattype` entfallen. Wird zusätzlich die Option `nonfloat`

Tabelle 13.3.: Gegenüberstellung von Beispielumgebung `remarkbox` und Umgebung `figure`

| Umgebung `remarkbox` | Umgebung `figure` | Optionen von `\DeclareNewTOC` | Kurzbeschreibung |
|---|---|---|---|
| `remarkbox` | `figure` | `type`, `float` | Gleitumgebung des jeweiligen Typs. |
| `remarkbox*` | `figure*` | `type`, `float` | spaltenübergreifende Gleitumgebung des jeweiligen Typs |
| `remarkbox` | `figure` | `type`, `float` | Zähler, der von `\caption` verwendet wird |
| `\theremarkbox` | `\thefigure` | `type`, `float` | Anweisung zur Ausgabe des jeweiligen Zählers |
| `\remarkboxformat` | `\figureformat` | `type`, `float` | Anweisung zur Formatierung des jeweiligen Zählers in der Ausgabe von `\caption` |
| `\remarkboxname` | `\figurename` | `type`, `float`, `name` | Name, der im Label von `\caption` verwendet wird |
| `\listofremarkboxes` | `\listoffigures` | `types`, `float` | Anweisung zur Ausgabe des jeweiligen Verzeichnisses |
| `\listremarkboxname` | `\listfigurename` | `type`, `float`, `listname` | Überschrift des jeweiligen Verzeichnisses |
| `\fps@remarkbox` | `\fps@figure` | `type`, `float`, `floattype` | numerischer Gleitumgebungstyp zwecks Reihenfolgenerhalts |
| `lor` | `lof` | | Dateiendung der Hilfsdatei für das jeweilige Verzeichnis |

angegeben, wird außerdem eine nicht gleitende Umgebung `remarkbox-` definiert, in der ebenfalls `\caption` verwendet werden kann.

Zum besseren Verständnis zeigt Tabelle 13.3 eine Gegenüberstellung der Anweisungen und Umgebungen für die neu erstellte Beispielumgebung `remarkbox` mit den entsprechenden Befehlen und Umgebungen für Abbildungen.

Und hier nun eine mögliche Verwendung der Umgebung:

```
\begin{remarkbox}
  \centering
  Gleiches sollte immer auf gleiche Weise und
  mit gleichem Aussehen gesetzt werden.
  \caption{Erster Hauptsatz der Typografie}
  \label{rem:typo1}
\end{remarkbox}
```

Ein Ausschnitt aus einer Beispielseite mit dieser Umgebung könnte dann so aussehen:

> Gleiches sollte immer auf gleiche Weise und mit gleichem Aussehen gesetzt werden.
>
> Merksatz 1: Erster Hauptsatz der Typografie

Benutzer von älteren Versionen von Paket hyperref sollten Option `listname` übrigens immer angeben. Anderenfalls kommt es in der Regel zu einer Fehlermeldung, weil hyperref nicht mit dem `\MakeUppercase` im Namen des Verzeichnisses zurecht kommt, das benötigt wird, um den ersten Buchstaben des Wertes von `types` in Großbuchstaben zu wandeln. Besser ist natürlich die Verwendung eines aktuellen hyperref mit einem aktuellen LaTeX.

## 13.7. Titel von (Gleit-)Umgebungen

v3.46

Wie im vorherigen Abschnitt erklärt wurde, können mit der Anweisung `\DeclareNewTOC` auch neue gleitende und nicht gleitende Umgebungen definiert werden, deren mit `\caption` gesetzte Titel automatisch in das jeweils zugehörige Verzeichnisse eingetragen werden. Auch dafür bietet tocbasic einige Erweiterungen.

Es wird darauf hingewiesen, dass alle in diesem Abschnitt beschriebenen Möglichkeiten zwingen eine LaTeX-Version ab 2022/06/01 benötigen. Mit älteren LaTeX-Versionen sind diese Möglichkeiten daher *nicht* verfügbar!

An dieser Stelle sei darauf hingewiesen, dass die in diesem Abschnitt dokumentierten Möglichkeiten derzeit auch bei Verwendung des caption-Pakets funktionieren. Getestet wurde dies mit Version 2023/08/05 v3.6o des caption-Pakets. Generell wird jedoch nicht empfohlen, sich bei Verwendung irgend eines Pakets, das `\caption` oder `\captionof` bereitstellt oder verändert, oder bei Verwendung von Klassen, die `\caption` in zu den Standardklassen oder den KOMA-Script-Klassen inkompatibler Weise bereitstellen, auf irgendeine der in diesem Abschnitt beschriebenen Möglichkeiten zu verlassen. Bei Verwendung des vorgenannten caption-Pakets sollten beispielsweise stattdessen die in dessen Anleitung dokumentierten Möglichkeiten genutzt werden. Siehe dazu [Som13].

```
captionposition=top
captionposition=bottom
captionabove=Ein-Aus-Wert
captionbelow=Ein-Aus-Wert
```

In der Voreinstellung werden die Abstände über und unter den Titeln so gewählt, dass sie für Titel unter dem jeweiligen Objekt, beispielsweise unter eine Abbildung oder einer Tabelle passen. Dies entspricht der Einstellung `captionposition=bottom` oder den synonymen Einstellungen `captionabove=false` oder `captionbelow=true`. Mit den Einstellungen

captionposition=top, captionabove=true oder captionbelow=false kann man die Abstände aber auch automatisch vertauschen, so dass sie für Titel über dem Objekt passen. Die Wahl erfolgt innerhalb des Aufrufs von \caption beziehungsweise \captionof unter der Voraussetzung, dass der \caption-Patch aktiviert wurde. Siehe dazu Option enablepatch in Abschnitt 13.8 ab Seite 410.

Es ist zu beachten, dass es sich bei diesen Optionen nicht um KOMA-Script-Optionen handelt. Sie können also *nicht* mit \KOMAoptions oder \KOMAoption gesetzt werden. Stattdessen sind sie beim Laden von tocbasic als optionales Argument von \usepackage – bzw. \RequirePackage für Klassen- und Paketautoren – anzugeben. Zum Ausgleich ist es möglich, das Paket mehrfach mit unterschiedlichen Optionen zu laden. Darüber hinaus kann man sie innerhalb des Dokuments jederzeit per \SetKeys[tocbasic]{*Optionen*} setzen.

Als *Ein-Aus-Wert* sind nur true oder false erlaubt. Die Verwendung von captionabove oder captionbelow ohne Wert entspricht captionabove=true beziehungsweise captionbelow=true.

Als Besonderheit definiert tocbasic darüber hinaus für alle mit Hilfe von \DeclareNewTOC definierten Umgebungen zusätzlich entsprechende Optionen mit dem Präfix des Namens der Umgebung, also beispielsweise für die bei den KOMA-Script-Klassen definierten Umgebungen figure und table die Optionen figurecaptionposition=Wert, figurecaptionabove=*Ein-Aus-Wert*, figurecaptionbelow=*Ein-Aus-Wert*, tablecaptionposition=Wert, tablecaptionabove=*Wert* und tablecaptionbelow=*Wert*. Dabei ist neben den oben für captionposition, captionabove und captionbelow angegebenen Werten zusätzlich jeweils der *Wert* default erlaubt. Dieser deaktiviert die expliziten Einstellungen für die jeweilige Umgebung, so dass wieder die Einstellungen der allgemeinen Option captionposition, captionabove beziehungsweise captionbelow verwendet werden.

```
\caption[Verzeichniseintrag]{Titel}
\captionbelow[Verzeichniseintrag]{Titel}
\captionabove[Verzeichniseintrag]{Titel}
```

Wie bei den Optionen captionposition erklärt, werden Gleitumgebungen und auch die nicht gleitenden Umgebungen, die mit \DeclareNewTOC definiert werden können, normalerweise per \caption mit einem Titel versehen. Dabei kann mit Hilfe von tocbasic und der genannten Option eingestellt werde, ob die Abstände für Titel über oder unter dem Objekt gewählt werden sollen. Da dies aber nur funktioniert, wenn \caption von tocbasic passend umdefiniert wird, gibt es zusätzlich die Anweisungen \captionbelow und \captionabove. Diese funktionieren in der Regel auch dann, wenn der entsprechende Patch nicht aktiviert wurde, also \caption nicht auf die Optionen captionposition, captionabove und captionbelow reagiert.

**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

```
\usepackage[enablepatch=caption,tablecaptionabove]{tocbasic}
```

verwenden. Dann genügt es, wenn Sie innerhalb der `table`-Umgebung `\caption` verwenden. Sie erhalten trotzdem das Ergebnis von `\captionabove`. Die Verwendung von `enablepatch=caption` ist wichtig für die Funktion der zweiten Option `tablecaptionabove`. Die KOMA-Script-Klassen laden tocbasic daher bereits in der Voreinstellung mit dieser Option.

```
\captionof{Objekttyp}[Verzeichniseintrag]{Titel}
\captionaboveof{Objekttyp}[Verzeichniseintrag]{Titel}
\captionbelowof{Objekttyp}[Verzeichniseintrag]{Titel}
```

Ähnlich wie die Pakete caption und capt-of bietet auch tocbasic 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.

Darüber hinaus bietet tocbasic zusätzlich auch die Anweisungen `\captionaboveof`. Diese dienen als Gegenstücke zu `\captionabove` und `\captionbelow`.

Selbstverständlich berücksichtigt `\captionof` auch die Einstellungen von Option `captionposition`, `captionabove` oder `captionbelow` und der individuellen Optionen für die mit `\DeclareNewTOC` definierten Umgebungen bezüglich der Formatierung des Titels als Über- oder Unterschrift.

**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}{5cm}
    \caption{Ein Rechteck}\label{fig:rechteck}
  \end{minipage}%
  \begin{minipage}{.5\linewidth}
    \centering
    \captionaboveof{table}
    [Maße des Rechtecks aus
      Abbildung~\ref{fig:rechteck}]%
    {Rechtecksmaße}
    \label{tab:rechteck}
    \begin{tabular}{ll}
      Breite: & 4\,cm\\
      Höhe:   & 5\,cm
    \end{tabular}
  \end{minipage}
\end{figure}
```

Um Abbildung und Tabelle nebeneinander zu setzen, wurden zwei `minipage`-Umgebungen verwendet. Wichtig ist das Prozentzeichen nach der ersten `minipage` zur Unterdrückung des Wortabstands zwischen den Umgebungen.

Für die Tabellenüberschrift wurde `\captionaboveof` verwendet. Als erstes Argument wurde `table` angegeben. Dadurch weiß tocbasic, dass es sich trotz `figure`-

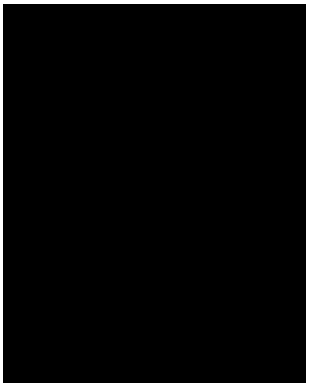


Abbildung 13.5.: Ein Rechteck

Tabelle 13.4.: Rechteckmaße

| | |
|---|---|
| Breite: | 4 cm |
| Höhe: | 5 cm |

Abbildung 13.6.: Verwendung von `\captionaboveof` innerhalb einer fremden Gleitumgebung

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 13.6.

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. Dabei versucht tocbasic auch einen Seitenumbruch zwischen der Überschrift und der Tabelle zu verhindern. Das kann allerdings nicht in allen Fällen sichergestellt werden. Daher 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. Einfacher ist natürlich direkt die Umgebung `table-` zu verwenden.

## 13.8. Nutzung von tocbasic mit Fremdklassen

v3.42

Das Paket tocbasic kann natürlich auch mit anderen Klassen als den KOMA-Script-Klassen verwendet werden. Ab Version 3.42 verfügt das Paket sogar über zusätzliche Möglichkeiten, um die Integration in solche Fremdklassen zu verbessern. Getestet sind diese Möglichkeiten ausschließlich mit den Standardklassen. Da diese neuen Möglichkeiten Eingriffe in die Fremdklasse benötigen, sind sie als invasiv zu betrachten. Aus diesem Grund ist keine der Möglichkeiten in

der Voreinstellung aktiv. Stattdessen müssen sie über Optionen – in einem Fall alternativ per Befehl – aktiviert werden.

Diese neuen Optionen können *nicht* mit \KOMAoptions oder \KOMAoption gesetzt werden. Stattdessen sind sie beim Laden von tocbasic als optionales Argument von \usepackage – bzw. \RequirePackage für Klassen- und Paketautoren – anzugeben. Zum Ausgleich ist es möglich, das Paket mehrfach mit unterschiedlichen Optionen zu laden. Allerdings benötigen die Optionen zwingen eine LaTeX-Version ab 2022/06/01. Mit älteren LaTeX-Versionen sind die neuen Möglichkeiten daher *nicht* verfügbar.

enablepatch=*Patchliste*
disablepatch=*Patchliste*

v3.42

Option enablepatch aktiviert die in der *Patchliste* angegebenen Patches. Dagegen deaktiviert disablepatch die in der *Patchliste* angegebenen Patches. In beiden Fällen ist *Patchliste* eine in einfache geschweifte Klammern eingeschlossene, durch Komma separierte Liste von einzelnen Patches. Eine Liste der verfügbaren Patches ist in Tabelle 13.5 zu finden. Beide Optionen können mehrfach verwendet werden. Dabei wäre beispielsweise

```
\usepackage[enablepatch=chapter,enablepatch=listoffigures]{tocbasic}
```

gleichbedeutend mit

```
\usepackage[enablepatch={chapter,listoffigure}]{tocbasic}
```

Man beachte, dass im zweiten Fall genau eine geschweiftes Klammernpaar für den Wert zu setzen ist. Das Vergessen dieser Klammern führt unweigerlich zu einer Fehlermeldung!

Die Patches selbst werden erst bei \begin{document} ausgeführt. Daher wäre ebenfalls

```
\usepackage[enablepatch=chapter]{tocbasic}
\usepackage[enablepatch=listoffigures]{tocbasic}
```

möglich und würde zu demselben Ergebnis führen.

Eine erste Besonderheit stellt enablepatch=all dar. Dies ist kein Patch im eigentlichen Sinn. Stattdessen sorgt die Einstellung dafür, dass alle Patches unabhängig von ihrer individuellen Einstellung ausgeführt werden. Dies kann mit disablepatch=all wieder aufgehoben werden. Mit disablepatch=all werden also nicht sämtliche Patches deaktiviert, sondern es werden wieder die Einstellungen für die einzelnen Patches beachtet.

Eine zweite Besonderheit stellt enablepatch=caption dar. Da dieser Patch für die korrekte Funktion der Optionen aus Abschnitt 13.7, Seite 406 und den darauf basierenden KOMA-Script-Optionen captions=heading, captions=signature, captions=figureheading, captions=figuresignature, captions=tableheading und captions=tablesignature zwingend notwendig ist, wird er von den KOMA-Script-Klassen automatisch aktiviert. Ohne die Aktivierung haben die genannten Optionen keine definierte Wirkung.

Eine dritte Besonderheit stellt enablepatch=chapter dar. Da dieser Patch auch Schaden anrichten kann, wird seine Verwendung immer in der Terminalausgabe und der log-Datei

Tabelle 13.5.: für tocbasic verfügbare Patches

`all`
: Bei Aktivierung werden alle anderen Patches unabhängig von der individuellen Einstellung ausgeführt. Falls eine KOMA-Script-Klasse geladen ist, gilt dies nicht für `chapter`.

`caption`
v3.46
: Die Anweisungen `\caption` und `\captionof` werden so umdefiniert, dass sie die Optionen `captionposition`, `captionabove`, `captionbelow` beziehungsweise die entsprechenden vom Type oder der Umgebung abhängenden Optionen beachten.

`chapter`
: Über einen generischen Hook wird die Anweisung `\@makechapterhead` so erweitert, dass bei einem neuen Kapitel ähnlich dem Abbildungsverzeichnis und dem Tabellenverzeichnis ein vertikaler Abstand in alle Verzeichnisse der Kategorie bzw. des Besitzers `float` eingefügt wird. Nicht nummerierte Kapitel, die mit `\chapter*` oder `\@makeschapterhead` realisiert werden, sind davon nicht betroffen.

`listoffigures`
: Die Anweisung `\listoffigures` wird so umdefiniert, dass sie durch tocbasic realisiert wird. Damit stehen all von tocbasic via `\setuptoc` bereitgestellten und in Abschnitt 13.2 ab Seite 375 dokumentierten Möglichkeiten zur Verfügung.

`listoftables`
: Die Anweisung `\listoftables` wird so umdefiniert, dass sie durch tocbasic realisiert wird. Damit stehen all von tocbasic via `\setuptoc` bereitgestellten und in Abschnitt 13.2 ab Seite 375 dokumentierten Möglichkeiten zur Verfügung.

`starttoc`
: Es wird die in Abschnitt 13.1, Seite 370 dokumentierte Anweisung `\tocbasicautomode` ausgeführt.

gemeldet. Bei Verwendung einer KOMA-Script-Klasse kann der Patch sogar echten Schaden anrichten. Daher wird in diesem Fall sogar eine Warnung ausgegeben.

Da der `chapter`-Patch mit den KOMA-Script-Klassen normalerweise keinen Sinn ergibt, wird er von `enablepatch=all` bei Verwendung einer KOMA-Script-Klasse zur Vermeidung eines eventuellen Schadens auch nicht aktiviert. Bei Verwendung einer KOMA-Script-Klasse müsste zur Aktivierung dieses Patches also explizit `enablepatch=chapter` verwendet werden. In der Regel ist in diesem Fall davon abzusehen.

**Beispiel:** Den Unterschied zwischen Verwendung von tocbasic mit und ohne Patch ist rasch zu erkennen, wenn wir das `remarkbox`-Beispiel aus Abschnitt 13.6 einmal mit der Standardklasse book betrachten. Laden wir tocbasic zunächst ohne Optionen:

```
\documentclass{book}
```

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

\DeclareNewTOC[%
  type=remarkbox,%
  types=remarkboxes,%
  float,% Gleitumgebungen sollen definiert werden.
  counterwithin=chapter,% Zähler von Kapitel abhängig.
  floattype=4,%
  name=Merksatz,%
  tocentryindent=1.5em,%
  tocentrynumwidth=2.3em,%
  listname={Verzeichnis der Merksätze}%
]{lor}

% Einträge im Inhaltsverzeichnis:
\setuptoc{lof}{totoc}% für das Abbildungsverzeichnis
\setuptoc{lor}{totoc}% für das Verzeichnis der Merksätze

\usepackage{mwe}

\begin{document}
\tableofcontents
\listoffigures
\listofremarkboxes

\chapter{Erstes Beispielkapitel}
\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Erste Abbildung}
\end{figure}
\begin{remarkbox}
  \caption{Erster Merksatz}
\end{remarkbox}

\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Zweite Abbildung}
\end{figure}
\begin{remarkbox}
  \caption{Zweiter Merksatz}
```

```
\end{remarkbox}

\blindtext

\chapter{Zweites Beispielkapitel}
\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Dritte Abbildung}
\end{figure}
\begin{remarkbox}
  \caption{Dritter Merksatz}
\end{remarkbox}

\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Vierte Abbildung}
\end{figure}
\begin{remarkbox}
  \caption{Vierter Merksatz}
\end{remarkbox}

\blindtext

\end{document}
```

Wie in den oberen Reihe von Abbildung 13.7 zu sehen, wird das Abbildungsverzeichnis nicht ins Inhaltsverzeichnis aufgenommen. Die Anweisung `\setuptoc{lof}{totoc}` hat also keine Auswirkungen. Dagegen erscheint das Verzeichnis der Merksätze in Folge von `\setuptoc{lor}{totoc}` sehr wohl im Inhaltsverzeichnis. Zwar erscheint im Abbildungsverzeichnis selbst ein vertikaler Abstand zwischen den Abbildungen von Kapitel 1 und Kapitel 2, nicht jedoch im Verzeichnis der Merksätze.

Die zweite Reihe von Abbildung 13.7 zeigt dagegen wie es sein soll, nämlich mit einem Eintrag sowohl für das Abbildungsverzeichnis als auch das Verzeichnis der Merksätze im Inhaltsverzeichhnis und außerdem mit einem vertikalen Abstand beim Kapitelwechsel in beiden dieser Verzeichnisse. Der Eintrag für das Abbildungsverzeichnis wurde in nachfolgendem Code durch den `listoffigures`-Patch erreicht. Dieser sogt dafür, dass `\setuptoc{lof}{totoc}` tatsächlich beachtet wird. Gleichzeitig sorgt der `chapter`-Patch dafür, dass der Abstand wir gewünscht ins Verzeichnis der Merksätze eingefügt wird.

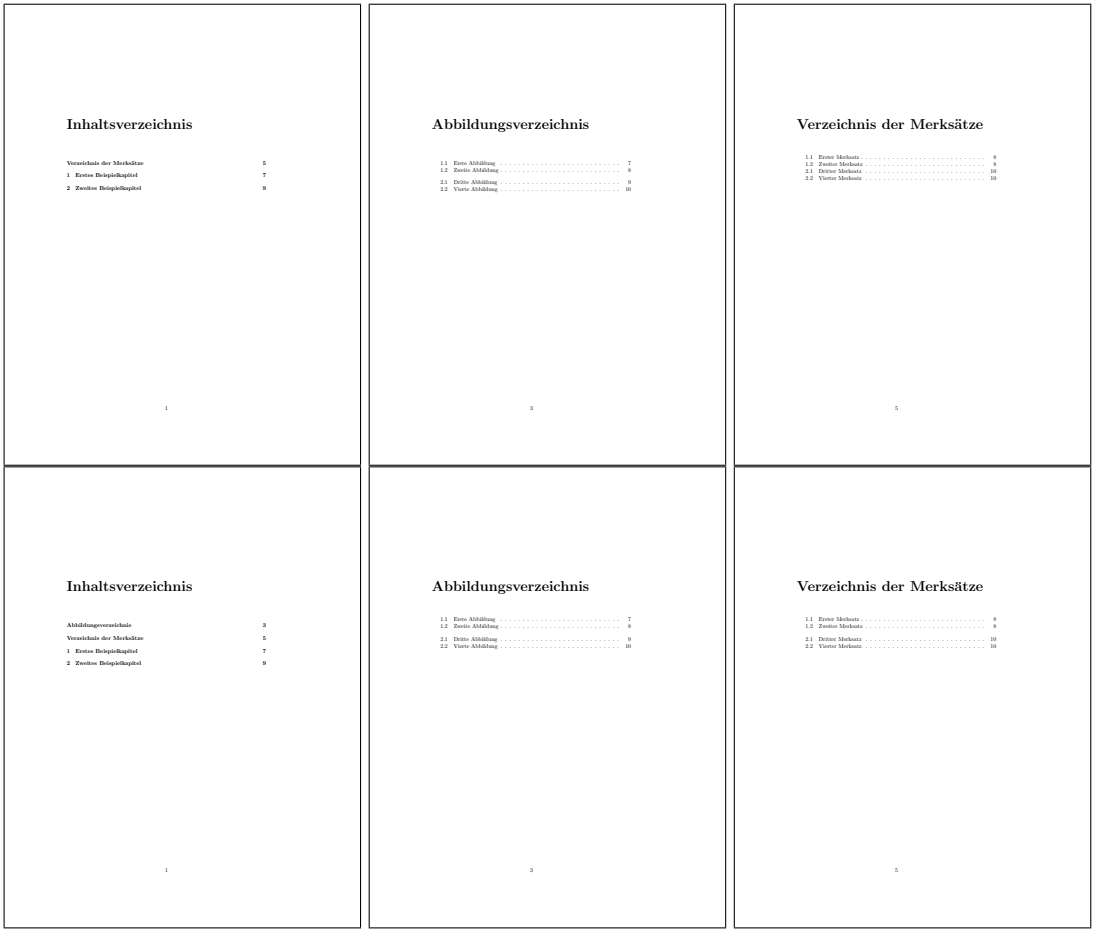



Abbildung 13.7.: Beispiel für tocbasic ohne (oben) und mit (unten) Patch bei Verwendung von Klasse book

```
\documentclass{book}
\usepackage[ngerman]{babel}
\usepackage[enablepatch={chapter,listoffigures}]{tocbasic}

\DeclareNewTOC[%
  type=remarkbox,%
  types=remarkboxes,%
  float,% Gleitumgebungen sollen definiert werden.
  counterwithin=chapter,% Zähler von Kapitel abhängig.
  floattype=4,%
  name=Merksatz,%
  tocentryindent=1.5em,%
```

```
  tocentrynumwidth=2.3em,%
  listname={Verzeichnis der Merksätze}%
]{lor}

% Einträge im Inhaltsverzeichnis:
\setuptoc{lof}{totoc}% für das Abbildungsverzeichnis
\setuptoc{lor}{totoc}% für das Verzeichnis der Merksätze

\usepackage{mwe}

\begin{document}
\tableofcontents
\listoffigures
\listofremarkboxes

\chapter{Erstes Beispielkapitel}
\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Erste Abbildung}
\end{figure}
\begin{remarkbox}
  \caption{Erster Merksatz}
\end{remarkbox}

\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Zweite Abbildung}
\end{figure}
\begin{remarkbox}
  \caption{Zweiter Merksatz}
\end{remarkbox}

\blindtext

\chapter{Zweites Beispielkapitel}
\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Dritte Abbildung}
\end{figure}
```

```
\begin{remarkbox}
  \caption{Dritter Merksatz}
\end{remarkbox}

\blindtext
\begin{figure}
  \centering
  \rule{1cm}{1cm}
  \caption{Vierte Abbildung}
\end{figure}
\begin{remarkbox}
  \caption{Vierter Merksatz}
\end{remarkbox}

\blindtext

\end{document}
```

Ja, in der Tat, der einzige Unterschied zwischen den beiden Beispielen ist das Laden von Paket tocbasic ohne oder mit Option `enablepatch={chapter,listoffigures}`.

## 13.9. Obsolete Befehle

Frühere Versionen von tocbasic verfügten über Befehle, die aufgrund von Äußerungen von Mitgliedern des LaTeX-Project-Teams umbenannt wurden. Diese veralteten Befehle sollten nicht mehr verwendet werden.

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

v3.12

# Definition von Ebenen und Seitenstilen mit scrlayer

Anwender von Grafikprogrammen sind mit dem Modell der Ebenen für eine Seite bereits vertraut. LaTeX selbst ist ein solches Modell jedoch eher fremd. Dennoch gibt es bereits einige Pakete wie eso-pic oder textpos, mit denen eine Art Hintergrund- oder Vordergrundebene in LaTeX verfügbar gemacht wurden. Das Paket scrlayer ist ein weiteres Paket, das solche Hintergrund- und Vordergrundebenen zur Verfügung stellt. Im Unterschied zu den anderen genannten Paketen sind die Ebenen bei scrlayer jedoch Teil des Seitenstils. Dadurch ist eine Umschaltung zwischen der Verwendung unterschiedlicher Ebenen einfach durch die Umschaltung des Seitenstils möglich.

Um dies zu erreichen, stellt das Paket eine Schnittstelle für die Verwaltung Seitenstilen bereit, die auf Ebenen basieren. Über diese können neue Seitenstil als Stapel oder Listen von Ebenen definiert werden. Ebenen können wahlweise am Anfang oder Ende dieser Listen eingefügt werden. Das Einfügen ist ebenfalls vor oder nach einer anderen Ebene in einer solchen Liste möglich. Ebenen können auch wieder aus den Listen entfernt oder Listen von Dupletten bereinigt werden. Natürlich gibt es außerdem Schnittstellen zur Definition und Manipulation der Ebenen selbst.

Nichtsdestoweniger wird die direkte Verwendung der Ebenen nur erfahrenen Anwendern empfohlen. Schnittstellen für Anfänger und durchschnittliche Anwender werden als zusätzliche Pakete wie scrlayer-scrpage angeboten, die dann ihrerseits scrlayer laden. Siehe hierzu Kapitel 5 ab Seite 267 in Teil I dieser Anleitung.

## 14.1. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde. Falls Sie also Abschnitt 2.4 bereits gelesen und verstanden haben, können Sie auf Seite 419 mit Abschnitt 14.2 fortfahren.

```
\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`

v3.00

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 ar-

beiten \documentclass und \usepackage bei KOMA-Script wie in [Tea23] oder jeder LaTeX-Einführung, beispielsweise [DGS+12], beschrieben.

Gegenüber der nachfolgend vorgestellten Schnittstelle zu Einstellungen von Optionen hat \documentclass einen Nachteil, der unbedingt zu beachten ist: Anweisungen, Längen, Zähler und ähnliches können darin leicht zerbrechen. So führt die Verwendung einer LaTeX-Länge im Wert einer Option bei dieser Anweisung bei vielen Nicht-KOMA-Script-Klassen zu einer Fehlermeldung, noch bevor der Wert an ein KOMA-Script-Paket übergeben wird, es also die Kontrolle darüber übernehmen könnte. Wertzuweisungen mit LaTeX-Längen oder LaTeX-Zählern sollten daher nie per \documentclass, sondern mit den nachfolgend dokumentierten Anweisungen \KOMAoptions oder \KOMAoption vorgenommen werden.

\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. Mit der Anweisung \KOMAoptions kann man wie bei \documentclass oder \usepackage 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 \KOMAoption der einen *Option* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *Werteliste* durch Komma voneinander getrennt.

Soll ein *Wert* ein Gleichheitszeichen oder ein Komma enthalten, 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«. Siehe dazu Teil II, Abschnitt 10.2, ab Seite 329.

Mit \KOMAoptions oder \KOMAoption gesetzte Optionen erreichen sowohl die KOMA-Script-Klasse als auch alle bereits geladenen KOMA-Script-Pakete, die diese Optionen kennen. Ist eine Option oder ein Wert insgesamt unbekannt, so wird die Option einschließlich des Wertes von scrbase als fehlerhaft gemeldet.

## 14.2. Einige grundlegende Informationen

Das Paket benötigt einige grundlegende Informationen über die verwendete Klasse. Autoren von Klassen können scrlayer helfen, indem sie entsprechende Angaben machen. Anderenfalls versucht das Paket diese Informationen selbst zu ermitteln. Das funktioniert beispielsweise

für die Standardklassen oder für die KOMA-Script-Klassen. Mit anderen Klassen kann es funktionieren oder auch ganz oder teilweise fehlschlagen. Dieser Abschnitt beschreibt einige der Informationen, die Autoren von Klassen bereitstellen können.

`\if@chapter` *Dann-Code* `\else` *Sonst-Code* `\fi`

Wenn `\if@chapter` definiert ist und `\iftrue` entspricht, berücksichtigt scrlayer bei seiner Arbeit die Kapitel-Ebene beispielsweise bei Verwendung von Option automark. Wenn es definiert ist, aber nicht `\iftrue` entspricht, behandelt scrlayer nur die Ebenen der Befehle `\part`, `\section`, `\subsection`, `\sub...subsection`, `\paragraph`, `\subparagraph`, `\sub...subparagraph`. Wenn das Makro nicht definiert ist, macht scrlayer die Frage, ob auch die Kapitel-Ebene zu behandeln ist, an der Anweisung `\chapter` fest. Ist diese Anweisung definiert und entspricht sie nicht `\relax`, dann definiert scrlayer das Makro `\if@chapter` selbst als Synonym für `\iftrue`. Anderenfalls definiert es `\if@chapter` als Synonym für `\iffalse`.

`\if@mainmatter` *Dann-Code* `\else` *Sonst-Code* `\fi`

Klassen wie book oder scrbook bieten `\frontmatter`, `\mainmatter` und `\backmatter`, um zwischen Vorderteil, Hauptteil und Endteil eines Buches umschalten zu können. In der Regel verwenden diese Klassen intern `\if@mainmatter`, um entscheiden zu können, ob gerade im Hauptteil des Dokuments gearbeitet wird oder nicht. Klassen wie report oder article haben kein `\frontmatter`, `\mainmatter` oder `\backmatter` und deshalb auch kein `\if@mainmatter`. Stattdessen gehen sie davon aus, dass es nur einen Hauptteil gibt.

Für scrlayer ist es aber einfacher, nicht ständig erneut die Existenz und Verwendung der Umschaltanweisungen zu erkennen und damit zu entscheiden, ob nun gerade im Hauptteil gearbeitet wird oder nicht, sondern stattdessen auch bei Klassen wie report oder article mit `\if@mainmatter` zu arbeiten. Das sollte bei den genannten Klassen dann schlicht `\iftrue` entsprechen. Wenn also `\if@mainmatter` nicht definiert ist, dann definiert scrlayer es als Synonym für `\iftrue`.

Einige Klassen definieren jedoch `\frontmatter`, `\mainmatter` oder `\backmatter` und trotzdem kein `\if@mainmatter`. In diesem Fall definiert scrlayer `\if@mainmatter` ebenfalls als Synonym für `\iftrue` und erweitert darüber hinaus die gefundenen Definitionen von `\frontmatter`, `\mainmatter` und `\backmatter` so, dass diese `\if@mainmatter` passend umdefinieren. Falls es jedoch weitere, vergleichbare Befehle zur Umschaltung zwischen unterschiedlichen Dokumentteilen gibt, so kennt scrlayer diese nicht, testet nicht auf diese und erweitert sie daher auch nicht passend. In diesem Fall ist scrlayer also auf die Mitarbeit des Klassenautors angewiesen.

\DeclareSectionNumberDepth{*Name der Gliederungsebene*}{*Tiefe der Gliederungsebene*}

Jeder Gliederungsebene ist normalerweise eine numerische Tiefe zugeordnet. Das ist notwendig, damit LaTeX die Hierarchie der Gliederungsebenen verwalten kann. Allerdings sind die Werte nur der jeweiligen Klasse bekannt, in der die Gliederungsbefehle definiert sind. Diese setzt dann in den entsprechenden LaTeX-Befehlen selbst die zugehörigen Nummern ein.

Das Paket scrlayer benötigt ebenfalls Informationen über die Hierarchie. Mit Hilfe von \DeclareSectionNumberDepth kann scrlayer zum Namen einer Gliederungsebene die zugehörige numerische Tiefe bekannt gemacht werden. Für die Standardklassen wäre *Name der Gliederungsebene* beispielsweise part, chapter, section, subsection, subsubsection, paragraph, subparagraph und die jeweilige *Tiefe der Gliederungsebene* -1, 0, 1, 2, 3, 4, 5.

Das Paket scrlayer versucht, diese numerischen Werte zunächst beim Laden des Pakets und noch einmal während \begin{document} selbst zu ermitteln. Aber für den Fall, dass dies einmal nicht zu einem korrekten Ergebnis führt, beispielsweise falls es vollkommen andere Gliederungsbefehle gibt, kann man die Zuordnung eben mit \DeclareSectionNumberDepth auch explizit vornehmen.

## 14.3. Deklaration von Ebenen

Eine Ebene (engl. *layer*) ist ein Denkmodell für eine Seite. Im Gegensatz zu echtem, physischem Papier ist diese Seite vollständig transparent. Üblicherweise werden mehrere Ebenen übereinander gestapelt und undurchsichtiges Material auf einer Ebene überdeckt Material auf den Ebenen darunter. Ein solcher Stapel von Ebenen wird dann auf eine reale Seite Papier abgebildet. Das Paket scrlayer stellt zwei solche Stapel für jede Seite zur Verfügung: einen Hintergrundstapel und einen Vordergrundstapel. Der Hintergrundstapel befindet sich unter oder hinter dem normalen Seiteninhalt, während der Vordergrundstapel über oder vor dem normalen Seiteninhalt ausgegeben wird. Der normale Seiteninhalt ist daher eine Art von Trennebene zwischen den beiden Ebenenstapeln.

Eine Ebene hat mehrere Eigenschaften, die als Antworten auf grundlegende Fragen verstanden werden können:

**Gehört die Ebene zum Vordergrund oder zum Hintergrund?** Hintergrundebenen werden ausgegeben, bevor der normale Inhalt der Seite gedruckt wird. Optisch erscheinen sie daher *hinter* oder *unter* dem normalen Inhalt der Seite. Vordergrundebenen werden an den normalen Inhalt anschließend ausgegeben. Optisch erscheinen sie daher *vor*, *auf* oder *über* dem normalen Inhalt der Seite. In der Voreinstellung ist eine Ebene sowohl eine Hintergrundebene als auch eine Vordergrundebene und wird daher zweimal ausgegeben. In der Regel ist es deshalb sinnvoll, dies explizit einzuschränken.

**Wie groß ist die Ebene?** Eine Ebene kann auch kleiner oder größer als das Papier sein. Damit werden Eigenschaften für die horizontale und vertikale Ausdehnung der Ebene benötigt.

**An welcher Position soll die Ebene ausgegeben werden?** Diese Frage stellt sich nicht nur aufgrund der von der Papiergröße möglichweise abweichenden Ausdehnung der Ebene. Auch gleich große Ebenen können gegeneinander verschoben sein. Ihre Beantwortung führt zu Eigenschaften für die Festlegung der horizontalen und vertikalen Position der Ebene.

**Wie werden die horizontale und die vertikale Position gemessen?** Diese Frage ergibt sich unmittelbar aus der vorherigen. Die Antwort darauf ist die Eigenschaft der Ausrichtung. Man kann von der linken Papierkante zur linken Kante der Ebene, zur Mitte der Ebene oder zur rechten Kante der Ebene messen. Entsprechend kann man von der oberen Kante des Papiers zur oberen Kante der Ebene, zur Mitte der Ebene oder zur unteren Kante der Ebene messen.

**Ist die Ebene für Textausgabe oder für Grafik vorgesehen?** Auch diese Frage ist eng mit der Position verknüpft. Während der Anwender bei der Grafikausgabe davon ausgeht, dass der Ursprung in der linken unteren Ecke der Ebene liegt, wäre dies bei der Textausgabe eher ungünstig. Daher liegt der Ursprung für Textebenen um die Höhe einer Standardtextzeile unterhalb der oberen, linken Ecke der Ebene. Grafikebenen wiederum spannen von sich aus bereits eine `picture`-Umgebung auf, in der zusätzliche Befehle zur Positionierung zur Verfügung stehen.

**Soll die Ebene auf linken oder rechten Seiten eines Dokuments gedruckt werden?** In der Voreinstellung wird eine Ebene auf allen Seiten gedruckt. Es ist zu beachten, dass LaTeX im doppelseitigen Satz Seiten mit geraden Seitenzahlen als linke Seiten und Seiten mit ungeraden Seitenzahlen als rechte Seiten behandelt, dass es jedoch im einseitigen Modus unabhängig von der Nummer nur rechte Seiten gibt. LaTeX bezeichnet, den Gepflogenheiten der englischen Sprache entsprechend, linke Seiten auch als gerade Seiten und rechte Seiten als ungerade Seiten. Dies erfolgt ungeachtet der Tatsache, dass im einseitigen Satz ungerade Seiten dann auch gerade Nummern tragen können.

**Soll die Ebene in einseitigen oder in doppelseitigen Dokumenten verwendet werden?** In der Voreinstellung ist die Ebene diesbezüglich unbeschränkt, wird also sowohl im einseitigen als auch im doppelseitigen Modus ausgegeben. Nichtsdestotrotz wird eine Ebene, die auf gerade Seiten beschränkt ist, im einseitigen Modus niemals ausgegeben werden und ist daher auch keine einseitige Ebene.

**Soll die Ebene auf Gleitseiten oder auf Normalseiten ausgegeben werden?** LaTeX erzeugt Gleitseiten für Objekte aus Umgebungen wie `table` oder `figure`, wenn diesen erlaubt wurde, auf eigenen Seiten ohne Teile des normalen Dokumentinhalts ausgegeben zu werden (siehe Option `p` für `table` oder `figure`). In gewisser Weise ist es so der gesamten Seite erlaubt, im Dokument zu gleiten. Normalseiten in diesem Sinne sind alle Seiten, die keine Gleitseiten sind. Normalseiten können ebenfalls Gleitumgebungen am Anfang,

im Inneren oder am Ende enthalten. Sehr große Gleitumgebungen können auch den Eindruck einer Gleitseite erzeugen, obwohl es sich bei ihnen in Wirklichkeit um oben auf einer Normalseite platzierte Gleitumgebungen handelt.

**Welchen Inhalt hat die Ebene?** Die zugehörige Eigenschaft gibt schlicht an, was gedruckt werden soll, wann immer die Ebene ausgegeben wird.

Damit haben wir derzeit acht Fragen an die Ebenen, aus denen sich unmittelbar eine Reihe von Eigenschaften ergeben. Später in dieser Anleitung werden wir weitere Eigenschaften kennenlernen, die jedoch auf diese primären Eigenschaften abgebildet werden können.

```
\DeclareLayer[Optionenliste]{Name der Ebene}
\DeclareNewLayer[Optionenliste]{Name der Ebene}
\ProvideLayer[Optionenliste]{Name der Ebene}
\RedeclareLayer[Optionenliste]{Name der Ebene}
\ModifyLayer[Optionenliste]{Name der Ebene}
```

Diese Anweisungen können verwendet werden, um Ebenen zu definieren oder zu verändern. Der *Name der Ebene* muss voll expandierbar sein. Die Expansion sollte in ASCII-Buchstaben resultieren. Einige zusätzliche Zeichen werden ebenfalls akzeptiert, ihre Verwendung wird jedoch nur erfahrenen Anwendern empfohlen.

Bei Verwendung von `\DeclareLayer` spielt es keine Rolle, ob eine Ebene *Name der Ebene* bereits existiert oder nicht. Sie wird in jedem Fall mit den über die *Optionenliste* angegebenen Eigenschaften definiert. Einzelne Optionen bestehen entweder nur aus einem Schlüssel oder aus einem Schlüssel, gefolgt von einem Gleichheitszeichen und einem Wert. Die Optionen sind durch Komma voneinander getrennt. Um innerhalb der Werte einer Option ein Komma oder ein Leerzeichen verwenden zu können, muss der entsprechende Wert in geschweifte Klammern gesetzt werden. Eine Übersicht über die Optionen und die Eigenschaften, die sie repräsentieren, findet sich in Tabelle 14.1.

Tabelle 14.1.: Optionen für die Beschreibung von Seiten-Ebenen mit ihrer jeweiligen Bedeutung als Ebenen-Eigenschaft

| | |
|---|---|
| v3.16 | `addcontents=`*Code*<br>Der angegebene Wert wird an den aktuellen Wert des Attributs `contents` angehängt. Es wird also ein zusätzlicher Inhalt generiert. Zu näheren Informationen über die Behandlung von *Code* siehe Option `contents`. |
| v3.16 | `addheight=`*zusätzliche Höhe*<br>Der aktuelle Wert von Attribut `height` wird um den Wert dieser Option erhöht. Als Wert sind die gleichen Angaben wie bei `height` möglich. |

...

Tabelle 14.1.: Optionen für die Beschreibung von Ebenen (*Fortsetzung*)

v3.16 `addhoffset=`*zusätzlicher horizontaler Abstand*

Der aktuelle Wert von Attribut `hoffset` wird um den Wert dieser Option erhöht. Als Wert sind die gleichen Angaben wie bei `hoffset` möglich.

v3.16 `addvoffset=`*zusätzlicher vertikaler Abstand*

Der aktuelle Wert von Attribut `voffset` wird um den Wert dieser Option erhöht. Als Wert sind die gleichen Angaben wie bei `voffset` möglich.

v3.16 `addwidth=`*zusätzliche Breite*

Der aktuelle Wert von Attribut `width` wird um den Wert dieser Option erhöht. Als Wert sind die gleichen Angaben wie bei `width` möglich.

`align=`*Ausrichtungszeichen*

Über die *Ausrichtungszeichen* wird die gewünschte Ausrichtung der Ebene bestimmt. Dabei steht jedes einzelne *Ausrichtungszeichen* für eine mögliche Anwendung der Werte *Abstand* der Optionen `hoffset` oder `voffset`. Mehrere *Ausrichtungszeichen* können ohne Leerzeichen oder Komma direkt hintereinander geschrieben werden und werden in der Reihenfolge ihres Auftretens ausgewertet. Makros sind im Wert der Option jedoch nicht zulässig. Zulässige *Ausrichtungszeichen* sind:

- `b` – der Wert der Option `voffset` ist der Abstand der Unterkante der Ebene von der Oberkante des Papiers.
- `c` – die Werte der Optionen `hoffset` und `voffset` sind die Abstände des Zentrums der Ebene von der linken und der oberen Kante des Papiers.
- `l` – der Wert der Option `hoffset` ist der Abstand der linken Kante der Ebene von der linken Kante des Papiers.
- `r` – der Wert der Option `hoffset` ist der Abstand der rechten Kante der Ebene von der linken Kante des Papiers.
- `t` – der Wert der Option `voffset` ist der Abstand der Oberkante der Ebene von der Oberkante des Papiers.

`area={`*horizontaler Abstand*`}{`*vertikaler Abstand*`}{`*Breite*`}{`*Höhe*`}`

Die zusammengesetzte Eigenschaft resultiert in den primären Eigenschaften `hoffset=`*horizontaler Abstand*, `voffset=`*vertikaler Abstand*, `width=`*Breite*, `height=`Höhe.

...

Tabelle 14.1.: Optionen für die Beschreibung von Ebenen (*Fortsetzung*)

v3.18

`backandforeground`
: Mit dieser Option wird die Einschränkung der Ebene auf den Vorder- oder Hintergrund wieder aufgehoben und diesbezüglich die Grundeinstellung wiederhergestellt. In der Regel ist dies wenig sinnvoll, daher existiert die Option nur aus Gründen der Vollständigkeit. Diese Option erwartet und erlaubt keinen Wert.

`background`
: Mit dieser Option wird die Ebene zu einer reinen Hintergrundebene. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr gleichzeitig im Hintergrund und im Vordergrund, sondern nur noch im Hintergrund ausgegeben. Diese Option erwartet und erlaubt keinen Wert.

`bottommargin`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene horizontal von der linken Kante bis zur rechten Kante des Papiers reicht und vertikal den gesamten Bereich unter dem Seitenfuß bis zur unteren Papierkante abdeckt. Diese Option erwartet und erlaubt keinen Wert.

`clone=`*`Name einer Ebene`*
: Die zusammengesetzte Eigenschaft setzt alle primären Eigenschaften entsprechend der aktuellen, primären Eigenschaften der Ebene mit dem angegebenen *`Name einer Ebene`*. Bezüglich *`Name einer Ebene`* siehe die Hinweise zu *`Name der Ebene`* am Anfang der Erklärung zu `\DeclareLayer`. Darüber hinaus muss die zu klonende Ebene bereits definiert sein.

`contents=`*`Code`*
: Der angegebene *`Code`* wird immer dann expandiert und ausgeführt, wenn die Ebene ausgegeben wird. Damit definiert *`Code`* das, was auf der Ebene zu sehen ist. Es werden keine Tests durchgeführt, ob *`Code`* gültig und korrekt ist. Fehler in *`Code`* können daher zu verschiedenen Fehlermeldungen auf jeder Seite führen, auf der die Ebene ausgegeben wird.

`evenpage`
: Mit dieser Option wird die Ebene zu einer Ebene für linke Seiten. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr sowohl auf linken als auch auf rechten Seiten ausgegeben. Da es linke Seiten nur im doppelseitigen Satz gibt, schließt diese Eigenschaft quasi `twoside` mit ein. Diese Option erwartet und erlaubt keinen Wert.

...

Tabelle 14.1.: Optionen für die Beschreibung von Ebenen (*Fortsetzung*)

v3.18 `everypage`
: Dies ist eine Kombination von `oddorevenpage` und `floatornonfloatpage`. Diese Option erwartet und erlaubt keinen Wert.

v3.18 `everyside`
: Mit dieser Option wird die Einschränkung der Ebene auf den einseitigen oder den doppelseitigen Satz aufgehoben. Sie wird damit wie in der Voreinstellung wieder sowohl im einseitigen als auch im doppelseitigen Satz ausgegeben. Diese Option erwartet und erlaubt keinen Wert.

v3.18 `floatornonfloatpage`
: Mit dieser Option wird die Einschränkung der Ebene auf Gleitseiten oder Seiten, die keine Gleitseiten sind, aufgehoben und diesbezüglich die Voreinstellung wiederhergestellt. Diese Option erwartet und erlaubt keinen Wert.

`floatpage`
: Mit dieser Option wird die Ebene zu einer Gleitseitenebene. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr auf allen Seiten, sondern nur noch auf Gleitseiten ausgegeben. Näheres zu Gleitseiten ist der Einleitung zu diesem Abschnitt zu entnehmen. Diese Option erwartet und erlaubt keinen Wert.

`foot`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den Seitenfuß in der Breite des Textbereichs überdeckt. Diese Option erwartet und erlaubt keinen Wert.

`footskip`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene vertikal den Bereich zwischen dem Textbereich und dem Seitenfuß in Breite des Textbereichs überdeckt. Es ist zu beachten, dass die Höhe dieses Bereichs zwar von der Länge `\footskip` abhängt, dieser jedoch nicht entspricht. Diese Option erwartet und erlaubt keinen Wert.

`foreground`
: Mit dieser Option wird die Ebene zu einer reinen Vordergrundebene. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr gleichzeitig im Hintergrund und im Vordergrund, sondern nur noch im Vordergrund ausgegeben. Diese Option erwartet und erlaubt keinen Wert.

...

Tabelle 14.1.: Optionen für die Beschreibung von Ebenen (*Fortsetzung*)

`head`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den Seitenkopf in der Breite des Textbereichs überdeckt. Diese Option erwartet und erlaubt keinen Wert.

`headsep`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den Abstand zwischen dem Seitenkopf und dem Textbereich in der Breite des Textbereichs überdeckt. Ihre Höhe entspricht damit der Länge `\headsep`. Diese Option erwartet und erlaubt keinen Wert.

`height=`*`Höhe`*
: Setzt die *`Höhe`* der Ebene. Beachten Sie, dass *`Höhe`* wahlweise eine LaTeX-Länge sein kann, die mit `\newlength` definiert wurde, eine TeX-Länge, die mit `\newdimen` oder `\newskip` definiert wurde, ein Längenwert wie 10 pt oder ein Längenausdruck unter Verwendung von +, -, /, *, (, und ). Die genaue Syntax eines Längenausdrucks ist [Tea98, Abschnitt 3.5] zu entnehmen.

`hoffset=`*`Abstand`*
: Setzt den *`Abstand`* der Ebene von der linken Kante des Papiers. Wie der *`Abstand`* gemessen wird, hängt von Eigenschaft `align` ab. Beachten Sie, dass *`Abstand`* wahlweise eine LaTeX-Länge sein kann, die mit `\newlength` definiert wurde, eine TeX-Länge, die mit `\newdimen` oder `\newskip` definiert wurde, ein Längenwert wie 10 pt oder ein Längenausdruck unter Verwendung von +, -, /, *, (, und ). Die genaue Syntax eines Längenausdrucks ist [Tea98, Abschnitt 3.5] zu entnehmen.

`innermargin`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den inneren Rand der Seite von der Papieroberkante bis zur Papierunterkante überdeckt. Der innere Rand entspricht im einseitigen Satz dem linken Rand. Diese Option erwartet und erlaubt keinen Wert.

`leftmargin`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den linken Rand der Seite von der Papieroberkante bis zur Papierunterkante überdeckt. Diese Option erwartet und erlaubt keinen Wert.

...

Tabelle 14.1.: Optionen für die Beschreibung von Ebenen (*Fortsetzung*)

v3.19 `mode=`*`Modus`*

Diese primäre Eigenschaft bestimmt, in welchem *`Modus`* der Inhalt der Ebene ausgegeben wird. Die Voreinstellung ist `text`. Dabei wird die oberste Grundlinie um die Höhe einer Standardtextzeile unterhalb der Oberkante der Ebene platziert. Damit ist Text normalerweise sauber am oberen Rand der Ebene ausgerichtet. Im `picture`-*`Modus`* wird hingegen eine `picture`-Umgebung mit dem Ursprung in der linken, unteren Ecke der Ebene aufgespannt. Der ebenfalls vordefinierte *`Modus`* `raw` entspricht in der Voreinstellung `text`.

Die Änderung des *`Modus`* einer Ebene führt in der Regel zu einer Verschiebung des Inhalts. Außerdem stehen beispielsweise im *`Modus`* `picture` zusätzliche Platzierungsbefehle zur Verfügung, die in einem anderen *`Modus`* zu Fehlermeldungen führen. Daher ist es normalerweise nicht sinnvoll, den *`Modus`* einer Ebene nachträglich zu ändern!

`nonfloatpage`

Mit dieser Option wird die Ebene auf Seiten beschränkt, die keine Gleitseiten sind. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr auf allen Seiten, sondern nur noch auf Nichtgleitseiten ausgegeben. Näheres zu Gleitseiten und Nichtgleitseiten ist der Einleitung zu diesem Abschnitt zu entnehmen. Diese Option erwartet und erlaubt keinen Wert.

v3.18 `oddorevenpage`

Mit dieser Option werden Beschränkungen der Ebene auf rechte oder linke Seiten aufgehoben. Damit wird die Ebene wie in der Voreinstellung sowohl auf linken als auch rechten Seiten ausgegeben. Diese Option erwartet und erlaubt keinen Wert.

`oddpage`

Mit dieser Option wird die Ebene zu einer Ebene für rechte Seiten. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr sowohl auf linken als auch auf rechten Seiten ausgegeben. Es ist zu beachten, dass im einseitigen Satz alle Seiten unabhängig von der Seitenzahl rechte Seiten sind. Diese Option erwartet und erlaubt keinen Wert.

`oneside`

Mit dieser Option wird die Ebene zu einer Ebene für den einseitigen Satz. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr sowohl im einseitigen als auch im doppelseitigen Satz ausgegeben. Diese Option erwartet und erlaubt keinen Wert.

...

Tabelle 14.1.: Optionen für die Beschreibung von Ebenen (*Fortsetzung*)

`outermargin`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den äußeren Rand der Seite von der Papieroberkante bis zur Papierunterkante überdeckt. Der äußere Rand entspricht im einseitigen Satz dem rechten Rand. Diese Option erwartet und erlaubt keinen Wert.

`page`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene die komplette Seite überdeckt. Diese Option erwartet und erlaubt keinen Wert.

v3.16 `pretocontents=`*`Code`*
: Der angegebene Wert wird dem aktuellen Wert des Attributs `contents` vorangestellt. Es wird also ein zusätzlicher Inhalt vor dem bisherigen Inhalt generiert. Zu näheren Informationen über die Behandlung von *`Code`* siehe Option `contents`.

`rightmargin`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den rechten Rand der Seite von der Papieroberkante bis zur Papierunterkante überdeckt. Diese Option erwartet und erlaubt keinen Wert.

`textarea`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den kompletten Textbereich überdeckt. Diese Option erwartet und erlaubt keinen Wert.

`topmargin`
: Die zusammengesetzte Eigenschaft setzt die primären Eigenschaften `hoffset`, `voffset`, `width`, `height` und `align` so, dass die Ebene den oberen Rand der Seite von der linken Kante des Papiers bis zu dessen rechter Kante überdeckt. Diese Option erwartet und erlaubt keinen Wert.

`twoside`
: Mit dieser Option wird die Ebene zu einer Ebene für den doppelseitigen Satz. Sie wird also im Gegensatz zur Grundeinstellung nicht mehr sowohl im einseitigen als auch im doppelseitigen Satz ausgegeben. Diese Option erwartet und erlaubt keinen Wert.

...

Tabelle 14.1.: Optionen für die Beschreibung von Ebenen (*Fortsetzung*)

v3.18 `unrestricted`
: Hebt alle Ausgabebeschränkungen auf. Damit ist die Option eine Kombination von `backandforeground`, `everyside` und `floatornonfloatpage`. Diese Option erwartet und erlaubt keinen Wert.

`voffset=`*`Abstand`*
: Setzt den *`Abstand`* der Ebene von der Papieroberkante. Wie der *`Abstand`* gemessen wird, hängt von Eigenschaft `align` ab. Beachten Sie, dass *`Abstand`* wahlweise eine LaTeX-Länge sein kann, die mit `\newlength` definiert wurde, eine TeX-Länge, die mit `\newdimen` oder `\newskip` definiert wurde, ein Längenwert wie 10 pt oder ein Längenausdruck unter Verwendung von +, -, /, *, (, und ). Die genaue Syntax eines Längenausdrucks ist [Tea98, Abschnitt 3.5] zu entnehmen.

`width=`*`Breite`*
: Setzt die *`Breite`* der Ebene. Beachten Sie, dass *`Breite`* wahlweise eine LaTeX-Länge sein kann, die mit `\newlength` definiert wurde, eine TeX-Länge, die mit `\newdimen` oder `\newskip` definiert wurde, ein Längenwert wie 10 pt oder ein Längenausdruck unter Verwendung von +, -, /, *, (, und ). Die genaue Syntax eines Längenausdrucks ist [Tea98, Abschnitt 3.5] zu entnehmen.

Im Unterschied zu `\DeclareLayer` meldet `\DeclareNewLayer` einen Fehler, falls eine Ebene mit dem angegebenen Namen bereits existiert. Damit wird der Anwender davor bewahrt, versehentlich mehrmals denselben Namen zu verwenden. Dies ist insbesondere auch dann nützlich, wenn Klassen oder Pakete intern ebenfalls Ebenen definieren.

Dagegen definiert `\ProvideLayer` die Ebene nur, wenn nicht bereits eine Ebene mit dem angegebenen Namen existiert. Wird der Name hingegen bereits für eine andere Ebene verwendet, so wird die neuerliche Definition ignoriert. Die Anweisung hat also die Bedeutung: *Definiere die Ebene, falls sie noch nicht existiert.*

Soll eine bereits existierende Ebene umdefiniert werden, so kann wahlweise `\RedeclareLayer` oder `\ModifyLayer` verwendet werden. Während mit `\RedeclareLayer` die Ebene zunächst auf die Grundeinstellungen zurückgesetzt und damit über die angegebene *`Optionenliste`* komplett neu definiert wird, unterbleibt bei `\ModifyLayer` das Zurücksetzen. Es werden dann nur die Eigenschaften geändert, für die in der *`Optionenliste`* auch Angaben vorhanden sind. Die Anwendung auf eine zuvor noch nicht definierte Ebene stellt bei beiden Anweisungen einen Fehler dar.

`\ModifyLayers[`*`Optionenliste`*`]{`*`Ebenenliste`*`}`

v3.26 Diese Anweisung führt `\ModifyLayer{`*`Optionenliste`*`}{`*`Ebene`*`}` für jede *`Ebene`* aus, die in der durch Komma separierten *`Ebenenliste`* angegeben ist. Sie dient also dazu, die Eigenschaften einer ganzen Reihe von Ebenen gleichzeitig auf gleiche Weise zu ändern.

```
\layerhalign
\layervalign
\layerxoffset
\layeryoffset
\layerwidth
\layerheight
```

Diese Anweisungen sind nur im mit `contents`, `addcontents` oder `pretocontents` angegebenen *`Code`* gültig. Sie enthalten die tatsächlich verwendete Ausrichtung, Position und Ausdehnung der Ebene während deren Ausgabe. Dies ist jedoch nicht zwangsläufig auch die tatsächliche Ausdehnung des Inhalts, falls dieser beispielsweise überbreit oder überhoch ist oder die Ebene nicht komplett ausfüllt.

v3.19 Die primäre Ebeneneigenschaft `align` wird auf `\layerhalign` und `\layervalign` abgebildet. Dabei werden die horizontalen Werte `l` und `r` nur in `\layerhalign` übernommen, während die vertikalen Werte `t` und `b` nur in `\layervalign` übernommen werden. Der sowohl horizontale als auch vertikale Wert `c` wird in beide Anweisungen übernommen. Sind bei `align` mehrere, widersprüchliche Angaben zu finden, so gewinnt die jeweils letzte. Damit ist also `\layerhalign` immer entweder `l`, `c` oder `r` und `\layervalign` immer entweder `t`, `c` oder `b`.

Eine Umdefinierung der Anweisungen und damit Änderung der in ihnen gespeicherten Werte ist nicht gestattet und führt zu unvorhersehbaren Ergebnissen.

```
\LenToUnit{Länge}
```

v3.19 Diese Anweisung stammt ursprünglich von eso-pic ab Version 2.0f. Sie rechnet Längen-Werte in Vielfache von `\unitlength` um und kann daher anstelle von Koordinaten oder anderen von `\unitlength` abhängigen Werten einer `picture`-Umgebung verwendet werden. Siehe dazu auch [Nie15] und die nachfolgende Erklärung zu `\putUL` bis `\putC`. Die Anweisung wird nur definiert, wenn sie nicht bereits, beispielsweise durch das Laden von eso-pic, definiert ist.

```
\putUL{Inhalt}
\putUR{Inhalt}
\putLL{Inhalt}
\putLR{Inhalt}
\putC{Inhalt}
```

v3.19 Diese Anweisungen können innerhalb der primären Ebeneneigenschaft `contents` verwendet werden, wenn die Ebene mit `mode=picture` erstellt wurde. In diesem Fall platziert `\putUL` den *`Inhalt`* relativ zur oberen, linken Ecke der Ebene und entspricht damit `\put(0,\LenToUnit{\layerheight})`. `\putUR` platziert *`Inhalt`* relativ zur oberen, rechten Ecke der Ebene und entspricht damit `\put(\LenToUnit{\layerwidth},\LenToUnit{\layerheight})`. `\putLL` platziert *`Inhalt`* relativ zur unteren, linken Ecke der Ebene und entspricht damit `\put(0,0)`. `\putLR`

platziert *Inhalt* relativ zur unteren, rechten Ecke der Ebene und entspricht damit \put(\LenToUnit{\layerwidth},0). \putC schlussendlich platziert *Inhalt* relativ zur Mitte der Ebene.

**Beispiel:** Sie wollen feststellen, wie genau die Höhe des Textbereichs bei DIV=classic tatsächlich der Breite der Seite entspricht, und erstellen dazu eine Ebene, die sowohl den Textbereich umrandet als auch einen Kreis mit der Papierbreite als Durchmesser im Zentrum des Textbereichs platziert:

```
\documentclass[DIV=classic]{scrartcl}
\usepackage{pict2e}
\usepackage{scrlayer}
\DeclareNewLayer[%
  textarea,background,mode=picture,
  contents={%
    \putLL{\line(1,0){\LenToUnit{\layerwidth}}}%
    \putLR{\line(0,1){\LenToUnit{\layerheight}}}%
    \putUR{\line(-1,0){\LenToUnit{\layerwidth}}}%
    \putUL{\line(0,-1){\LenToUnit{\layerheight}}}%
    \putC{\circle{\LenToUnit{\paperwidth}}}%
  }
]{showtextarea}
\DeclareNewPageStyleByLayers{test}{showtextarea}
\pagestyle{test}
\begin{document}
\null
\end{document}
```

Wie Sie sehen werden, passt die von typearea vorgenommene Abbildung auf einen ganzzahligen *DIV*-Wert im Beispiel sehr gut.

Näheres zu dem im Beispiel skizzierten spätmittelalterlichen Buchseitenkanon finden Sie übrigens in Abschnitt 2.3, Seite 30.

Die Anweisung \DeclareNewPageStyleByLayers, die im Beispiel bereits verwendet wurde, dient der Definition eines Seitenstils, der die neu definiert Ebene ausgibt. Sie wird in Abschnitt 14.4, Seite 436 erklärt werden.

\GetLayerContents{*Name der Ebene*}

v3.16

Mit dieser Anweisung kann der aktuelle Inhalt einer Ebene ermittelt werden. Es ist unbedingt zu beachten, dass bei Verwendung dieser Anweisung im *Code* der Ebenen-Attribute contents, addcontents oder pretocontents unendliche Rekursionen entstehen können, wenn dabei auf den Inhalt der aktuellen Ebene zugegriffen wird. Der Anwender ist selbst dafür verantwortlich, solche Situationen zu vermeiden!

`\IfLayerExists{`*Name der Ebene*`}{`*Dann-Code*`}{`*Sonst-Code*`}`

Diese Anweisung kann dazu verwendet werden, Code in Abhängigkeit davon, ob eine Ebene existiert oder nicht, auszuführen. Wenn die Ebene *Name der Ebene* existiert, so wird der *Dann-Code* ausgeführt, anderenfalls der *Sonst-Code*. Bitte beachten Sie, dass die Anweisung nicht wirklich testen kann, ob eine Ebene existiert. Sie verwendet stattdessen Heuristiken, die niemals falsch-negativ sein können, jedoch im Extremfall falsch-positiv sein könnten. Falsch-positive Entscheidungen weisen auf ein Problem, beispielsweise die Verwendung eines inkompatiblen Pakets oder ungünstige Wahl von internen Makronamen durch den Anwender.

`\DestroyLayer{`*Name der Ebene*`}`

Existiert eine Ebene *Name der Ebene*, so werden alle zu dieser Ebene gehörenden Makros zu `\relax`. Die Ebene kann nicht länger verwendet werden. In bereits mit scrlayer definierten Seitenstils werden derart zerstörte Ebenen ignoriert. Zerstörte Ebenen können mit `\DeclareNewLayer` oder `\ProvideLayer` neu definiert werden. Sie können jedoch vor einer neuerlichen Definition nicht länger mit `\RedeclareLayer` oder `\ModifyLayer` verändert werden.

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

Mit Hilfe der KOMA-Script-Option `draft` kann für das Paket scrlayer ein Entwurfsmodus aktiviert werden. In diesem Entwurfsmodus wird hinter jeder Ebene zunächst eine Bemaßung der Ebene ausgegeben. Diese Bemaßung erfolgt mit `\layercontentsmeasure`. Diese Anweisung zeigt am oberen und linken Rand der Ebene ein Maßband in Zentimeter und am rechten und unteren Rand der Ebene ein Maßband in Zoll. Die Anweisung `\layercontentsmeasure` kann statt über die Option auch schlicht als alleiniger *Code* für die Eigenschaft `contents` einer Ebene verwendet werden.

## 14.4. Deklaration und Verwaltung von Seitenstilen

Wir kennen nun Ebenen und wissen, wie diese definiert und verwaltet werden. Aber bisher wissen wir kaum, wie sie verwendet werden. Die möglicherweise überraschende Antwort lautet: mit Hilfe von Seitenstilen. Üblicherweise werden Seitenstile in LaTeX zur Definition von Kopf und Fuß der Seite verwendet.

Kopf und Fuß für ungerade oder rechte Seiten werden im doppelseitigen Modus auf Seiten mit ungerader Seitenzahl ausgegeben. Im einseitigen Modus werden sie auf allen Seiten verwendet. Das ist unmittelbar mit den Optionen `oddpage` und `evenpage` für Ebenen vergleichbar.

Der Seitenkopf wird vor dem normalen Seiteninhalt ausgegeben. Der Seitenfuß wird entsprechend nach dem normalen Seiteninhalt ausgegeben. Dies korrespondiert also unmittelbar mit den Optionen `background` und `foreground` für Ebenen.

Daher liegt es nahe, Seitenstile als Listen von Ebenen zu definieren. Aber statt nur den genannten vier Optionen können dabei alle Eigenschaften verwendet werden, die in Abschnitt 14.3, Tabelle 14.1, ab Seite 423 erklärt wurden.

Als Ergebnis dieser Überlegungen ist eine Form von Seitenstilen, die scrlayer bietet, der Ebenen-Seitenstil. Ein solcher Ebenen-Seitenstil besteht aus Ebenen und zusätzlich aus mehreren Haken (engl. *hooks*). Die Ebenen wurden bereits in Abschnitt 14.3 beschrieben. Die Haken sind Punkte in der Expansion oder Anwendung von Seitenstilen, zu denen zusätzlicher Code hinzugefügt werden kann. Erfahrene Anwender kennen dieses Konzept bereits von beispielsweise `\AtBeginDocument` (siehe [Tea26]) oder `\BeforeClosingMainAux` (siehe Seite 359).

Eine zweite Form von Seitenstilen, die scrlayer bietet, ist der Alias-Seitenstil oder Seitenstil-Alias. Ein Seitenstil-Alias besteht in Wirklichkeit aus einem anderen Seitenstil. Anders ausgedrückt ist der Name eines Seitenstil-Alias ein Alias-Name für einen anderen Seitenstil-Alias oder einen primären Seitenstil. Daher führt die Manipulation an einem Seitenstil-Alias zu einer Manipulation am originären Seitenstil. Ist der originäre Seitenstil selbst ebenfalls ein Seitenstil-Alias, so führt dessen Manipulation wiederum zu einer Manipulation dessen originären Seitenstils und immer so weiter, bis schließlich ein realer Seitenstil verändert wird. Der Ausdruck *realer Seitenstil* wird zur Unterscheidung von einem Seitenstil-Alias verwendet. Alle Seitenstile, die kein Seitenstil-Alias sind, sind reale Seitenstile. Seitenstil-Aliase können nicht nur für Seitenstile definiert werden, die mit scrlayer definiert wurden, sondern für alle Seitenstile.

`\currentpagestyle`
`\toplevelpagestyle`

Das Paket scrlayer erweitert die LaTeX-Anweisung `\pagestyle` so, dass diese `\currentpagestyle` als den Namen des jeweils aktiven Seitenstils definiert. Es ist zu beachten, dass `\thispagestyle` selbst `\currentpagestyle` nicht verändert. Wird `\thispagestyle` verwendet, so kann sich `\currentpagestyle` aber innerhalb der LaTeX-Ausgabefunktion verändern. Dies hat jedoch nur dann Auswirkungen, wenn `\currentpagestyle` bis in die LaTeX-Ausgabefunktion geschützt verwendet wird.

Es sei darauf hingewiesen, dass die später in diesem Abschnitt dokumentierten Ebenen-Seitenstile nicht auf diese Erweiterung von `\pagestyle` angewiesen sind, da sie selbst auch `\currentpagestyle` umdefinieren. Die Erweiterung wurde für die Verwendung von anderen Seitenstilen, die nicht auf scrlayer basieren, vorgenommen. Es ist außerdem zu beachten, dass `\currentpagestyle` vor der ersten Verwendung von `\pagestyle` nach dem Laden von scrlayer leer ist. Bei der Definition einer Endanwender-Schnittstelle dürfte es daher nützlich sein, mit einer impliziten `\pagestyle`-Anweisung den aktuellen Seitenstil auf eine Voreinstellung zu setzen.

v3.16 Wird mit `\pagestyle` ein Alias-Seitenstils aktiviert, so liefert `\currentpagestyle` nicht den Alias-Namen, sondern den des originären Seitenstils. Den Alias-Namen kann man in diesem Fall mit `\toplevelpagestyle` erhalten. Es wird davon abgeraten, Seitenstile zu definieren,

die beispielsweise per \Ifstr abhängig von \toplevelpagestyle unterschiedliche Ergebnisse liefern, da dies bei Aktivierung per \thispagestyle zu falschen Resultaten führen kann.

\BeforeSelectAnyPageStyle{*Code*}
\AfterSelectAnyPageStyle{*Code*}

Die Anweisung \BeforeSelectAnyPageStyle fügt einem Haken (engl. *hook*) *Code* hinzu, der innerhalb der Ausführung von Anweisung \pagestyle, unmittelbar vor der Auswahl des Seitenstils ausgeführt wird. Innerhalb von *Code* kann #1 als Platzhalter für das Argument von \pagestyle verwendet werden.

Die Anweisung \AfterSelectAnyPageStyle arbeitet ähnlich. Allerdings wird hier *Code* ausgeführt, nachdem der Seitenstil gewählt und \currentpagestyle auf den Namen des realen Seitenstils gesetzt wurde.

Es ist zu beachten, dass *Code* jeweils nur bei der Wahl eines Seitenstils mit Hilfe von \pagestyle ausgeführt wird. Wird ein Seitenstil auf andere Art, beispielsweise mit Hilfe von \thispagestyle, gewählt, so wird *Code* nicht ausgeführt. Es ist außerdem zu beachten, dass einmal hinzugefügter *Code* nicht mehr entfernt werden kann. Allerdings wird der *Code* lokal hinzugefügt. Sein Gültigkeitsbereich kann daher mit einer Gruppe beschränkt werden.

\DeclarePageStyleAlias{*Seitenstil-Alias-Name*}{*originärer Seitenstil-Name*}
\DeclareNewPageStyleAlias{*Seitenstil-Alias-Name*}{*originärer Seitenstil-Name*}
\ProvidePageStyleAlias{*Seitenstil-Alias-Name*}{*originärer Seitenstil-Name*}
\RedeclarePageStyleAlias{*Seitenstil-Alias-Name*}{*originärer Seitenstil-Name*}

Diese Anweisungen können verwendet werden, um einen Seitenstil mit dem Namen *Seitenstil-Alias-Name* zu definieren, der einfach nur ein Alias für einen bereits existierenden Seitenstil mit dem Namen *originärer Seitenstil-Name* ist. Falls bereits ein Seitenstil mit dem Namen *Seitenstil-Alias-Name* existiert, wird dieser vor der Erzeugung des Alias mit \DeclarePageStyleAlias oder \RedeclarePageStyleAlias zerstört.

Die Anweisung \DeclareNewPageStyleAlias erzeugt eine Fehlermeldung, falls zuvor bereits ein Seitenstil *Seitenstil-Alias-Name* definiert wurde. Dabei spielt es keine Rolle, ob der existierende Seitenstil selbst ein Alias-Seitenstil, ein Ebenen-Seitenstil oder eine andere Art von Seitenstil ist.

Die Anweisung \ProvidePageStyleAlias definiert den Seitenstil-Alias nur, falls nicht bereits ein Seitenstil *Seitenstil-Alias-Name* existiert. Falls ein solcher Seitenstil existiert, bleibt dieser erhalten und die Anweisung tut schlicht nichts.

Im Gegensatz zu den drei vorgenannten Anweisungen erwartet \RedeclarePageStyleAlias, dass bereits ein Seitenstil mit dem Namen *Seitenstil-Alias-Name* existiert. Anderenfalls erzeugt die Anweisung eine Fehlermeldung.

\DestroyPageStyleAlias{*Seitenstil-Alias-Name*}

Mit dieser Anweisung wird der Seitenstil-Alias mit dem angegebenen Namen *Seitenstil-Alias-Name* für LaTeX wieder undefiniert, wenn es tatsächlich einen Alias-Seitenstil dieses Namens gibt. Anschließend kann der Seitenstil auch mit \DeclareNewPageStyleAlias oder \ProvidePageStyleAlias neu definiert werden.

\GetRealPageStyle{*Seitenstil-Name*}

Diese Anweisung sucht rekursiv nach dem tatsächlichen Namen eines Seitenstils, wenn der angegebene *Seitenstil-Name* zu einem Alias-Seitenstil gehört. Ist *Seitenstil-Name* nicht der Name eines Alias-Seitenstils, so ist das Ergebnis *Seitenstil-Name* selbst. Das gilt auch, falls ein Seitenstil namens *Seitenstil-Name* gar nicht existiert. Die Anweisung ist voll expandierbar und kann damit beispielsweise auch im zweiten Argument einer \edef-Anweisung verwendet werden.

\DeclarePageStyleByLayers[*Optionenliste*]{*Seitenstil-Name*}{*Ebenenliste*}
\DeclareNewPageStyleByLayers[*Optionenliste*]{*Seitenstil-Name*}{*Ebenenliste*}
\ProvidePageStyleByLayers[*Optionenliste*]{*Seitenstil-Name*}{*Ebenenliste*}
\RedeclarePageStyleByLayers[*Optionenliste*]{*Seitenstil-Name*}{*Ebenenliste*}

Diese Anweisungen deklarieren einen Seitenstil mit dem Namen *Seitenstil-Name*. Der Seitenstil besteht aus einer Anzahl von Ebenen, die in der mit Komma separierten *Ebenenliste* angegeben sind. Es ist zu beachten, dass sowohl *Seitenstil-Name* als auch *Ebenenliste* voll expandierbar sein müssen und die Expansion zu einer Reihe von Buchstaben führen sollte. Einige zusätzliche Zeichen werden ebenfalls akzeptiert, ihre Verwendung wird jedoch nur erfahrenen Anwendern empfohlen.

Die *Optionenliste* ist eine mit Komma separierte Liste von Optionen der Form *Schlüssel=Code*. Diese Optionen können verwendet werden, um zusätzliche Eigenschaften zu setzen und zusätzliche Möglichkeiten zu nutzen. Derzeit werden sie verwendet, um Code an bestimmten Stellen der Aktivierung oder Verwendung eines Seitenstils über Haken (engl. *hooks*) auszuführen. Für allgemeine Informationen zu Haken sei auf die Einleitung zu diesem Abschnitt verwiesen. Details zu den Haken und ihrer Bedeutung sind Tabelle 14.2 zu entnehmen. Es ist zu beachten, dass die Haken auch für den nachfolgend erklärten Pseudo-Seitenstil @everystyle@ ausgeführt werden.

Tabelle 14.2.: Optionen und gleichnamige Haken für Ebenen-Seitenstile (in der Reihenfolge ihrer Abarbeitung)

`onselect=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn der Seitenstil, beispielsweise mit `\pagestyle`, ausgewählt wird. Es ist zu beachten, dass `\thispagestyle` selbst keinen Seitenstil unmittelbar auswählt, sondern der Seitenstil in diesem Fall erst innerhalb der Ausgaberoutine von LaTeX aktiviert wird.

`oninit=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn die Ausgabe der Ebenen für den Seitenstil initialisiert wird. Beachten Sie, dass dies für jede Seite zweimal geschieht: einmal für Hintergrund-Ebenen und einmal für Vordergrund-Ebenen.

`ononeside=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn im einseitigen Modus die Ausgabe der Ebenen für den Seitenstil initialisiert wird. Beachten Sie, dass dies für jede Seite zweimal geschieht: einmal für Hintergrund-Ebenen und einmal für Vordergrund-Ebenen.

`ontwoside=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn im doppelseitigen Modus die Ausgabe der Ebenen für den Seitenstil initialisiert wird. Beachten Sie, dass dies für jede Seite zweimal geschieht: einmal für Hintergrund-Ebenen und einmal für Vordergrund-Ebenen.

`onoddpage=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn die Ausgabe der Ebenen für den Seitenstil auf einer rechten Seite initialisiert wird. Beachten Sie, dass dies für jede Seite zweimal geschieht: einmal für Hintergrund-Ebenen und einmal für Vordergrund-Ebenen. Beachten Sie außerdem, dass im einseitigen Modus alle Seiten rechte Seiten sind.

`onevenpage=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn die Ausgabe der Ebenen für den Seitenstil auf einer linken Seite initialisiert wird. Beachten Sie, dass dies für jede Seite zweimal geschieht: einmal für Hintergrund-Ebenen und einmal für Vordergrund-Ebenen. Beachten Sie außerdem, dass im einseitigen Modus keine linken Seiten existieren.

...

Tabelle 14.2.: Optionen für die Haken von Ebenen-Seitenstilen (*Fortsetzung*)

`onfloatpage=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn die Ausgabe der Ebenen für den Seitenstil auf einer Gleitumgebungsseite initialisiert wird. Beachten Sie, dass dies für jede Seite zweimal geschieht: einmal für Hintergrund-Ebenen und einmal für Vordergrund-Ebenen. Beachten Sie außerdem, dass Gleitumgebungsseiten nur diejenigen Seiten sind, auf denen eine oder mehrere p-platzierte Gleitumgebungen ausgegeben werden.

`onnonfloatpage=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn die Ausgabe der Ebenen für den Seitenstil auf einer Seite initialisiert wird, die keine Gleitumgebungsseite ist. Beachten Sie, dass dies für jede Seite zweimal geschieht: einmal für Hintergrund-Ebenen und einmal für Vordergrund-Ebenen. Beachten Sie außerdem, dass Gleitumgebungsseiten nur diejenigen Seiten sind, auf denen eine oder mehrere p-platzierte Gleitumgebungen ausgegeben werden, und auf anderen Seiten sehr wohl t-, b- oder h-platzierte Gleitumgebungen stehen können.

`onbackground=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn die Ausgabe der Ebenen für den Hintergrund einer Seite initialisiert wird. Beachten Sie, dass dies auf jeder Seite genau einmal der Fall ist.

`onforeground=`*`Code`*
: Der *`Code`* dieses Hakens wird immer dann ausgeführt, wenn die Ausgabe der Ebenen für den Vordergrund einer Seite initialisiert wird. Beachten Sie, dass dies auf jeder Seite genau einmal der Fall ist.

Die Anweisung `\DeclarePageStyleByLayers` definiert den Seitenstil unabhängig davon, ob bereits ein Seitenstil des Namens *`Seitenstil-Name`* existiert. Gegebenenfalls wird der existierende Seitenstil komplett neu definiert. Falls bereits ein Alias-Seitenstil *`Seitenstil-Name`* existiert wird jedoch nicht der Alias-Seitenstil selbst neu definiert, sondern der zugehörige reale Seitenstil (siehe `\GetRealPageStyle` zuvor in diesem Abschnitt).

Die Anweisung `\DeclareNewPageStyleByLayers` unterscheidet sich in dem Fall, dass bereits ein gleichnamiger Seitenstil existiert. Unabhängig davon, ob es sich um einen realen Seitenstil oder einen Alias-Seitenstil handelt, wird in diesem Fall ein Fehler gemeldet.

Im Unterschied dazu wird bei `\ProvidePageStyleByLayers` der Seitenstil unverändert erhalten, wenn bereits ein Seitenstil des Namens *`Seitenstil-Name`* existiert. Existiert kein solcher Seitenstil, so wird er wie bei `\DeclarePageStyleByLayers` definiert.

Die Anweisung `\RedeclarePageStyleByLayers` wiederum erwartet, dass bereits ein Seitenstil des Namens *`Seitenstil-Name`* existiert und definiert dessen realen Seitenstil dann

um. Existiert jedoch noch kein Seitenstil des angegebenen Namens, so resultiert daraus eine Fehlermeldung.

```
\pagestyle{@everystyle@}
\pagestyle{empty}
```

Das Paket scrlayer definiert von sich aus bereits zwei in gewisser Weise spezielle Seitenstile. Der erste davon ist `@everystyle@`. Dieser Seitenstil sollte niemals als normaler Seitenstil, beispielsweise mit `\pagestyle` oder `\thispagestyle`, oder als Ziel eines Alias-Seitenstils verwendet werden. Stattdessen werden die Ebenen und Haken dieses Seitenstils von allen anderen Ebenen-Seitenstilen mit verwendet. Dabei werden die Haken von `@everystyle@` jeweils vor den entsprechenden Haken und die Ebenen jeweils vor den entsprechenden Ebenen des aktiven Seitenstils ausgeführt.

Damit ist das Hinzufügen einer Ebene zum Pseudo-Seitenstil `@everystyle@` oder von Code zu einem Haken dieses Seitenstils vergleichbar mit dem Hinzufügen einer Ebene beziehungsweise von Haken-Code zu allen Ebenen-Seitenstile. Der eine entscheidende Unterschied ist: Befehle, die sich auf die Ebenen eines Seitenstils beziehen, das sind neben `\ForEachLayerOfPageStyle` beispielsweise auch die Anweisungen `\AddLayersToPageStyleBeforeLayer` oder `\AddLayersToPageStyleAfterLayer`, lassen die Ebenen des Seitenstils `@everystyle@` unberücksichtigt, wenn sie auf einen anderen Ebenen-Seitenstil angewendet werden.

Der zweite etwas andere Ebenen-Seitenstil ist `empty`. Bereits vom LaTeX-Kern wird ein Seitenstil dieses Namens definiert, der einen leeren Kopf und Fuß hat. Das Paket scrlayer definiert diesen Seitenstil als Ebenen-Seitenstil ohne Ebenen um. Der Hauptvorteil dieses Ebenen-Seitenstils gegenüber dem ursprünglichen Seitenstil aus dem LaTeX-Kern ist, dass er ebenfalls die Haken und Ebenen des Pseudo-Seitenstils `@everystyle@` ausführt.

```
onpsselect=Code
onpsinit=Code
onpsoneside=Code
onpstwoside=Code
onpsoddpage=Code
onpsevenpage=Code
onpsfloatpage=Code
onpsnonfloatpage=Code
onpsbackground=Code
onpsforeground=Code
```

Für jeden der Haken aus Tabelle 14.2 existiert außerdem eine KOMA-Script-Option, in deren Namen lediglich ein `ps` eingefügt wurde. Die Werte dieser KOMA-Script-Optionen werden als Anfangswerte für die entsprechenden Haken verwendet. Dieser Anfangswert wird

dann um alle Werte, die dem entsprechenden Haken in der *Optionenliste* der Deklarationsbefehle zugewiesen werden, erweitert. Der Anfangswert kann mit Hilfe der Anweisung \ModifyLayerPageStyleOptions, die später in diesem Abschnitt erklärt wird, entfernt werden.

singlespacing=*Ein-Aus-Wert*

v3.24 Wird ein Dokument, beispielsweise mit Hilfe von Paket setspace, mit erhöhtem Zeilenabstand gesetzt, ist es oft dennoch nicht erwünscht, dass Kopf und Fuß der Seite ebenfalls mit diesem erhöhten Zeilenabstand gesetzt werden. Das gilt umso mehr, wenn Kopf und Fuß nur aus jeweils einer Zeile bestehen. In diesem Fall kann man KOMA-Script-Option singlespacing setzen. In der Voreinstellung ist die Option jedoch nicht gesetzt! Die Option wirkt generell für alle Ebenen-Seitenstile. Will man hingegen nur einige Ebenen-Seitenstil einzeilig gesetzt haben, so sollte man stattdessen für diese Seitenstile oninit=\linespread{1}\selectfont verwenden.

deactivatepagestylelayers=*Ein-Aus-Wert*
\ForEachLayerOfPageStyle{*Seitenstil-Name*}{*Code*}
\ForEachLayerOfPageStyle*{*Seitenstil-Name*}{*Code*}

Solange KOMA-Script-Option deactivatepagestylelayers nicht aktiviert ist, kann mit \ForEachLayerOfPageStyle für jede Ebene des Seitenstils mit dem Namen *Seitenstil-Name* beliebiger *Code* ausgeführt werden. Innerhalb von *Code* dient dabei #1 als Platzhalter für den Namen der gerade abgearbeiteten Ebene.

**Beispiel:** Angenommen, Sie wollen die Namen aller Ebenen des Seitenstils scrheadings als Komma-separierte Liste, so können Sie dies mit

```
\newcommand*\commaatlist{}
\ForEachLayerOfPageStyle{scrheadings}{%
  \commaatlist#1\gdef\commaatlist{, }}
\let\commaatlist\relax
```

erreichen.

Die Verwendung von \gdef anstelle von \def ist im Beispiel notwendig, weil *Code* innerhalb einer Gruppe ausgeführt wird, um unerwünschte Seiteneffekte zu minimieren. Die Anweisung \gdef definiert \commaatlist jedoch global um, so dass beim Aufruf des Codes für die nächste Ebene die Änderung Bestand hat.

v3.18 Alternativ hätte man auch zwar mit \def, dafür aber mit der Sternvariante \ForEachLayerOfPageStyle* arbeiten können. Diese Form verzichtet bei der Ausführung von *Code* auf eine Gruppe. Allerdings muss der Anwender dann selbst sicherstellen, dass *Code* keine unerwünschten Seiteneffekte hat. Insbesondere würde die Deaktivierung der Ebenen mit deactivatepagestylelayers=true innerhalb von *Code* dann über den Aufruf von \ForEachLayerOfPageStyle* hinaus Bestand haben.

Diverse Anweisungen von scrlayer setzen intern selbst ebenfalls `\ForEachLayerOfPageStyle` ein. Auch deren Funktion kann daher über die KOMA-Script-Option `deactivatepagestylelayers` verändert werden. Diese Option kann also verwendet werden, um alle Ebenen aller Seitenstile temporär zu deaktivieren und damit zu verstecken.

`\AddLayersToPageStyle{`*Seitenstil-Name*`}{`*Ebenenliste*`}`
`\AddLayersAtBeginOfPageStyle{`*Seitenstil-Name*`}{`*Ebenenliste*`}`
`\AddLayersAtEndOfPageStyle{`*Seitenstil-Name*`}{`*Ebenenliste*`}`
`\RemoveLayersFromPageStyle{`*Seitenstil-Name*`}{`*Ebenenliste*`}`

Diese Anweisungen können verwendet werden, um Ebenen zu einem Seitenstil hinzuzufügen oder davon zu entfernen. Der Seitenstil wird dabei über *Seitenstil-Name* referenziert. Die Ebenen werden in einer durch Komma separierten *Ebenenliste* angegeben.

Sowohl die Anweisung `\AddLayersToPageStyle` als auch die Anweisung `\AddLayersAtEndOfPageStyle` fügt die Ebenen am Ende der Ebenenliste des Seitenstils ein. Logisch liegen die neu hinzugefügten Ebenen also über oder vor den bereits vorhandenen Ebenen, wobei Hintergrund-Ebenen natürlich weiterhin logisch hinter der Textebene und damit auch hinter allen Vordergrund-Ebenen liegen.

Die Anweisung `\AddLayersAtBeginOfPageStyle` fügt die Ebenen hingegen am Anfang der Ebenenliste des Seitenstils ein. Dabei werden die Ebenen in der Reihenfolge am Anfang eingefügt, in der sie auch in der *Ebenenliste* stehen. Damit wird die Ebene, die ganz am Ende von *Ebenenliste* steht, nach dem Einfügen die erste und damit die unterste oder hinterste Ebene (jeweils entweder der Vordergrund- oder der Hintergrundebenen) sein.

Werden nicht existente Ebenen zu einem Seitenstil hinzugefügt, werden diese bei der Anwendung des Stils ignoriert.

Der Versuch, mit Hilfe von `\RemoveLayersFromPageStyle` Ebenen von einem Seitenstil zu entfernen, die gar nicht Teil des Seitenstils sind, wird ignoriert, führt also nicht zu einer Fehlermeldung. Dagegen ist der Versuch, Ebenen zu einem Seitenstil, der kein Ebenen-Seitenstil ist und auch kein Alias-Seitenstil, der zu einem Ebenen-Seitenstil führt, hinzuzufügen oder von einem solchen zu entfernen, ein Fehler und wird als solcher gemeldet.

`\AddLayersToPageStyleBeforeLayer{`*Seitenstil-Name*`}{`*Ebenenliste*`}{`*Referenzebenen-Name*`}`
`\AddLayersToPageStyleAfterLayer{`*Seitenstil-Name*`}{`*Ebenenliste*`}{`*Referenzebenen-Name*`}`

Die Befehle ähneln den vorherigen. Die existierenden Ebenen des Seitenstils werden jedoch nach *Referenzebenen-Name* durchsucht. Die Ebenen der *Ebenenliste* werden dann vor respektive nach jedem Auftreten der Referenzebene eingefügt. Dabei bleibt die Reihenfolge der Ebenen der *Ebenenliste* erhalten.

Ist die Referenzebene nicht Bestandteil des Seitenstils, so wird auch nichts eingefügt. Ist der Seitenstil hingegen kein Ebenen-Seitenstil und auch kein Alias-Seitenstil, der zu einem Ebenen-Seitenstil führt, so wird ein Fehler gemeldet.

\UnifyLayersAtPageStyle{*Seitenstil-Name*}

Bei den Befehlen zur Definition eines Seitenstils oder zum Hinzufügen von Ebenen zu einem Seitenstil wird nicht darauf geachtet, ob eine Ebene mehrfach Bestandteil eines Seitenstils ist oder wird. Dies ist also durchaus zulässig. In den meisten Fällen hat es allerdings keinen Sinn, eine Ebene mehrfach als Bestandteil eines Seitenstils zu haben. Daher kann mit Hilfe von \UnifyLayersAtPageStyle dafür gesorgt werden, dass alle Ebenen-Dubletten vom Seitenstil mit dem angegebenen *Seitenstil-Name* entfernt werden.

Es ist zu beachten, dass sich dabei die Reihenfolge der Ebenen ändern kann. Wird also eine spezielle Reihenfolge gewünscht, sollten stattdessen alle Ebenen entfernt und die gewünschten Ebenen in der erwarteten Reihenfolge neu hinzugefügt werden. In einem solchen Fall ist \UnifyLayersAtPageStyle also nicht geeignet.

\ModifyLayerPageStyleOptions{*Seitenstil-Name*}{*Optionenliste*}
\AddToLayerPageStyleOptions{*Seitenstil-Name*}{*Optionenliste*}

Mit diesen beiden Anweisungen können die Optionen und damit die Haken eines Ebenen-Seitenstils nachträglich verändert werden. Bei \ModifyLayerPageStyleOptions werden dabei genau die Optionen, die in der durch Komma separierten *Optionenliste* angegeben sind, auf die dortigen neuen Werte gesetzt. Die bisherigen Werte gehen dabei verloren. Es sind alle Optionen aus Tabelle 14.2, Seite 437 erlaubt. Optionen beziehungsweise Haken, die nicht in der *Optionenliste* angegeben sind, bleiben hingegen unverändert. Diese Anweisung ist damit auch die einzige Möglichkeit, die globalen Voreinstellungen der KOMA-Script-Optionen von einem Seitenstil zu entfernen.

Die Anweisung \AddToLayerPageStyleOptions überschreibt dagegen die bisher vorhandenen Werte nicht, sondern fügt die neuen zu den bisherigen hinzu oder – genauer gesagt – hängt die neuen Werte an die alten an.

\IfLayerPageStyleExists{*Seitenstil-Name*}{*Dann-Code*}{*Sonst-Code*}
\IfRealLayerPageStyleExists{*Seitenstil-Name*}{*Dann-Code*}{*Sonst-Code*}

Mit den Anweisungen kann Code in Abhängigkeit davon ausgeführt werden, ob ein Seitenstil ein Ebenen-Seitenstil ist oder nicht. Dabei führt \IfLayerPageStyleExists den *Dann-Code* nur dann aus, wenn *Seitenstil-Name* der Name eines Ebenen-Seitenstils oder der Name eines Alias-Seitenstils ist, der zu einem Ebenen-Seitenstil führt. Anderenfalls führt die Anweisung den *Sonst-Code* aus.

Die Anweisung \IfRealLayerPageStyleExists geht einen Schritt weiter und führt den *Dann-Code* nur dann aus, wenn der über *Seitenstil-Name* angegebene Seitenstil selbst ein Ebenen-Seitenstil ist. Im Falle eines Alias-Seitenstils führt diese Anweisung also selbst dann *Sonst-Code* aus, wenn dieser Alias-Seitenstil zu einem Ebenen-Seitenstil führt.

```
\IfLayerAtPageStyle{Seitenstil-Name}{Ebenen-Name}{Dann-Code}{Sonst-Code}
\IfSomeLayersAtPageStyle{Seitenstil-Name}{Ebenenliste}{Dann-Code}{Sonst-Code}
\IfLayersAtPageStyle{Seitenstil-Name}{Ebenenliste}{Dann-Code}{Sonst-Code}
```

Mit diesen Anweisungen kann überprüft werden, ob ein oder mehrere Ebenen Bestandteil eines Seitenstils sind. `\IfLayerAtPageStyle` erwartet dabei den *Seitenstil-Name* im ersten Argument und im zweiten Argument genau einen *Ebenen-Name*. Ist die entsprechende Ebene Bestandteil des Seitenstils, so wird der *Dann-Code* ausgeführt, anderenfalls der *Sonst-Code*.

Im Unterschied dazu erlauben `\IfSomeLayersAtPageStyle` und `\IfLayersAtPageStyle` im zweiten Argument eine durch Komma separierte *Ebenenliste*. Dabei führt `\IfSomeLayersAtPageStyle` den *Dann-Code* bereits aus, wenn *mindestens eine* der Ebenen Bestandteil des Seitenstils ist. Dagegen führt `\IfLayersAtPageStyle` den *Dann-Code* nur aus, wenn *alle* Ebenen Bestandteil des Seitenstils sind. Ist die Bedingung nicht erfüllt, so wird jeweils *Sonst-Code* ausgeführt.

Durch geeignete Schachtelung sind auch komplexe Bedingungen abbildbar. Gibt man statt einer *Ebenenliste* jeweils nur einen *Ebenen-Name* an, so sind alle drei Anweisungen gleichbedeutend.

```
\DestroyRealLayerPageStyle{Ebenen-Seitenstil-Name}
```

Mit dieser Anweisung kann ein Ebenen-Seitenstil aus LaTeX-Sicht wieder undefiniert werden. Dies gilt jedoch nur, wenn es einen Ebenen-Seitenstil mit dem Namen *Ebenen-Seitenstil-Name* auch tatsächlich gibt. Handelt es sich stattdessen um einen Alias-Seitenstil – auch wenn dieser zu einem Ebenen-Seitenstil führt – oder um einen anderen Seitenstil oder ist gar kein Seitenstil mit dem Namen *Ebenen-Seitenstil-Name* bekannt, so geschieht nichts, die Anweisung wird ignoriert.

Falls *Ebenen-Seitenstil-Name* der Name des aktuellen Seitenstils ist, so wird dieser auf eine Art leeren Seitenstil gesetzt. Falls der mit `\thispagestyle` gesetzte Seitenstil *Ebenen-Seitenstil-Name* lautet, so wird dieser einfach nur zurückgesetzt. Die vorherige Anweisung `\thispagestyle` verliert damit ihre aktuelle Auswirkung.

Es ist zu beachten, dass die Ebenen des Seitenstils nicht automatisch mit vernichtet werden. Falls Sie die Ebenen ebenfalls vernichten möchten, so können Sie dies beispielsweise mit

```
\ForEachLayerOfPageStyle{...}{\DestroyLayer{#1}}
```

vor der Vernichtung des Seitenstils selbst erreichen.

## 14.5. Höhe von Kopf und Fuß

Es gilt sinngemäß, was in Abschnitt 5.2 geschrieben wurde. Falls Sie also Abschnitt 5.2 bereits gelesen und verstanden haben, können Sie unten mit Abschnitt 14.6 fortfahren.

```
\footheight
\headheight
```

Das Paket scrlayer führt als neue Länge `\footheight` analog zur Höhe `\headheight` des LaTeX-Kerns ein. Gleichzeitig interpretiert scrlayer `\footskip` so, dass es den Abstand der letzten Grundlinie des Textbereichs von der ersten Standard-Grundlinie des Fußes darstellt. Das Paket typearea betrachtet dies in gleicher Weise, so dass die dortigen Optionen zum Setzen der Höhe des Fußes (siehe die Optionen `footheight` und `footlines` in Abschnitt 2.6, Seite 45) und zur Berücksichtigung des Fußes bei der Berechnung des Satzspiegels (siehe Option `footinclude` in demselben Abschnitt, Seite 42) sehr gut zum Setzen der Werte für scrlayer verwendet werden können und auch das gewünschte Ergebnis liefern.

Wird das Paket typearea nicht verwendet, so sollte man gegebenenfalls die Höhe von Kopf und Fuß über entsprechende Werte für die Längen einstellen. Zumindest für den Kopf bietet aber beispielsweise auch das Paket geometry Einstellmöglichkeiten.

Wurde der Kopf oder Fuß für den tatsächlich verwendeten Inhalt zu klein gewählt, so toleriert scrlayer dies in der Regel ohne Fehlermeldung oder Warnung. Der Kopf dehnt sich dann entsprechend seiner Höhe in der Regel weiter nach oben, der Fuß entsprechend weiter nach unten aus. Informationen darüber erhält man jedoch nicht automatisch. Pakete wie scrlayer-scrpage, die auf scrlayer aufbauen, enthalten dagegen gegebenenfalls eigene Tests, die auch zu eigenen Aktionen führen können (siehe `\headheight` und `\footheight` auf Seite 269).

## 14.6. Beeinflussung von Seitenstilen

Obwohl scrlayer selbst keine konkreten Seitenstile mit Inhalt definiert – die erwähnten Seitenstile `@everystyle@` und `empty` werden ja zunächst ohne Ebenen, also leer definiert –, stellt es einige Optionen und Befehle zur Beeinflussung von Inhalten zur Verfügung.

```
automark
autooneside=Ein-Aus-Wert
manualmark
\automark[Gliederungsebene der rechten Marke]{Gliederungsebene der linken Marke}
\automark*[Gliederungsebene der rechten Marke]{Gliederungsebene der linken Marke}
\manualmark
```

Bei den meisten Klassen bestimmt die Wahl des Seitenstils, meist `headings` und `myheadings`, darüber, ob die Kolumnentitel automatisch oder manuell erzeugt werden. Bei scrlayer erfolgt die Unterscheidung stattdessen über die beiden Anweisungen `\automark` und `\manualmark`.

Mit `\manualmark` wird dabei auf manuelle Marken umgeschaltet. Es deaktiviert also das automatische Setzen der Marken. Demgegenüber kann mit `\automark` und `\automark*` festgelegt werden, welche Gliederungsebenen für das automatische Setzen der Marken verwendet werden sollen. Das optionale Argument gibt dabei die *Gliederungsebene der rechten Marke* an, während das obligatorische Argument die *Gliederungsebene der linken Marke* ist. Als

Argument werden jeweils die Namen der Gliederungsebenen angegeben, also `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph` oder `subparagraph`.

Normalerweise sollte die höhere Ebene die linke Marke setzen, während die tiefere Ebene für die rechte Marke zu verwenden ist. Diese übliche Konvention ist jedoch keine Pflicht, sondern lediglich sinnvoll.

Es ist zu beachten, dass nicht alle Klassen Kolumnentitel für alle Ebenen ermöglichen. So setzen die Standardklassen beispielsweise nie Kolumnentitel für `\part`. Die KOMA-Script-Klassen unterstützen hingegen alle Ebenen.

Der Unterschied zwischen `\automark` und `\automark*` liegt darin, dass `\automark` alle vorherigen Befehle zum automatischen Setzen der Marken aufhebt, während die Stern-Version `\automark*` lediglich die Aktionen für die angegebenen Gliederungsebenen ändert.

Außer mit den Befehlen kann auch direkt mit den beiden Optionen `manualmark` und `automark` zwischen automatischen und manuellen Kolumnentiteln hin- und hergeschaltet werden. Dabei verwendet `automark` bei Klassen mit `\chapter`-Anweisung immer die Voreinstellung

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

und bei anderen Klassen:

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

Im einseitigen Modus will man in der Regel, dass nur die höheren Ebenen den Kolumnentitel vorgeben. Diese Voreinstellung entspricht einer aktiven Option `autooneside`. Die Option versteht die Werte für einfache Schalter, die in Tabelle 2.5 auf Seite 41 angegeben sind. Wird die Option deaktiviert, so wirken sich im einseitigen Satz sowohl das optionale als auch das obligatorische Argument auf den Kolumnentitel aus.

Das Laden des Pakets selbst hat übrigens noch keine Auswirkung darauf, ob mit automatischen Kolumnentiteln gearbeitet wird oder nicht. Erst die explizite Verwendung einer der Optionen `automark` oder `manualmark` oder einer der beiden Anweisungen `\automark` oder `\manualmark` schafft hier klare Verhältnisse.

Beispiele finden Sie in Abschnitt 5.5, ab Seite 284.

`\MakeMarkcase{`*`Text`*`}`
`markcase=`*`Wert`*

Die automatischen, nicht jedoch die manuellen Kolumnentitel verwenden `\MakeMarkcase` für ihre Ausgabe. Ist die Anweisung beim Laden von scrlayer nicht definiert, so gibt sie in der Voreinstellung ihr Argument *`Text`* unverändert aus. Die Voreinstellung kann jedoch durch Umdefinierung von `\MakeMarkcase` geändert werden. Dies kann auch automatisch durch Verwendung von Option `markcase` mit einem der Werte aus Tabelle 5.2, Seite 288 erfolgen.

Aufgrund der mangelnden typografischen Qualität der primitiven Umwandlung in Großbuchstaben (siehe die Erklärung zu `markcase` in Abschnitt 5.5 auf Seite 287) empfiehlt der KOMA-Script-Autor den Verzicht auf Versalsatz. Dies ist normalerweise mit `markcase=used`

möglich. Allerdings fügen einige Klassen selbst beispielsweise bei den Kolumnentitel für Verzeichnisse ein `\MakeUppercase` oder sogar die TEX-Anweisung `\uppercase` ein. Für diese Fälle gibt es auch noch die Einstellung `markcase=noupper`, mit deren Hilfe `\MakeUppercase` und `\uppercase` für die Kolumnentitel lokal deaktiviert werden können.

```
\righttopmark
\rightbotmark
\rightfirstmark
\rightmark
\lefttopmark
\leftbotmark
\leftfirstmark
\leftmark
```

v3.16

LATEX verwendet für die Seitenstile normalerweise eine zweiteilige TEX-Marke. Im Kolumnentitel kann auf den linken Teil der Marke mit `\leftmark` zugegriffen werden, während der rechte Teil der Marke über `\rightmark` verfügbar ist. Tatsächlich ist es wohl auch so gedacht, dass `\leftmark` für linke Seiten und `\rightmark` für rechte Seiten im doppelseitigen Druck verwendet wird. Im einseitigen Layout setzen die Gliederungsbefehle der Standardklassen den linken Teil der Marke hingegen gar nicht erst.

TEX selbst kennt drei Möglichkeiten, auf eine Marke zuzugreifen. `\botmark` ist die auf der zuletzt zusammengestellten Seite zuletzt gültige Marke. Das entspricht der letzten gesetzten Marke der Seite. Wurde auf der Seite keine Marke gesetzt, so entspricht es der zuletzt gesetzten Marke auf den bereits ausgegebenen Seiten. Die LATEX-Anweisung `\leftmark` verwendet genau diese Marke, gibt also den linken Teil der letzten Marke der Seite aus. Dies entspricht ebenfalls `\leftbotmark`. Im Vergleich dazu gibt `\rightbotmark` den rechten Teil dieser Marke aus.

`\firstmark` ist die erste Marke der zuletzt zusammengestellten Seite. Das entspricht der ersten Marke, die auf der Seite gesetzt wurde. Wurde auf der Seite keine Marke gesetzt, so entspricht es der zuletzt gesetzten Marke auf den bereits ausgegebenen Seiten. Die LATEX-Anweisung `\rightmark` verwendet genau diese Marke, gibt also den rechten Teil der ersten Marke der Seite aus. Dies entspricht ebenfalls `\rightfirstmark`. Im Vergleich dazu gibt `\leftfirstmark` den linken Teil dieser Marke aus.

`\topmark` ist der Inhalt, den `\botmark` hatte, bevor die aktuelle Seite zusammengestellt wurde. LATEX verwendet dies selbst nie. scrlayer bietet die Möglichkeit, mit `\lefttopmark` den linken Teil dieser Marke und mit `\righttopmark` den rechten Teil auszugeben.

Es ist zu beachten, dass der linke und rechte Teil der Marke immer nur gemeinsam gesetzt werden kann. Selbst wenn man mit `\markright` nur den rechten Teil verändert, wird der linke Teil (unverändert) mitgesetzt. Entsprechend setzen im doppelseitigen Layout die höheren Gliederungsebenen beim Seitenstil `headings` immer beide Teile. Beispielsweise verwendet `\chaptermark` dann `\markboth` mit einem leeren rechten Argument. Das ist auch der Grund, warum `\rightmark` beziehungsweise `\rightfirstmark` auf der Seite einer Kapitelüberschrift

immer einen leeren Wert zurückgibt, selbst wenn danach beispielsweise über \sectionmark oder indirekt über \section ein neuer rechter Teil gesetzt wurde.

Bitte beachten Sie, dass die Verwendung einer der hier erklärten Anweisungen zur Ausgabe des linken oder rechten Teils der Marke innerhalb einer Seite zu unerwarteten Ergebnissen führen kann. Sie sind wirklich nur zur Verwendung im Kopf oder Fuß eines Seitenstils gedacht. Daher sollten sie bei scrlayer immer Teil des Inhalts einer Ebene sein. Dagegen spielt es keine Rolle, ob sie auf den Hintergrund oder den Vordergrund beschränkt werden, da alle Ebenen erst nach der Zusammenstellung der aktuellen Seite ausgegeben werden.

Näheres zum Mark-Mechanismus von TEX ist beispielsweise [Knu90, Kapitel 23] zu entnehmen. Das Thema ist dort als absolutes Expertenwissen markiert.

```
\headmark
\pagemark
```

Mit \headmark kann man sich das Leben erleichtern. Diese Erweiterung von scrlayer entspricht je nachdem, ob die aktuelle Seite eine linke oder rechte ist, \leftmark oder \rightmark.

Die Anweisung \pagemark hat genau genommen nichts mit den Marken von TEX zu tun. Sie dient dazu, eine formatierte Seitenzahl zu platzieren. Bei ihrer Ausgabe wird dann auch die Schrifteinstellung für das Element pagenumber verwendet. Diese kann mit Hilfe der Anweisungen \setkomafont und \addtokomafont verändert werden (siehe auch Abschnitt 3.6, Seite 60).

Für ein Beispiel siehe Abschnitt 5.5, Seite 289.

```
\partmarkformat
\chaptermarkformat
\sectionmarkformat
\subsectionmarkformat
\subsubsectionmarkformat
\paragraphmarkformat
\subparagraphmarkformat
```

Diese Anweisungen werden von den KOMA-Script-Klassen und auch von scrlayer intern üblicherweise verwendet, um die Gliederungsnummern der automatischen Kolumnentitel zu formatieren. Dabei wird auch der \autodot-Mechanismus der KOMA-Script-Klassen unterstützt. Bei Bedarf können diese Anweisungen umdefiniert werden, um eine andere Formatierung der Nummern zu erreichen. Siehe dazu gegebenenfalls das Beispiel in Abschnitt 5.5, auf Seite 290.

```
\partmark{Text}
\chaptermark{Text}
\sectionmark{Text}
\subsectionmark{Text}
\subsubsectionmark{Text}
\paragraphmark{Text}
\subparagraphmark{Text}
```

Diese Anweisungen werden intern von den meisten Klassen verwendet, um die Marken entsprechend der Gliederungsbefehle zu setzen. Dabei wird als Argument lediglich der Text, nicht jedoch die Nummer erwartet. Die Nummer wird stattdessen automatisch über den aktuellen Zählerstand ermittelt, falls mit nummerierten Überschriften gearbeitet wird.

Allerdings verwenden nicht alle Klassen in allen Gliederungsebenen eine solche Anweisung. So wird beispielsweise `\partmark` von den Standardklassen nie aufgerufen, während die KOMA-Script-Klassen selbstverständlich auch `\partmark` unterstützen.

Falls diese Anweisungen vom Anwender umdefiniert werden, sollte er unbedingt darauf achten, vor dem Setzen der Nummer ebenfalls über `secnumdepth` zu prüfen, ob die Nummern auszugeben sind. Dies gilt auch, wenn der Anwender `secnumdepth` selbst nicht verändert, weil Pakete und Klassen sich eventuell auf die Wirkung von `secnumdepth` verlassen!

Das Paket scrlayer definiert diese Anweisungen außerdem bei jedem Aufruf von `\automark` oder `\manualmark` oder den entsprechenden Optionen teilweise neu, um so die gewünschten automatischen oder manuellen Kolumnentitel zu erreichen.

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

Unabhängig davon, ob gerade mit manuellen oder automatischen Kolumnentiteln gearbeitet wird, kann man jederzeit die *linke Marke* oder *rechte Marke* mit einer dieser Anweisungen setzen. Dabei ist zu beachten, dass die resultierende linke Marke in `\leftmark` die letzte auf der entsprechenden Seite gesetzte Marke ist, während die resultierende rechte Marke in `\rightmark` die erste auf der entsprechenden Seite gesetzte Marke ausgibt. Näheres dazu ist den weiterführenden Erklärungen zu `\rightmark` in Abschnitt 18.1, Seite 481 oder zu `\rightfirstmark`, Seite 446 zu entnehmen.

Wird mit manuellen Kolumnentiteln gearbeitet, so bleiben die Marken gültig, bis sie durch erneute Verwendung der entsprechenden Anweisung explizit ersetzt werden. Bei automatischen Kolumnentiteln können Marken hingegen je nach Konfigurierung des Automatismus ihre Gültigkeit mit einer der nächsten Gliederungsüberschriften verlieren.

Auch im Zusammenhang mit den Sternvarianten der Gliederungsbefehle können diese Anweisungen nützlich sein. Ausführliche Beispiele für die Verwendung von `\markboth` mit dem von scrlayer abgeleiteten Paket scrlayer-scrpage sind in Abschnitt 5.5, ab Seite 291 zu finden.Die

Anweisung \markdouble setzt sowohl die linke als auch rechte Marke auf denselben Inhalt. Damit ist \markdouble{*Marke*} eine abkürzende Schreibweise für \markboth{*Marke*}{*Marke*} mit zwei identischen Argumenten.

\GenericMarkFormat{*Gliederungsname*}

Diese Anweisung wird in der Voreinstellung zur Formatierung aller Gliederungsnummern in automatischen Kolumnentiteln unterhalb der Unterabschnitte und bei Klassen ohne \chapter zusätzlich auch für die Ebene der Abschnitte und Unterabschnitte verwendet, soweit die entsprechenden Mark-Anweisungen nicht bereits anderweitig definiert sind. Dabei verwendet die Anweisung in der Voreinstellung \@seccntmarkformat, wenn eine solche interne Anweisung wie bei den KOMA-Script-Klassen definiert ist. Anderenfalls wird mit \@seccntformat eine Anweisung verwendet, die bereits vom LaTeX-Kern für Klassen und Pakete bereitgestellt und von KOMA-Script etwas modifiziert wird. Als Argument erwartet \GenericMarkFormat den Namen der Gliederung, also beispielsweise chapter oder section *ohne* vorangestellten umgekehrten Schrägstrich (engl. *backslash*).

Durch Umdefinierung dieser Anweisung kann damit die Standardformatierung aller Gliederungsnummern im Kolumnentitel geändert werden, die darauf zurückgreifen. Ebenso kann eine Klasse darüber eine andere Standardformatierung vorgeben, ohne alle Befehle einzeln ändern zu müssen.

Ein ausführliches Beispiel für das Zusammenspiel der Anweisung \GenericMarkFormat mit den auf Seite 448 erklärten Anweisungen \sectionmarkformat und \subsectionmarkformat beziehungsweise \chaptermarkformat bei Verwendung des von scrlayer abgeleiteten Pakets scrlayer-scrpage ist in Abschnitt 15.1, ab Seite 452 zu finden.

\@mkleft{*linke Marke*}
\@mkright{*rechte Marke*}
\@mkdouble{*Marke*}
\@mkboth{*linke Marke*}{*rechte Marke*}

Innerhalb der Klassen und Pakete kommt es vor, dass Kolumnentitel nur dann gesetzt werden sollen, wenn automatische Kolumnentitel (siehe Option automark und Anweisung \automark auf Seite 444) aktiviert sind. Bei den Standardklassen geht dies ausschließlich über \@mkboth. Diese Anweisung entspricht entweder \@gobbletwo, einer Anweisung, die ihre beiden Argumente vernichtet, oder \markboth, einer Anweisung, mit der sowohl eine *linke Marke* als auch eine *rechte Marke* gesetzt wird. Pakete wie babel hängen sich ebenfalls in \@mkboth ein, um beispielsweise noch eine Sprachumschaltung im Kolumnentitel vorzunehmen.

Will man nun jedoch nur eine *linke Marke* oder nur eine *rechte Marke* setzen, ohne die jeweils andere Marke zu verändern, so fehlen entsprechende Anweisungen. Das Paket scrlayer selbst benötigt entsprechende Anweisungen beispielsweise im Rahmen der automatischen Kolumnentitel. Sind \@mkleft zum Setzen nur der *linken Marke*, \@mkright zum Setzen nur der *rechten Marke* oder \@mkdouble zum Setzen sowohl der rechten als auch der linken *Marke*

mit demselben Inhalt beim Laden von scrlayer nicht definiert, so werden sie vom Paket selbst definiert. Dabei wird eine Definition gewählt, die am Zustand von `\@mkboth` erkennt, ob mit automatischen Kolumnentiteln gearbeitet wird. Nur in diesem Fall setzen die Befehle auch eine entsprechende Marke.

Klassen- und Paketautoren können ebenfalls auf die passende der vier Anweisungen zurückgreifen, wenn sie linke oder rechte Marken setzen und dies auf den Fall beschränken wollen, dass mit automatischen Kolumnentiteln gearbeitet wird.

Zu weiteren Möglichkeiten zur Beeinflussung der Inhalte von Seitenstilen siehe auch Abschnitt 5.5, Seite 284.

## 14.7. Definition und Verwaltung von Schnittstellen für Endanwender

v3.28

Bis Version 3.27 stellte das Paket scrlayer noch einen als experimentell dokumentierten Mechanismus zur Verwaltung konkurrierender Benutzerschnittstellen zur Verfügung. Da der Nutzen dieses lediglich von KOMA-Script selbst eingesetzten Mechanismus sehr begrenzt war, ist er inzwischen als überholt gekennzeichnet. Die Befehle `\scrlayerInitInterface`, `\scrlayerOnAutoRemoveInterface`, `\scrlayerAddToInterface` und `\scrlayerAddCsToInterface` und die Optionen `forceoverwrite` und `autoremoveinterfaces` sollten daher nicht mehr verwendet werden.

v3.12

# Zusätzliche Möglichkeiten des Pakets scrlayer-scrpage

Über die Erklärungen in Kapitel 5 von Teil I dieser Anleitung hinaus bietet das Paket scrlayer-scrpage viele weitere Möglichkeiten. Diese stellen jedoch Erweiterungen dar, die der durchschnittliche Anwender nicht zwingend benötigt oder die nur aus Gründen der Kompatibilität zu scrpage2 existieren. Ihre Dokumentation hier in Teil II dient der Vertiefung und der Erweiterung des Wissens. Ihre Beherrschung geht über grundlegende Fähigkeiten hinaus.

## 15.1. Beeinflussung von Seitenstilen

Dieser Abschnitt ist als Ergänzung zu Abschnitt 14.6 zu verstehen und beschreibt Dinge, die sich dem Anfänger nicht unbedingt sofort erschließen. Wenn Sie jenen Abschnitt bereits gelesen und verstanden haben, können Sie natürlich auch direkt mit Abschnitt 15.2 auf Seite 454 forfahren.

```
\righttopmark
\rightbotmark
\rightfirstmark
\rightmark
\lefttopmark
\leftbotmark
\leftfirstmark
\leftmark
```

v3.16

LaTeX verwendet für die Seitenstile normalerweise eine zweiteilige TeX-Marke. Im Kolumnentitel kann auf den linken Teil der Marke mit `\leftmark` zugegriffen werden, während der rechte Teil der Marke über `\rightmark` verfügbar ist. Tatsächlich ist es wohl auch so gedacht, dass `\leftmark` für linke Seiten und `\rightmark` für rechte Seiten im doppelseitigen Druck verwendet wird, während im einseitigen Layout nur rechte Marken gesetzt werden

TeX selbst kennt drei Möglichkeiten, auf eine Marke zuzugreifen. `\botmark` ist die auf der zuletzt zusammengestellten Seite zuletzt gültige Marke. Das entspricht der letzten gesetzten Marke der Seite. Wurde auf der Seite keine Marke gesetzt, so entspricht es der zuletzt gesetzten Marke auf den bereits ausgegebenen Seiten. Die LaTeX-Anweisung `\leftmark` verwendet genau diese Marke, gibt also den linken Teil der letzten Marke der Seite aus. Dies entspricht ebenfalls `\leftbotmark`. Im Vergleich dazu gibt `\rightbotmark` den rechten Teil dieser Marke aus.

`\firstmark` ist die erste Marke der zuletzt zusammengestellten Seite. Das entspricht der ersten Marke, die auf der Seite gesetzt wurde. Wurde auf der Seite keine Marke gesetzt, so entspricht es der zuletzt gesetzten Marke auf den bereits ausgegebenen Seiten. Die LaTeX-Anweisung `\rightmark` verwendet genau diese Marke, gibt also den rechten Teil der ersten Marke der Seite aus. Dies entspricht ebenfalls `\rightfirstmark`. Im Vergleich dazu gibt

\leftfirstmark den linken Teil dieser Marke aus.

\topmark ist der Inhalt, den \botmark hatte, bevor die aktuelle Seite zusammengestellt wurde. LaTeX verwendet dies selbst nie. scrlayer bietet die Möglichkeit, mit \lefttopmark den linken Teil dieser Marke und mit \righttopmark den rechten Teil auszugeben.

Es ist zu beachten, dass der linke und rechte Teil der Marke immer nur gemeinsam gesetzt werden kann. Selbst wenn man mit \markright nur den rechten Teil verändert, wird der linke Teil (unverändert) mitgesetzt. Entsprechend setzen im doppelseitigen Layout die höheren Gliederungsebenen beim Seitenstil headings immer beide Teile. Beispielsweise verwendet \chaptermark dann \markboth mit einem leeren rechten Argument. Das ist auch der Grund, warum \rightmark beziehungsweise \rightfirstmark auf der Seite einer Kapitelüberschrift immer einen leeren Wert zurückgibt, selbst wenn danach beispielsweise über \sectionmark oder indirekt über \section ein neuer rechter Teil gesetzt wurde.

Bitte beachten Sie, dass die Verwendung einer der hier erklärten Anweisungen zur Ausgabe des linken oder rechten Teils der Marke innerhalb einer Seite zu unerwarteten Ergebnissen führen kann. Sie sind wirklich nur zur Verwendung im Kopf oder Fuß eines Seitenstils gedacht. Daher sollten sie bei scrlayer immer Teil des Inhalts einer Ebene sein. Dagegen spielt es keine Rolle, ob sie auf den Hintergrund oder den Vordergrund beschränkt werden, da alle Ebenen erst nach der Zusammenstellung der aktuellen Seite ausgegeben werden.

Näheres zum Mark-Mechanismus von TeX ist beispielsweise [Knu90, Kapitel 23] zu entnehmen. Das Thema ist dort als absolutes Expertenwissen markiert.

\GenericMarkFormat{*Gliederungsname*}

Diese Anweisung wird in der Voreinstellung zur Formatierung aller Gliederungsnummern in automatischen Kolumnentiteln unterhalb der Unterabschnitte und bei Klassen ohne \chapter zusätzlich auch für die Ebene der Abschnitte und Unterabschnitte verwendet, soweit die entsprechenden Mark-Anweisungen nicht bereits anderweitig definiert sind. Dabei verwendet die Anweisung in der Voreinstellung \@seccntmarkformat, wenn eine solche interne Anweisung wie bei den KOMA-Script-Klassen definiert ist. Anderenfalls wird mit \@seccntformat eine Anweisung verwendet, die bereits vom LaTeX-Kern für Klassen und Pakete bereitgestellt und von KOMA-Script etwas modifiziert wird. Als Argument erwartet \GenericMarkFormat den Namen der Gliederung, also beispielsweise chapter oder section *ohne* vorangestellten umgekehrten Schrägstrich (engl. *backslash*).

Durch Umdefinierung dieser Anweisung kann damit die Standardformatierung aller Gliederungsnummern im Kolumnentitel geändert werden, die darauf zurückgreifen. Ebenso kann eine Klasse darüber eine andere Standardformatierung vorgeben, ohne alle Befehle einzeln ändern zu müssen.

**Beispiel:** Angenommen, Sie wollen, dass bei allen Gliederungsnummern im Kolumnentitel eines Artikels die Nummer als weiße Schrift auf einem schwarzen Kasten ausgegeben wird. Da bei Artikeln mit Klasse article die Anweisungen \sectionmarkformat und

\subsectionmarkformat von scrlayer mit Hilfe von \GenericMarkFormat definiert werden, genügt dafür die entsprechende Umdefinierung dieser einen Anweisung:

```
\documentclass{article}
\usepackage{blindtext}
\usepackage[automark]{scrlayer-scrpage}
\usepackage{xcolor}
\newcommand*{\numberbox}[1]{%
  \colorbox{black}{\strut~\textcolor{white}{#1}~}%
}
\renewcommand*{\GenericMarkFormat}[1]{%
  \protect\numberbox{\csname the#1\endcsname}%
  \enskip
}
\begin{document}
\blinddocument
\end{document}
```

Für die Farbumschaltungen werden Anweisungen des Pakets xcolor verwendet. Näheres dazu ist der Anleitung zum Paket zu entnehmen (siehe [Ker07]). Außerdem wird eine unsichtbare Stütze mit \strut eingefügt.

Für den Kasten mit der Nummer wird eine eigene Hilfsanweisung \numberbox definiert. Diese wird in der Umdefinierung von \GenericMarkFormat mit \protect vor der Expansion geschützt. Dies ist notwendig, weil sonst durch das \MakeUppercase für den Versalsatz der Kolumnentitel nicht mehr die Farben »black« und »white«, sondern die Farben »BLACK« und »WHITE« verlangt würden, die gar nicht definiert sind. Alternativ könnte man \numberbox auch mit Hilfe von \DeclareRobustCommand* statt mit \newcommand* definieren (siehe [Tea06]).

Wollte man dasselbe mit einer KOMA-Script-Klasse oder mit den Standardklassen book oder report erreichen, so müsste man übrigens zusätzlich \sectionmarkformat und – je nach Klasse – \subsectionmarkformat beziehungsweise \chaptermarkformat umdefinieren, da diese bei Verwendung der genannten Klassen \GenericMarkFormat nicht verwenden:

```
\documentclass[headheight=19.6pt]{scrbook}
\usepackage{blindtext}
\usepackage[automark]{scrlayer-scrpage}
\usepackage{xcolor}
\newcommand*{\numberbox}[1]{%
  \colorbox{black}{\strut~\textcolor{white}{#1}~}%
}
\renewcommand*{\GenericMarkFormat}[1]{%
  \protect\numberbox{\csname the#1\endcsname}%
  \enskip
```

```
}
\renewcommand*{\chaptermarkformat}{%
  \GenericMarkFormat{chapter}%
}
\renewcommand*{\sectionmarkformat}{%
  \GenericMarkFormat{section}%
}
\begin{document}
\blinddocument
\end{document}
```

Über Option headheight wird dabei auch die Warnung beseitigt, die im vorherigen Beispiel noch erzeugt wurde.

```
\@mkleft{linke Marke}
\@mkright{rechte Marke}
\@mkdouble{Marke}
\@mkboth{linke Marke}{rechte Marke}
```

Innerhalb der Klassen und Pakete kommt es vor, dass Kolumnentitel nur dann gesetzt werden sollen, wenn automatische Kolumnentitel (siehe Option automark und Anweisung \automark auf Seite 444) aktiviert sind. Bei den Standardklassen geht dies ausschließlich über \@mkboth. Diese Anweisung entspricht entweder \@gobbletwo, einer Anweisung, die ihre beiden Argumente vernichtet, oder \markboth, einer Anweisung, mit der sowohl eine *linke Marke* als auch eine *rechte Marke* gesetzt wird. Pakete wie babel hängen sich ebenfalls in \@mkboth ein, um beispielsweise noch eine Sprachumschaltung im Kolumnentitel vorzunehmen.

Will man nun jedoch nur eine *linke Marke* oder nur eine *rechte Marke* setzen, ohne die jeweils andere Marke zu verändern, so fehlen entsprechende Anweisungen. Das Paket scrlayer selbst benötigt entsprechende Anweisungen beispielsweise im Rahmen der automatischen Kolumnentitel. Sind \@mkleft zum Setzen nur der *linken Marke*, \@mkright zum Setzen nur der *rechten Marke* oder \@mkdouble zum Setzen sowohl der rechten als auch der linken *Marke* mit demselben Inhalt beim Laden von scrlayer nicht definiert, so werden sie vom Paket selbst definiert. Dabei wird eine Definition gewählt, die am Zustand von \@mkboth erkennt, ob mit automatischen Kolumnentiteln gearbeitet wird. Nur in diesem Fall setzen die Befehle auch eine entsprechende Marke.

Klassen- und Paketautoren können ebenfalls auf die passende der vier Anweisungen zurückgreifen, wenn sie linke oder rechte Marken setzen und dies auf den Fall beschränken wollen, dass mit automatischen Kolumnentiteln gearbeitet wird.

## 15.2. Definition eigener Seitenstil-Paare

In Abschnitt 5.4 wurden die beiden Seitenstile scrheadings und plain.scrheadings vorgestellt. Diese bilden quasi ein Paar, bei dem scrheadings als Haupt-Seitenstil mit Kolumnen-

titel vorgesehen ist, während `plain.scrheadings` ein dazu passender `plain`-Seitenstil ohne Kolumnentitel, aber üblicherweise mit Paginierung, also mit Seitenzahl ist. Neben der Konfiguration dieses vordefinierten Paares bietet scrlayer-scrpage auch die Möglichkeit, zusätzliche Paare zu definieren. Der Name des Haupt-Seitenstils, beispielsweise `scrheadings`, dient dabei quasi auch als Name des Seitenstil-Paares.

Die allermeisten Anwender werden in der Regel mit dem einen vordefinierten Seitenstil-Paar `scrheadings` auskommen. Die in diesem Abschnitt dokumentierten Anweisungen sind daher eher Ergänzungen für besondere Fälle. Da mir während des Abfassens dieser Anleitung keine handlichen Anwendungsbeispiele eingefallen sind, gibt es auch keine ausführlichen Beispiele. Sollte mir im Support einmal eine besonders schöne Anwendung begegnen, werde ich solche aber in zukünftigen Fassungen gerne aufgreifen. Ich bin jedoch gleichzeitig sicher, dass sich all diese Fälle auch mit dem einen Paar `scrheadings` lösen lassen.

```
\defpairofpagestyles[Eltern-Paar]{Name}{Definition}
\newpairofpagestyles[Eltern-Paar]{Name}{Definition}
\renewpairofpagestyles[Eltern-Paar]{Name}{Definition}
\providepairofpagestyles[Eltern-Paar]{Name}{Definition}
```

Mit diesen Anweisungen können Paare von Seitenstilen vergleichbar zu `scrheadings` und `plain.scrheadings` definiert werden. Dabei ist *Name* der Name des zu `scrheadings` vergleichbaren Hauptseitenstils, der für die Verwendung mit Kolumnentiteln ausgelegt ist. Für den Namen des zugehörigen `plain`-Seitenstil wird *Name* automatisch `plain.` vorangestellt. *Name* ist also gleichzeitig der Name des Paares und des Hauptseitenstils dieses Paares, während `plain.`*Name* der Name des `plain`-Seitenstils dieses Paares ist.

Ist das optionale Argument *Eltern-Paar* angegeben, so ist dies der Name eines Seitenstil-Paares, mit dessen Einstellungen das neue Paar initialisiert werden soll. Das neue Paar erbt also quasi die Konfiguration des Eltern-Paares.

Während in Abschnitt 5.4 der Eindruck entstanden sein mag, dass sich die dort erläuterten Anweisungen nur auf `scrheadings` und `plain.scrheadings` beziehen, gilt das tatsächlich nur, solange diese beiden Seitenstile das einzige Seitenstil-Paar darstellen. Sobald es mehrere Seitenstil-Paare gibt, beziehen sich `\lehead`, `\cehead`, `\rehead`, `\lohead`, `\cohead`, `\rohead`, `\lefoot`, `\cefoot`, `\refoot`, `\lofoot`, `\cofoot`, `\rofoot`, `\ihead`, `\chead`, `\ohead`, `\ifoot`, `\cfoot` und `\ofoot` auf das zuletzt aktive Paar.

Neben den achtzehn vorgenannten sind auch die drei nachfolgend dokumentierten Anweisungen `\clearmainofpairofpagestyles`, `\clearplainofpairofpagestyles` und `\clearpairofpagestyles` für die Verwendung im letzten Parameter, *Definition*, gedacht. In diesem Fall stellen sie eine Art Grundkonfiguration des Seitenstil-Paares dar, die immer dann ausgeführt wird, wenn das Seitenstil-Paar aktiviert wird. Ein Seitenstil-Paar wird aktiviert, indem einer der beiden Seitenstile des Paares aktiviert wird. Dies geschieht in der Regel mit Hilfe von `\pagestyle`.

Es sei darauf hingewiesen, dass die Anweisungen aus Abschnitt 5.5 ab Seite 284 ohnehin allgemeiner Natur sind und für alle mit `scrlayer-scrpage` definierten Seitenstile gelten.

Während `\defpairofpagestyles` das Seitenstil-Paar unabhängig davon, ob entsprechende Seitenstile bereits existieren, definiert, tun `\newpairofpagestyles` und `\providepairofpagestyles` dies nur, wenn die Seitenstile noch nicht definiert sind. Ist mindestens einer der beiden Seitenstile des Paares bereits definiert, so wird die neuerliche Definition bei `\providepairofpagestyles` ignoriert, wohingegen sie bei `\newpairofpagestyles` zu einem Fehler führt. Für die Umdefinierung bereits existierender Paare kann `\renewpairofpagestyles` verwendet werden. Hier wird ein Fehler gemeldet, wenn einer der beiden Seitenstile oder beide Seitenstile des Paares noch nicht existieren.

An dieser Stelle sei auch noch darauf hingewiesen, dass jedes Paar aus einer Reihe von mit scrlayer definierten Ebenen besteht. Interessierte Anwender werden die Namen der Ebenen im nachfolgenden Abschnitt in Tabelle 15.1 auf Seite 458 kennenlernen. Die dort angebenen Ebenen werden für ein Seitenstilpaar durch die Tatsache verdoppelt, dass sie sowohl für *`Name`* als auch für `plain.`*`Name`* definiert werden.

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

Mit `\clearmainofpairofpagestyles` wird der Hauptseitenstil des zuletzt aktivierten Seitenstil-Paares leer konfiguriert. Dagegen wird mit der Anweisung `\clearplainofpairofpagestyles` der `plain`-Seitenstil des entsprechenden Seitenstil-Paares leer konfiguriert. Die Anweisung `\clearpairofpagestyles` konfiguriert schließlich beide Seitenstile des entsprechenden Paares als leer.

Es ist jedoch zu beachten, dass keine dieser Anweisungen die Definitionen aus dem Parameter *`Definition`*, der bei der Definition des Seitenstil-Paares angegeben wurde (siehe oben), entfernt. Bei der erneuten Auswahl eines Seitenstils des Paares werden jene Einstellungen also erneut ausgeführt!

Die Anweisungen können ebenfalls innerhalb von *`Definition`* bei der zuvor erklärten Definition eines Seitenstil-Paares verwendet werden. Sie können aber auch jederzeit außerhalb der Definition eines Seitenstil-Paares verwendet werden. In diesem Fall beziehen sie sich auf das zuletzt aktivierte Paar.

## 15.3. Definition komplexer Seitenstile

Neben den vordefinierten Seitenstilen bietet scrlayer-scrpage auch noch eine eher grundlegende Schnittstelle zur Definition von Seitenstilen. Die bisher erläuterten Konzepte greifen bei der Implementierung ebenso wie die von Abschnitt 15.4 auf diese Möglichkeit zurück. Aufgrund ihrer hohen Komplexität wird sie jedoch nur erfahrenen Anwendern empfohlen. Weniger er-

fahrene Anwender können mit den vorgenannten Möglichkeiten bereits nahezu alles erreichen, was auch mit dieser grundlegenden Schnittstelle möglich ist.

Ausdrücklich wird davon abgeraten, die hier vorgestellten Anweisungen dazu zu verwenden, um einzelne Teile bereits definierte Seitenstil-Paare umzudefinieren. Unter anderem würde dabei die Verknüpfung zu einem Paar beschädigt. Für die Umdefinierung von Seitenstil-Paaren sei auf Abschnitt 15.2 ab Seite 454 verwiesen.

```
\defpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\newpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\providepagestyle{Name}{Kopfdefinition}{Fußdefinition}
\renewpagestyle{Name}{Kopfdefinition}{Fußdefinition}
```

Diese Anweisungen dienen der Definition eines einzelnen Seitenstils mit maximaler Flexibilität. Dabei ist *Name* der Name des Seitenstils, der definiert werden soll.

Die beiden Parameter *Kopfdefinition* und *Fußdefinition* haben den identischen Aufbau:

```
(Länge der oberen Linie,Dicke der oberen Linie)%
{Definition für linke Seiten im doppelseitigen Layout}%
{Definition für rechte Seiten im doppelseitigen Layout}%
{Definition für Seiten im einseitigen Layout}%
(Länge der unteren Linie,Dicke der unteren Linie)
```

Dabei sind die Argumente in den runden Klammern optional, das heißt, sie können zusammen mit den Klammern weggelassen werden. In diesem Fall richten sich die Längen und Dicken je nach Linie nach den Angaben der Optionen `headtopline`, `headsepline`, `footsepline` und `footbotline` (siehe Abschnitt 5.5, Seite 293).

Die drei Argumente mit den Definitionen sind obligatorisch und werden je nach Seite und Layouteinstellung verwendet. Der Inhalt der Definitionen ist frei wählbar. Für Seitenstile mit Kolumnentitel wird jedoch die Verwendung von `\headmark`, `\leftmark` oder `\rightmark` innerhalb der Definitionen empfohlen. Keinesfalls sollte man hier direkt eine Gliederungsnummer oder einen Überschriftentext als Kolumnentitel angeben. Aufgrund des asynchronen Seitenaufbaus von LaTeX können sonst die falsche Nummer oder der falsche Text im Seitenkopf oder Seitenfuß erscheinen.

Bei `\defpagestyle` wird der Seitenstil unabhängig davon, ob er bereits existiert oder nicht, neu definiert. Demgegenüber meldet `\newpagestyle` einen Fehler, wenn bereits ein Seitenstil gleichen Namens existiert. Im Unterschied dazu wird die Definition bei `\providepagestyle` ignoriert, falls der *Name* bereits für einen Seitenstil verwendet wurde. Umgekehrt kann mit `\renewpagestyle` nur ein bereits vorhandener Seitenstil umdefiniert werden. Für einen neuen Namen meldet diese Anweisung dagegen einen Fehler.

Alle vier Anweisungen basieren auf `\DeclarePageStyleByLayers` des Pakets scrlayer. Die dabei für einen Seitenstil *Name* definierten Ebenen sind in Tabelle 15.1 aufgeführt. Näheres zu Ebenen und Ebenen-Seitenstilen ist Kapitel 14 ab Seite 418 zu entnehmen.

Tabelle 15.1.: Die von scrlayer-scrpage zu einem Seitenstil *Name* definierten Ebenen

| Name der Ebene | Bedeutung der Ebene |
|---|---|
| *Name*`.head.above.line` | die Linie über dem Kopf |
| *Name*`.head.odd` | der Kopf von rechten Seiten im doppelseitigen Layout |
| *Name*`.head.even` | der Kopf von linken Seiten im doppelseitigen Layout |
| *Name*`.head.oneside` | der Kopf von Seiten im einseitigen Layout |
| *Name*`.head.below.line` | die Linie unter dem Kopf |
| *Name*`.foot.above.line` | die Linie über dem Fuß |
| *Name*`.foot.odd` | der Fuß von rechten Seiten im doppelseitigen Layout |
| *Name*`.foot.even` | der Fuß von linken Seiten im doppelseitigen Layout |
| *Name*`.foot.oneside` | der Fuß von Seiten im einseitigen Layout |
| *Name*`.foot.below.line` | die Linie unter dem Fuß |

**Beispiel:** Angenommen, Sie wollen den gesamten Kopf des Seitenstil `scrheadings` mit einer Farbe hinterlegen. Aufgrund der Einleitung zu diesem Kapitel und Tabelle 15.1, wissen Sie, dass `scrheadings` ein Ebenen-Seitenstil ist, der unter anderem aus den Ebenen `scrheadings.head.oneside`, `scrheadings.head.odd` und `scrheadings.head.even` besteht. Sie definieren nun drei weitere Ebenen für deren Hintergrund und fügen diese am Anfang des Seitenstils ein:

```
\documentclass{scrartcl}
 \usepackage[automark]{scrlayer-scrpage}
 \usepackage{xcolor}
 \usepackage{blindtext}
 \DeclareLayer[clone=scrheadings.head.oneside,
   contents={%
     \color{yellow}%
     \rule[-\dp\strutbox]%
          {\layerwidth}{\layerheight}%
   }%
 ]{scrheadings.head.oneside.background}
 \DeclareLayer[clone=scrheadings.head.odd,
   contents={%
     \color{yellow}%
     \rule[-\dp\strutbox]%
          {\layerwidth}{\layerheight}%
   }%
 ]{scrheadings.head.odd.background}
 \DeclareLayer[clone=scrheadings.head.even,
   contents={%
     \color{yellow}%
```

```
    \rule[-\dp\strutbox]%
         {\layerwidth}{\layerheight}%
  }%
]{scrheadings.head.even.background}
\AddLayersAtBeginOfPageStyle{scrheadings}{%
  scrheadings.head.oneside.background,%
  scrheadings.head.odd.background,%
  scrheadings.head.even.background%
}
\pagestyle{scrheadings}
\begin{document}
\blinddocument
\end{document}
```

Wie Sie sehen, wurden in dem Beispiel drei Ebenen verwendet, damit Position und Größe der Hintergrund-Ebenen per Option `clone` einfach jeweils von der Ebene für den Kopf kopiert werden konnten. Das ist einfacher, als nur eine Hintergrundebene zu verwenden und für diese die Position umständlich dynamisch zu bestimmen.

Der farbige Hintergrund selbst wurde in diesem Beispiel mit einer `\rule`-Anweisung gesetzt. Für die Größe wurden dabei mit `\layerwidth` und `\layerheight` die aktuelle Breite und Höhe der Ebene verwendet. Per optionalem Argument von `\rule` wurde der Hintergrund um die Höhe der Unterlängen, `\dp\strutbox`, nach unten verschoben.

Statt wie im Beispiel für die Hintergrundfarbe neue Ebenen hinzuzufügen, hätte man das Problem übrigens auch mit `\colorbox` und `\chead` lösen können. Es wird empfohlen, eine entsprechende Lösung als Übung zu erarbeiten. Ebenso hätte man die Hintergrundebenen auch einzeln und jeweils unmittelbar vor der entsprechenden Inhaltsebene einfügen können. Eine entsprechende Umsetzung bietet sich als weitere Übung an.

## 15.4. Definition einfacher Seitenstile mit dreigeteiltem Kopf und Fuß

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

## 15.5. Das obsolete Erbe von scrpage2

Das Paket scrlayer-scrpage enthält auch noch einige Altlasten, die von scrpage2 stammen und nur existieren, um möglichst kompatibel mit jenem Paket zu sein. Kenntnisse dazu benötigen Anwender nur, wenn sie ein altes Dokument, das noch auf scrpage2 basiert, bearbeiten wollen. Für neue Dokumente sollten die hier dokumentieren Dinge dagegen nicht verwendet werden!

`hmode=`*`Ein-Aus-Wert`*

Bei scrpage2 wurden die Köpfe und Füße der Seitenstile noch grundsätzlich im horizontalen Modus ausgegeben. Bei scrlayer-scrpage wird hingegen in der Voreinstellung erst durch entsprechende Ausgaben selbst in den horizontalen Modus geschaltet. Aktiviert man jedoch Option `hmode` verhält sich scrlayer-scrpage auch in dieser Hinsicht kompatibel mit scrpage2 und schaltet bereits vor der Ausgabe in den horizontalen Modus. Dies kann Auswirkungen sowohl auf die Verarbeitung von Leerzeichen am Anfang der Ausgabe als auch auf die vertikale Ausrichtung haben.

Die Option versteht die Standardwerte für einfache Schalter, die in Tabelle 2.5 auf Seite 41 angegeben sind. In der Voreinstellung ist der Schalter deaktiviert.

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

# Notizspalten mit scrlayer-notecolumn

Bis einschließlich Version 3.11b unterstützte KOMA-Script Notizspalten nur in Form der Marginalienspalte, die mit `\marginpar` und `\marginline` (siehe Abschnitt 3.21, Seite 155) mit Inhalt versehen werden können. Jene Art der Randnotizen hat allerdings einige Nachteile:

- Randnotizen können nur vollständig auf einer einzelnen Seite gesetzt werden. Seitenumbrüche innerhalb von Randnotizen sind nicht möglich. Dies führt teilweise dazu, dass die Randnotizen bis in den unteren Rand hineinragen.
- Randnotizen in der Nähe des Seitenumbruchs können auf die nächste Seite rutschen und dort im Falle des doppelseitigen Layouts mit alternierenden Marginalienspalten im falschen Rand ausgegeben werden. Dieses Problem ist mit dem Zusatzpaket mparhack oder durch Verwendung von `\marginnote` aus dem Paket marginnote (siehe [Koh12]) lösbar.
- Randnotizen innerhalb von Gleitumgebungen oder Fußnoten sind nicht möglich. Auch dieses Problem ist mit marginnote lösbar.
- Es gibt nur eine Marginalienspalte oder allenfalls zwei, wenn mit `\reversemarginpar` und `\normalmarginpar` gearbeitet wird, wobei `\reversemarginpar` bei doppelseitigen Dokumenten kaum zu gebrauchen ist.

Die Verwendung von marginnote führt zu einem weiteren Problem. Da das Paket keine Kollisionserkennung besitzt, können sich Randnotizen, die in unmittelbarer Nähe veranlasst wurden, gegenseitig ganz oder teilweise überdecken. Darüber hinaus führt `\marginnote`, je nach den gewählten Einstellungen von marginnote, manchmal zu Veränderungen beim Zeilenabstand des normalen Textes.

Das Paket scrlayer-notecolumn tritt an, all diese Probleme zu lösen. Dazu stützt es sich auf die Grundfunktionalität von scrlayer. Damit geht aber auch ein Nachteil einher: Notizen können nur auf den Seiten ausgegeben werden, die einen auf scrlayer basierenden Seitenstil besitzen. Dieser Nachteil lässt sich mit Hilfe von scrlayer-scrpage jedoch leicht auflösen oder sogar in einen Vorteil verwandeln.

## 16.1. Hinweise zum Entwicklungsstand

Das Paket wurde ausschließlich zur Demonstration des Potentials von scrlayer als sogenannter *Proof of Concept* entwickelt. Obwohl es sich derzeit noch in einem recht frühen Entwicklungsstadium befindet, ist die Stabilität von weiten Teilen weniger eine Frage von scrlayer-notecolumn als von scrlayer. Dennoch ist davon auszugehen, dass sich auch in scrlayer-notecolumn noch Fehler befinden. Es wird darum gebeten, diese bei Auffinden zu melden. Einige *Unzulänglichkeiten*

des Pakets sind jedoch auch der Minimierung des Aufwands geschuldet. So können Notizspalten zwar über Seiten hinweg umbrochen werden, allerdings findet dabei kein neuerlicher Absatzumbruch statt. Dies ist bei TEX schlicht nicht vorgesehen.

Da das Paket eher als experimentell gilt, findet sich die Anleitung hier im zweiten Teil der KOMA-Script-Anleitung. Dementsprechend richtet sie sich auch in erster Linie an erfahrene Anwender. Für Anfänger oder Anwender, die sich bereits deutlich auf dem Weg zum LATEX-Experten befinden, mag daher einiges in den nachfolgenden Erklärungen unklar oder gar unverständlich sein. Ich bitte um Nachsicht, dass ich bei experimentellen Paketen den Aufwand für die Anleitung halbwegs erträglich halten will.

## 16.2. Frühe oder späte Optionenwahl

Es gilt sinngemäß, was in Abschnitt 2.4 geschrieben wurde. Falls Sie also Abschnitt 2.4 bereits gelesen und verstanden haben, können Sie auf Seite 463 mit Abschnitt 16.3 fortfahren.

```
\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 [Tea23] oder jeder LATEX-Einführung, beispielsweise [DGS+12], beschrieben.

v3.00

Gegenüber der nachfolgend vorgestellten Schnittstelle zu Einstellungen von Optionen hat `\documentclass` einen Nachteil, der unbedingt zu beachten ist: Anweisungen, Längen, Zähler und ähnliches können darin leicht zerbrechen. So führt die Verwendung einer LATEX-Länge im Wert einer Option bei dieser Anweisung bei vielen Nicht-KOMA-Script-Klassen zu einer Fehlermeldung, noch bevor der Wert an ein KOMA-Script-Paket übergeben wird, es also die Kontrolle darüber übernehmen könnte. Wertzuweisungen mit LATEX-Längen oder LATEX-Zählern sollten daher nie per `\documentclass`, sondern mit den nachfolgend dokumentierten Anweisungen `\KOMAoptions` oder `\KOMAoption` vorgenommen werden.

```
\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. Mit der Anweisung `\KOMAoptions` kann man wie bei `\documentclass` oder `\usepackage` 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 `\KOMAoption` der einen *Option* nacheinander eine Reihe von Werten zuzuweisen. Die einzelnen Werte sind dabei in der *Werteliste* durch Komma voneinander getrennt.

Soll ein *Wert* ein Gleichheitszeichen oder ein Komma enthalten, 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`«. Siehe dazu Teil II, Abschnitt 10.2, ab Seite 329.

Mit `\KOMAoptions` oder `\KOMAoption` gesetzte Optionen erreichen sowohl die KOMA-Script-Klasse als auch alle bereits geladenen KOMA-Script-Pakete, die diese Optionen kennen. Ist eine Option oder ein Wert insgesamt unbekannt, so wird die Option einschließlich des Wertes von scrbase als fehlerhaft gemeldet.

## 16.3. Textauszeichnungen

Es gilt sinngemäß, was in Abschnitt 3.6 geschrieben wurde. Falls Sie also Abschnitt 3.6 bereits gelesen und verstanden haben, können Sie auf Seite 465 mit Abschnitt 16.4 fortfahren.

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 [DGS+12], [Tea23] und [Tea05] zu entnehmen.

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

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 Schriftattribute umgeschaltet werden. In der Regel werden dies Befehle wie `\rmfamily`, `\sffamily`, `\ttfamily`, `\upshape`, `\itshape`, `\slshape`, `\scshape`, `\mdseries`, `\bfseries`, `\normalfont` oder einer der Befehle `\Huge`, `\huge`, `\LARGE`, `\Large`, `\large`, `\normalsize`, `\small`, `\footnotesize`, `\scriptsize` und `\tiny` sein. Die Erklärung zu diesen Befehlen entnehmen Sie bitte [DGS+12], [Tea23] oder [Tea05]. Auch Farbumschaltungen wie `\normalcolor` sind möglich (siehe [Car17] 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.

Mit der Anweisung \usekomafont kann die aktuelle Schriftart auf die für das angegebene *Element* umgeschaltet werden.

```
\usefontofkomafont{Element}
\useencodingofkomafont{Element}
\usesizeofkomafont{Element}
\usefamilyofkomafont{Element}
\useseriesofkomafont{Element}
\useshapeofkomafont{Element}
```

v3.12 Manchmal werden in der Schrifteinstellung eines Elements auch Dinge vorgenommen, die mit der Schrift eigentlich gar nichts zu tun haben, obwohl dies ausdrücklich nicht empfohlen wird. Soll dann nur die Schrifteinstellung, aber keine dieser zusätzlichen Einstellungen ausgeführt werden, so kann statt \usekomafont die Anweisung \usefontofkomafont verwendet werden. Diese Anweisung übernimmt nur die Schriftgröße und den Grundlinienabstand, die Codierung (engl. *encoding*), die Familie (engl. *family*), die Strichstärke oder Ausprägung (engl. *font series*) und die Form oder Ausrichtung (engl. *font shape*).

Mit den übrigen Anweisungen können auch einzelne Schriftattribute übernommen werden. Dabei übernimmt \usesizeofkomafont sowohl die Schriftgröße als auch den Grundlinienabstand.

Diese Befehle sollten jedoch nicht als Legitimation dafür verstanden werden, in die Schrifteinstellungen der Elemente beliebige Anweisungen einzufügen. Das kann nämlich sehr schnell zu Fehlern führen (siehe Abschnitt 18.5, Seite 485).

## 16.4. Deklaration neuer Notizspalten

Beim Laden des Pakets wird bereits automatisch eine Notizspalte namens marginpar deklariert. Wie der Name andeutet, liegt diese Notizspalte im Bereich der normalen Marginalienspalte von \marginpar und \marginline. Dabei werden auch die Einstellungen \reversemarginpar und \normalmarginpar beachtet, allerdings nicht für die einzelnen Notizen, sondern nur für die gesamten Notizen einer Seite. Maßgeblich ist die Einstellung, die am Ende der Seite, nämlich bei der Ausgabe der Notizspalte, gilt. Will man hingegen innerhalb einer Seite sowohl Notizen links als auch rechts neben dem Haupttext haben, so sollte man sich eine zweite Notizspalte definieren.

Die Voreinstellungen für alle neu deklarierten Notizspalten entsprechen im Übrigen den erwähnten Einstellungen für die vordefinierte marginpar. Diese können bei der Deklaration jedoch leicht geändert werden.

Es ist zu beachten, dass Notizspalten nur auf Seiten ausgegeben werden, deren Seitenstil auf dem Paket scrlayer basiert. Das Paket scrlayer wird von scrlayer-notecolumn automatisch geladen und stellt in der Voreinstellung lediglich den Seitenstil empty bereit. Werden weitere Seitenstile benötigt, wird zusätzlich das Paket scrlayer-scrpage empfohlen.

```
\DeclareNoteColumn[Liste der Einstellungen]{Name der Notizspalte}
\DeclareNewNoteColumn[Liste der Einstellungen]{Name der Notizspalte}
\ProvideNoteColumn[Liste der Einstellungen]{Name der Notizspalte}
\RedeclareNoteColumn[Liste der Einstellungen]{Name der Notizspalte}
```

Mit Hilfe dieser Anweisungen können Notizspalten angelegt werden. Dabei erzeugt `\DeclareNoteColumn` die Notizspalte ungeachtet der Tatsache, ob sie bereits existiert, während `\DeclareNewNoteColumn` einen Fehler ausgibt, falls der *Name der Notizspalte* bereits für eine andere Notizspalte vergeben ist, und `\ProvideNoteColumn` in eben diesem Fall schlicht nichts tut. Mit `\RedeclareNoteColumn` wiederum kann nur eine bereits existierende Notiz-Spalte neu konfiguriert werden.

Bei der Neukonfigurierung bereits existierender Notizspalten mit `\DeclareNoteColumn` oder `\RedeclareNoteColumn` gehen im Übrigen die bereits erzeugten Notizen für diese Spalte nicht verloren, sondern bleiben erhalten.

Für neue Notizspalten wird immer ein Element zur Änderung der Schriftattribute mit `\setkomafont` und `\addtokomafont` angelegt, falls dieses noch nicht existiert. Als Name für das Element wird `notecolumn.`*Name der Notizspalte* verwendet. Dementsprechend existiert für die vordefinierte Notizspalte `marginpar` das Element `notecolumn.marginpar`. Die Voreinstellung kann bei der Deklaration einer Notizspalte direkt über die Option `font` innerhalb der optionalen *Liste der Einstellungen* angegeben werden.

Die *Liste der Einstellungen* ist eine durch Komma separierte Liste von Einstellungen oder Optionen. Die verfügbaren Optionen sind in Tabelle 16.1 zu finden. Als Voreinstellung ist `marginpar` immer gesetzt, kann aber durch individuelle Einstellungen überschrieben werden.

Tabelle 16.1.: Mögliche Einstellungen für die Deklaration von Notizspalten

`font=`*Schriftattribute*
: Einstellung der *Schriftattribute* der Notizspalte mit Hilfe von `\setkomafont`. Für erlaubte Werte sei auf Abschnitt 16.3, Seite 463 verwiesen.
Voreinstellung: *leer*

`marginpar`
: Position und Breite der Notizspalte werden so eingestellt, dass sie der Marginalienspalte von `\marginpar` entsprechen. Eine Umschaltung zwischen `\reversemarginpar` und `\normalmarginpar` wird immer nur am Ende der Seite bei der Ausgabe der Notizspalte beachtet. Es wird darauf hingewiesen, dass diese Option kein Argument erwartet oder erlaubt.
Voreinstellung: *ja*

...

Tabelle 16.1.: Mögliche Einstellungen für die Deklaration von Notizspalten (*Fortsetzung*)

`normalmarginpar`
: Position und Breite der Notizspalte werden so eingestellt, dass sie der Marginalienspalte von `\marginpar` bei Einstellung `\normalmarginpar` entsprechen. Es wird darauf hingewiesen, dass diese Option kein Argument erwartet oder erlaubt.
Voreinstellung: *nein*

`position=`*`Abstand`*
: Die Notizspalte wird mit *`Abstand`* vom linken Rand des Papiers gesetzt. Dabei sind für *`Abstand`* auch komplexe Ausdrücke gestattet, solange diese voll expandierbar sind und zum Zeitpunkt der Ausgabe der Notizspalte zu einer Länge oder zu einem Längenwert oder einem Längenausdruck expandieren. Siehe [Tea98, Abschnitt 3.5] für weitere Informationen zu Längenausdrücken.
Voreinstellung: *durch Option* *`marginpar`*

`reversemarginpar`
: Position und Breite der Notizspalte werden so eingestellt, dass sie der Marginalienspalte von `\marginpar` bei Einstellung `\reversemarginpar` entsprechen. Es wird darauf hingewiesen, dass diese Option kein Argument erwartet oder erlaubt.
Voreinstellung: *nein*

`width=`*`Breite`*
: Die Notizspalte wird mit der angegebenen Breite gesetzt. Dabei sind für *`Breite`* auch komplexe Ausdrücke gestattet, solange diese voll expandierbar sind und zum Zeitpunkt der Ausgabe der Notizspalte zu einer Länge oder einem Längenwert oder einem Längenausdruck expandieren. Siehe [Tea98, Abschnitt 3.5] für weitere Informationen zu Längenausdrücken.
Voreinstellung: *durch Option* *`marginpar`*

Da die Notizspalten mit Hilfe von scrlayer definiert werden, wird auch für jede Notizspalte eine Ebene angelegt. Als Name für diese Ebene wird ebenfalls `notecolumn.`*`Name der Notizspalte`* verwendet. Näheres zu Ebenen ist Abschnitt 14.3, ab Seite 421 zu entnehmen.

**Beispiel:** Angenommen, Sie sind Professor für ulkiges Recht und wollen eine Abhandlung über das neue »Gesetz über die ausgelassene Verbreitung allgemeiner Späße«, kurz GüdaVaS, schreiben. Der Hauptaugenmerk soll dabei jeweils auf dem Kommentar zu einzelnen Paragraphen liegen. Sie entscheiden sich für ein zweispaltiges Layout, wobei der Kommentar in der Hauptspalte enthalten sein soll und die Paragraphen jeweils klein und in Farbe in einer schmaleren Notizspalte rechts daneben.

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

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

\usepackage{contract}
\setkomafont{contract.Clause}{\bfseries}
\setkeys{contract}{preskip=-\dp\strutbox}

\usepackage{scrlayer-scrpage}
\usepackage{scrlayer-notecolumn}

\newlength{\paragraphscolwidth}
\AfterCalculatingTypearea{%
  \setlength{\paragraphscolwidth}{%
    .333\textwidth}%
  \addtolength{\paragraphscolwidth}{%
    -\marginparsep}%
}
\recalctypearea
\DeclareNewNoteColumn[%
  position=\oddsidemargin+1in
           +.667\textwidth
           +\marginparsep,
  width=\paragraphscolwidth,
  font=\raggedright\footnotesize
       \color{blue}
]{paragraphs}
```

Es wird ein einseitiger Artikel verfasst. Dazu wird die Sprache mit Hilfe des babel-Pakets auf Deutsch (neue Rechtschreibung) festgelegt. Bezüglich der Eingabecodierung wird von UTF-8 und einer LaTeX-Version ab 2018-04-01 ausgegangen. Als Schrift wird Latin Modern in 8-Bit-Codierung verwendet. Für die Farbeinstellungen wird das Paket xcolor genutzt.

Bezüglich des Setzens von Gesetzestexten mit contract sei auf dessen Anleitung verwiesen.

Da ein Seitenstil mit Seitenzahl verwendet werden soll, wird das Paket scrlayer-scrpage geladen. Somit können Notizspalten auf allen Seiten ausgegeben werden.

Dann wird das Paket scrlayer-notecolumn für die Notizspalten geladen. Die gewünschte Breite der Notizspalte wird über `\AfterCalculatingTypearea` nach jeder etwaigen Neuberechnung des Satzspiegels neu berechnet. Sie soll jeweils ein Drittel der Satzspiegelbreite betragen, wobei der Abstand zwischen Text und Notizspalte zu Lasten der Notizspalte geht. Diese ist also effektiv um `\marginparsep` schmaler.

Mit dieser Information kann dann die neue Notizspalte definiert werden. Bei der Festlegung der Position wird ein einfacher Längenausdruck genutzt. Dabei ist zu beachten, dass `\oddsidemargin` nicht der gesamte linke Rand ist, sondern aus historischen Gründen der linke Rand abzüglich 1 inch. Daher muss dieser Wert noch hinzugezählt werden.

Damit ist die Deklaration abgeschlossen. Es ist zu beachten, dass die Notizspalte bisher im Textbereich ausgegeben wird. Die Notizspalte würde also den Text überschreiben.

```
\begin{document}

\title{Kommentar zum GüdaVaS}
\author{Professor R. O. Tenase}
\date{11.\,11.~2011}
\maketitle
\tableofcontents

\section{Vormerkung}
Das GüdaVaS ist ohne jeden Zweifel das wichtigste
Gesetz, das in Spaßmanien in den letzten eintausend
Jahren verabschiedet wurde. Die erste Lesung fand
bereits am 11.\,11.~1111 im obersten spaßmanischen
Kongress statt, wurde aber vom damaligen Spaßvesier
abgelehnt. Erst nach Umwandlung der spaßmanischen,
aberwitzigen Monarchie in eine repräsentative,
witzige Monarchie durch W. Itzbold,
den Urkomischen, am 9.\,9.~1999 war der Weg für
dieses Gesetz endlich frei.
```

Dadurch, dass der Textbereich nicht verkleinert wurde, wird hier der ganze Vorspann über die Gesamtbreite ausgegeben. Um das Beispiel zu testen, können Sie vorübergehend

```
\end{document}
```

ergänzen.

Offen blieb in dem Beispiel die Frage, wie der Text für den Kommentar schmaler gesetzt werden kann. Dies werden Sie bei der Fortsetzung des Beispiels erfahren.

## 16.5. Erstellen einer Notiz

Nachdem eine Notizspalte deklariert wurde, können Notizen für diese Spalte erstellt werden. Diese Notizen werden allerdings nicht unmittelbar ausgegeben, sondern zunächst nur in eine Hilfsdatei mit Endung »`.slnc`« geschrieben. Ganz genau werden die Notizen sogar zunächst in

die aux-Datei geschrieben und erst beim Lesen der aux-Datei innerhalb von \end{document} in die slnc-Datei übertragen. Dabei wird gegebenenfalls auch die Einstellung \nofiles beachtet. Beim nächsten LaTeX-Lauf wird diese Hilfsdatei dann Stück für Stück je nach Fortschritt des Dokuments wieder eingelesen und am Ende der Seite werden die Notizen für die jeweilige Seite ausgegeben.

Es ist jedoch zu beachten, dass Notizspalten nur auf Seiten ausgegeben werden, deren Seitenstil auf dem Paket scrlayer basiert. Das Paket scrlayer wird von scrlayer-notecolumn automatisch geladen und stellt in der Voreinstellung lediglich den Seitenstil empty bereit. Werden weitere Seitenstile benötigt, wird zusätzlich das Paket scrlayer-scrpage empfohlen.

```
\makenote[Name der Notizspalte]{Notiz}
\makenote*[Name der Notizspalte]{Notiz}
```

Mit Hilfe dieser Anweisungen kann eine *Notiz* erstellt werden. Dabei wird die aktuelle vertikale Position als vertikale Position für den Anfang der *Notiz* verwendet. Die horizontale Position für die Notiz ergibt sich aus der definierten Position der Notizspalte. Für die korrekte Funktion ist das Paket dabei auf \pdfsavepos, \pdflastypos und \pdfpageheight beziehungsweise deren Entsprechungen bei neueren LuaTeX-Versionen angewiesen. Ohne diese Befehle funktioniert scrlayer-notecolumn nicht.

Wird allerdings bei der Ausgabe der *Notiz* eine Kollision mit einer früheren Notiz in derselben Notizspalte erkannt, so wird die *Notiz* bis unter diese frühere Notiz verschoben. Passt die *Notiz* nicht mehr auf die Seite, so wird sie ganz oder teilweise auf die nächste Seite umbrochen.

Für welche Notizspalte die *Notiz* erstellt werden soll, wird über das optionale Argument *Name der Notizspalte* bestimmt. Ist kein optionales Argument angegeben, so wird die vordefinierte Notizspalte marginpar verwendet.

**Beispiel:** Fügen wir nun dem Beispiel aus dem vorherigen Abschnitt einen kommentierten Paragraphen hinzu, wobei der Paragraph selbst in der neu definierten Notizspalte gesetzt werden soll.

```
\section{Analyse}
\begin{addmargin}[0pt]{.333\textwidth}
  \makenote[paragraphs]{%
    \protect\begin{contract}
      \protect\Clause[%
        title={Kein Witz ohne Publikum}%
      ]
      Ein Witz kann nur dort witzig sein, wo er
      auf ein Publikum trifft.
    \protect\end{contract}%
  }
  Dies ist eine der zentralsten Aussagen des
  Gesetzes. Sie ist derart elementar, dass es
  durchaus angebracht ist, sich vor der Weisheit
```

```
der Verfasser zu verbeugen.
```

Die in Abschnitt 3.18, Seite 134 dokumentierte Umgebung `addmargin` wird genutzt, um den Haupttext in der Breite um die Spalte für die Paragraphen zu vermindern.

Hier ist auch eines der wenigen Probleme bei Verwendung von `\makenote` zu erkennen. Da das obligatorische Argument in Dateien geschrieben wird, können Befehle innerhalb des Arguments leider *zerbrechen*. Um das zu verhindern, wird empfohlen, vor alle Befehle ein `\protect` zu setzen. Anderenfalls kann die Verwendung von Befehlen innerhalb dieses Arguments zu Fehlermeldungen führen.

Prinzipiell könnten Sie das Beispiel nun bereits mit

```
\end{addmargin}
\end{document}
```

beenden, wenn Sie ein Ergebnis sehen wollen.

Beim Testen des Beispiels, werden Sie feststellen, dass die Gesetzesspalte tiefer hinunter reicht als der Kommentartext. Wenn Sie zwecks Übung einen weiteren Abschnitt mit einem weiteren Paragraphen hinzufügen, ergibt sich eventuell das Problem, dass der Kommentar nicht unterhalb des Gesetzestextes, sondern direkt im Anschluss an den bisherigen Kommentar fortgesetzt wird. Eine Lösung für dieses Problem werden Sie gleich kennenlernen.

v0.1.2583

Das im Beispiel erwähnte Problem mit dem Zerbrechen von Befehlen tritt bei der Sternvariante normalerweise nicht auf. Diese verwendet `\detokenize`, um die Expansion zu verhindern. Das bedeutet aber auch, dass man in der *`Notiz`* keine Befehle verwenden sollte, die ihre Bedeutung innerhalb des Dokuments verändern.

Allerdings treten bei beiden Formen zwei andere Probleme auf. Das erste betrifft die Verwendung von Farbe mit Hilfe von color oder xcolor innerhalb der Notizspalten. Um Farbumschaltungen innerhalb der Notizspalten zu ermöglichen, wäre für jede Notizspalte eine eigene Farbverwaltung mit einem sogenannten *Color Stack* notwendig. Da das Paket lediglich als Machbarkeitsstudie entworfen ist und XeTeX nicht mehrere *Color Stacks* unterstützt, sind mit XeTeX Farbumschaltungen nur eingeschränkt über die Schriftattribute des Elements `notecolumn.`*`Name der Notizspalte`* möglich, wodurch der Aufwand für die Implementierung einer eigenen Farbverwaltung umgangen wurde.

Das zweite, eher konzeptionelle Problem ist, dass die Hilfsdatei mit den Informationen zum Inhalt der Notizspalte während der Verarbeitung der Kopfzeile einer Seite eingelesen wird. Das hat vor allem dann Auswirkungen, wenn dies geschieht, während eine Umgebung wie `verbatim` aktiv ist. In diesem Fall wären während des Einlesens der Hilfsdatei die `\catcode`-Einstellungen dieser Umgebung aktiv. Das würde zwangsläufig zu Fehlern in der Verarbeitung und Ausgabe führen. Um dies abzumildern, werden während `\begin{document}` die `\catcode`-Einstellungen der in `\dospecials` gespeicherten Zeichen gespeichert und während dem Einlesen der Hilfsdatei explizit wiederhergestellt.

\syncwithnotecolumn[*Name der Notizspalte*]

Mit Hilfe dieser Anweisung wird in einer Notizspalte und im Haupttext des Dokuments je ein Synchronisierungspunkt erstellt. Wann immer bei der Ausgabe einer Notizspalte oder des Haupttextes ein solcher Synchronisierungspunkt erreicht wird, wird eine Marke angelegt, deren Inhalt die aktuelle Seite und die aktuelle vertikale Position ist.

Parallel zum Erstellen der Synchronisierungspunkte wird ermittelt, ob in der Notizspalte und im Haupttext beim letzten LaTeX-Lauf eine Marke angelegt wurde. Falls das der Fall ist, werden deren Werte miteinander verglichen. Liegt die Marke der Notizspalte tiefer auf der Seite oder auf einer späteren Seite, so wird im Haupttext bis zu der Stelle der Notizspalte vorgerückt.

In der Regel sollten Synchronisierungspunkte nicht innerhalb eines Absatzes des Haupttextes, sondern nur zwischen diesen gesetzt werden. Wird `\syncwithnotecolumn` dennoch innerhalb eines Absatzes verwendet, so wird der Synchronisierungspunkt im Haupttext tatsächlich erst nach der aktuellen Zeile eingefügt. In dieser Hinsicht ähnelt `\syncwithnotecolumn` also beispielsweise `\vspace`.

Dadurch, dass Synchronisierungspunkte in den Notizspalten erst beim nächsten LaTeX-Lauf erkannt werden, benötigt der Mechanismus mindestens drei LaTeX-Läufe. Aus jeder neuen Synchronisierung können sich außerdem Verschiebungen für spätere Synchronisierungspunkte ergeben, was wiederum die Notwendigkeit weiterer LaTeX-Läufe nach sich zieht. Zu erkennen sind solche Verschiebungen in der Regel an der Meldung: »LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.« Aber auch Meldungen über undefinierte *Labels* können auf die Notwendigkeit eines weiteren LaTeX-Laufs hinweisen.

Wird das optionale Argument nicht angegeben, so wird an seiner Stelle `marginpar`, also die vordefinierte Notizspalte verwendet. Es sei an dieser Stelle darauf hingewiesen, dass ein leeres optionales Argument nicht gleichbedeutend mit dem Weglassen eines optionalen Arguments ist!

Es ist nicht erlaubt, `\syncwithnotecolumn` innerhalb einer Notiz selbst, also im obligatorischen Argument von `\makenote` zu verwenden! Dieser Fehler kann derzeit nicht abgefangen werden und führt dazu, dass mit jedem LaTeX-Lauf neue Verschiebungen auftreten, so dass nie ein endgültiger Zustand erreicht wird. Um zwei oder mehrere Notizspalten miteinander zu synchronisieren, sind sie stattdessen mit dem Haupttext zu synchronisieren, da dabei auch die Spalten miteinander synchronisiert werden. Die hierzu empfohlene Anweisung wird nachfolgend beschrieben.

**Beispiel:** Führen wir nun das obige Beispiel fort, indem wir zunächst einen Synchronisationspunkt und dann einen weiteren Paragraphen mit Kommentar hinzufügen:

```
\syncwithnotecolumn[paragraphs]\bigskip
\makenote[paragraphs]{%
  \protect\begin{contract}
    \protect\Clause[title={Komik der Kultur}]
    \setcounter{par}{0}%
```

```
    Die Komik eines Witzes kann durch das
    kulturelle Umfeld, in dem er erzählt wird,
    bestimmt sein.

    Die Komik eines Witzes kann durch das
    kulturelle Umfeld, in dem er spielt,
    bestimmt sein.
  \protect\end{contract}
}
Die kulturelle Komponente eines Witzes ist
tatsächlich nicht zu vernachlässigen. Über die
politische Korrektheit der Nutzung des
kulturellen Umfeldes kann zwar treff"|lich
gestritten werden, nichtsdestotrotz ist die
Treffsicherheit einer solchen Komik im
entsprechenden Umfeld frappierend. Auf der
anderen Seite kann ein vermeintlicher Witz im
falschen kulturellen Umfeld auch zu einer
echten Gefahr für den Witzeerzähler werden.
```

Außer dem Synchronisationspunkt wurde hier auch noch ein vertikaler Abstand mit `\bigskip` eingefügt, um die einzelnen Paragraphen und ihre Kommentare besser voneinander abzusetzen.

Außerdem wird hier ein weiterer Punkt, der zu einem Problem werden kann, sichtbar. Da die Notizspalten mit Boxen arbeiten, die zusammengebaut und zerlegt werden, kann es bei Zählern innerhalb der Notizspalten teilweise zu Verschiebungen kommen. Im Beispiel würde daher der erste Absatz nicht mit 1, sondern mit 2 nummeriert. Dies kann jedoch mit einem beherzten Zurücksetzen des entsprechenden Zählers leicht korrigiert werden.

Das Beispiel ist damit fast fertig, was noch fehlt, ist das Ende der Umgebungen:

```
\end{addmargin}
\end{document}
```

Tatsächlich wären natürlich auch noch die restlichen Paragraphen des Gesetzes zu kommentieren. Dies sei mir hier jedoch erlassen.

Doch halt! Was wäre, wenn in diesem Beispiel der Paragraph nicht mehr auf die Seiten passen würde? Würde er dann auf der nächsten Seite ausgegeben? Diese Frage wird im nächsten Abschnitt beantwortet werden.

\syncwithnotecolumns[*Liste von Notizspalte*]

Diese Anweisung führt eine Synchronisierung des Haupttextes mit allen in der mit Komma separierten *Liste von Notizspalten* angegebenen Notitzspalten durch. Dabei wird der Haupttext mit der Notizspalte synchronisiert, deren Marke am weitesten hinten im Dokument steht. Somit werden als Nebeneffekt auch die Notizspalten untereinander synchronisiert.

Wird das optionale Argument nicht angegeben oder ist es leer, so wird mit allen deklarierten Notizspalten synchronisiert.

## 16.6. Erzwungene Ausgabe von Notizspalten

Im vorherigen Abschnitt wurde dokumentiert, wie die Ausgabe der Notitzspalten normalerweise erfolgt. Manchmal ist es aber auch erforderlich, alle aufgesammelten Notizen, die noch nicht ausgegeben wurden, unmittelbar auszugeben. Das ist insbesondere dann sinnvoll, wenn längere Notizen dazu führen, dass immer mehr Notizen nach unten und auf neue Seiten verschoben werden. Ein guter Zeitpunkt für eine solche erzwungene Ausgabe ist beispielsweise das Ende eines Kapitels oder das Ende des Dokuments.

\clearnotecolumn[*Name der Notizspalte*]

Mit dieser Anweisung werden alle Notizen einer bestimmen Notizspalte ausgegeben, die bis zum Ende der aktuellen Seite noch nicht ausgegeben sind, aber auf dieser oder einer vorherigen Seite erstellt wurden. Zur Ausgabe dieser noch anhängigen Notizen werden nach Bedarf Leerseiten erstellt. Während der Ausgabe der anhängigen Notizen dieser Notizspalte werden gegebenenfalls auch anhängige Notizen anderer Notizspalten ausgegeben, jedoch nur so lange, wie dies zur Ausgabe der anhängigen Notizen der angegebenen Notizspalte notwendig ist.

Während der Ausgabe der anhängigen Notizen kann es auch geschehen, dass irrtümlich Notizen ausgegeben werden, die im vorherigen LaTeX-Lauf auf den Seiten erstellt wurden, die nun durch die eingefügten Leerseiten ersetzt werden. Dies normalisiert sich in einem der nächsten LaTeX-Läufe. Zu erkennen sind solche Verschiebungen in der Regel an der Meldung: »LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.«

Die Notizspalte, deren anhängige Notizen ausgegeben werden soll, ist über das optionale Argument *Name der Notizspalte* angegeben. Ist kein solches Argument angegeben, so wird die vordefinierte Notizspalte `marginpar` verwendet.

Dem aufmerksamen Leser wird nicht entgangen sein, dass die erzwungene Ausgabe einer Notizspalte der Synchronisierung nicht unähnlich ist. Allerdings befindet man sich nach der erzwungenen Ausgabe im Fall, dass tatsächlich eine Ausgabe stattfindet, am Anfang der Seite nach der letzten Ausgabe und nicht unmittelbar unterhalb der letzten Ausgabe. Dafür terminiert die erzwungene Ausgabe in der Regel mit weniger LaTeX-Läufen.

`\clearnotecolumns[`*Liste von Notizspalten-Namen*`]`

Diese Anweisung arbeitet vergleichbar mit `\clearnotecolumn`. Allerdings kann hier als optionales Argument nicht nur eine Notizspalte angegeben werden, sondern es ist eine durch Komma getrennte Liste mehrerer Namen von Notizspalten erlaubt. Es werden dann die anhängigen Notizen all dieser Spalten ausgegeben.

Wurde das optionale Argument nicht angegeben oder ist es leer, so werden die anhängigen Notizen aller Notizspalten ausgegeben.

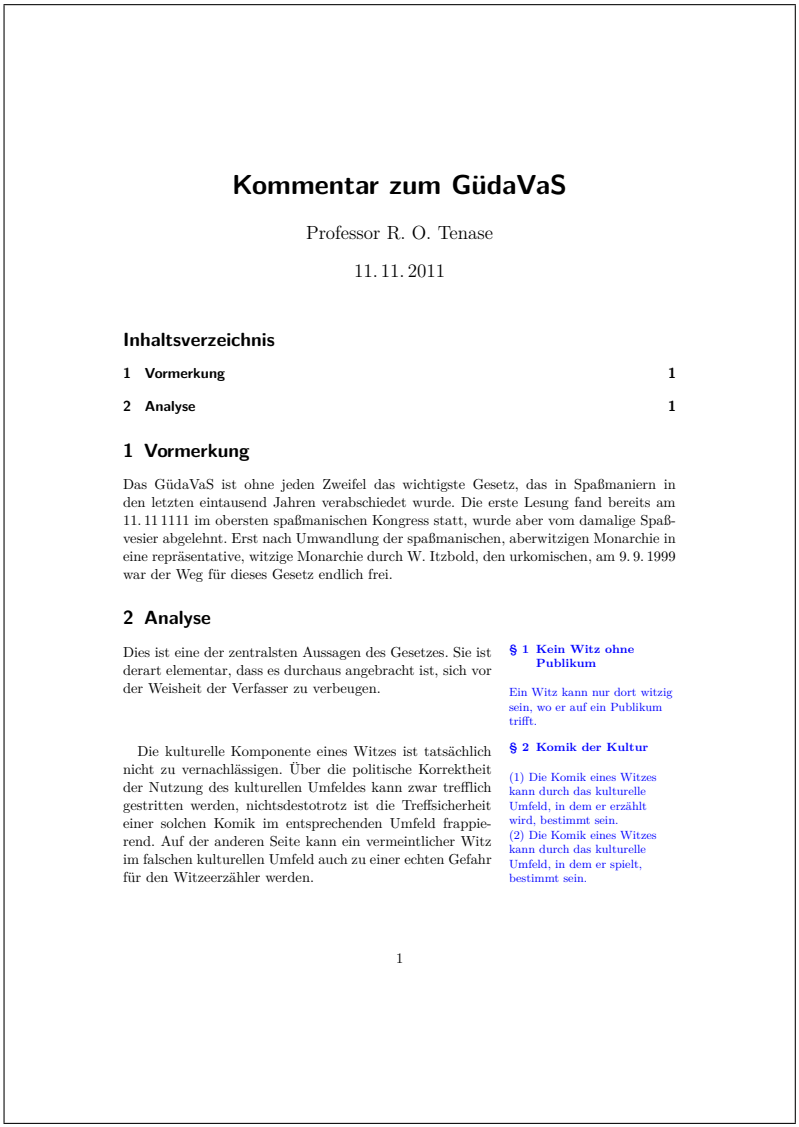
`autoclearnotecolumns=`*Ein-Aus-Wert*

In der Regel wird man anhängige Notizen immer dann ausgeben lassen, wenn im Dokument explizit oder implizit – beispielsweise in Folge von `\chapter` – die Anweisung `\clearpage` ausgeführt wird. Dies ist auch am Ende eines Dokuments innerhalb von `\end{document}` der Fall. Über die Option `autoclearnotecolumns` kann daher gesteuert werden, ob bei Ausführung von `\clearpage` automatisch auch `\clearnotecolumns` ohne Argument ausgeführt werden soll. Da davon ausgegangen wird, dass dies in der Regel erwünscht ist, ist die Option in der Voreinstellung aktiv. Man kann sie jedoch über die entsprechenden Werte für einfache Schalter (siehe Tabelle 2.5 auf Seite 41) jederzeit aus- und auch wieder einschalten.

Es ist zu beachten, dass im Falle der Deaktivierung der automatischen Ausgabe anhängiger Notizen am Ende des Dokuments Notizen ganz oder teilweise verloren gehen können. Daher sollte man in diesem Fall vor `\end{document}` sicherheitshalber ein `\clearnotecolumns` einfügen.

Damit ist nun auch die Frage nach dem Beispiel im letzten Abschnitt beantwortet, ob der Paragraph auch komplett ausgegeben würde, wenn er auf die nächste Seite umbrochen werden müsste. Dies ist in der Voreinstellung selbstverständlich der Fall. Da es jedoch nach dem Ende der `addmargin`-Umgebung geschehen würde, könnte es eventuell noch zu Überlappungen durch nachfolgenden Text kommen. Daher wäre es im Beispiel durchaus sinnvoll, nach der `addmargin`-Umgebung einen weiteren Synchronisationspunkt einzufügen.

Das Ergebnis des Beispiels ist übrigens in Abbildung 16.1 zu sehen.



# Kommentar zum GüdaVaS

Professor R. O. Tenase

11. 11. 2011

## Inhaltsverzeichnis

1 Vormerkung 1

2 Analyse 1

## 1 Vormerkung

Das GüdaVaS ist ohne jeden Zweifel das wichtigste Gesetz, das in Spaßmaniern in den letzten eintausend Jahren verabschiedet wurde. Die erste Lesung fand bereits am 11. 11 1111 im obersten spaßmanischen Kongress statt, wurde aber vom damalige Spaßvesier abgelehnt. Erst nach Umwandlung der spaßmanischen, aberwitzigen Monarchie in eine repräsentative, witzige Monarchie durch W. Itzbold, den urkomischen, am 9. 9. 1999 war der Weg für dieses Gesetz endlich frei.

## 2 Analyse

Dies ist eine der zentralsten Aussagen des Gesetzes. Sie ist derart elementar, dass es durchaus angebracht ist, sich vor der Weisheit der Verfasser zu verbeugen.

Die kulturelle Komponente eines Witzes ist tatsächlich nicht zu vernachlässigen. Über die politische Korrektheit der Nutzung des kulturellen Umfeldes kann zwar trefflich gestritten werden, nichtsdestotrotz ist die Treffsicherheit einer solchen Komik im entsprechenden Umfeld frappierend. Auf der anderen Seite kann ein vermeintlicher Witz im falschen kulturellen Umfeld auch zu einer echten Gefahr für den Witzeerzähler werden.

**§ 1 Kein Witz ohne Publikum**

Ein Witz kann nur dort witzig sein, wo er auf ein Publikum trifft.

**§ 2 Komik der Kultur**

(1) Die Komik eines Witzes kann durch das kulturelle Umfeld, in dem er erzählt wird, bestimmt sein.
(2) Die Komik eines Witzes kann durch das kulturelle Umfeld, in dem er spielt, bestimmt sein.



Abbildung 16.1.: Eine Ergebnisseite zu dem Beispiel aus diesem Kapitel

# 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 [Koh20a] vorbehalten. Dies sollte kein Problem sein, denn der normale Anwender, der das Paket einfach nur verwenden will, wird diese Informationen eher selten 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.

## 17.1. Experimentelle Möglichkeiten

In diesem Abschnitt werden experimentelle Möglichkeiten beschrieben. Experimentell bedeutet in diesem Zusammenhang, dass die Funktion nicht garantiert werden kann.

`usegeometry=`*Ein-Aus-Wert*

Normalerweise kümmert sich typearea wenig darum, ob es in irgend einer Konstellation zusammen mit dem Paket geometry (siehe [Ume10]) verwendet wird. Das bedeutet insbesondere, dass geometry nach wie vor nichts davon mitbekommt, wenn man mit typearea beispielsweise die Papiergröße ändert – etwas, das geometry selbst gar nicht bietet.

v3.17 Sobald Option `usegeometry` gesetzt wird, versucht typearea, alle eigenen Optionen für geometry in dessen Optionen zu übersetzen. Innerhalb des Dokuments wird sogar `\newgeometry` aufgerufen, wenn neue Parameter aktiviert werden (siehe `\activateareas` im nachfolgenden Abschnitt). Da geometry keine Änderung der Papiergröße oder Papierausrichtung via `\newgeometry` bietet, wird diese bei Bedarf über interne Anweisungen und Längen von geometry umgesetzt. Getestet ist dies für geometry 5.3 bis 5.6.

Die Option bedeutet übrigens nicht, dass bei Verwendung von geometry nach einer Papiergrößen- oder Papierausrichtungsänderung mit typearea das neue Papier direkt optimal genutzt wird. Da geometry aus Komfortgründen deutlich mehr Optionen für die Papiereinstellung bietet, als für die Bestimmung von Textbereich, Rändern, Kopf, Fuß etc. benötigt werden – man spricht von *Überbestimmung* – und gleichzeitig bei neuen Aufrufen von `\newgeometry` fehlende Angaben aus bereits bekannten ableitet – man spricht von *Werterhalt* –, muss man gegebenenfalls durch vollständige Bestimmung neuer Werte mit einem eigenen Aufruf von `\newgeometry` alle gewünschten Einstellungen explizit vornehmen. Nichtsdestotrotz kann die Berücksichtigung von geometry durch typearea zusätzliche Möglichkeiten eröffnen.

Von typearea werden mit `usegeometry` für geometry derzeit die Optionen `bindingoffset`, `footskip`, `headheight`, `headsep`, `includefoot`, `includehead`, `includemp`, `lmargin`,

marginparsep, marginparwidth, textheight, textwidth, top und in der Dokumentpräambel zusätzlich paperheight und paperwidth gesetzt.

areasetadvanced=*Ein-Aus-Wert*
\areaset[*BCOR*]{*Breite*}{*Höhe*}

v3.11

Normalerweise berücksichtigt \areaset Optionen zur Bestimmung der Höhe von Kopf und Fuß oder zur Festlegung, ob Randelemente Teil des Satzspiegels sein sollen, nicht in gleicher Weise wie \typearea. Mit Option areasetadvanced kann jedoch eingestellt werden, dass sich \areaset diesbezüglich mehr wie \typearea verhalten soll. Trotzdem unterscheiden sich Einstellungen, die zu gleich großen Textbereichen führen zwischen den beiden Befehlen weiterhin, da \typearea immer auf ganze Zeilen rundet und dabei gegebenenfalls den unteren Rand um bis zu eine Zeile kleiner wählt, während \areaset den oberen und unteren Rand immer im Verhältnis 1:2 einstellt. Die Textbereiche der unterschiedlichen Befehle können also bei gleicher Größe vertikal leicht verschoben sein.

Nach Aktivierung von areasetadvanced kann jedoch mit einer Kombination wie

```
\typearea{BCOR}{DIV}
\areaset[current]{\textwidth}{\textheight}
```

das unterschiedliche Verhalten ausgeglichen werden, ohne dass dabei die eingangs erwähnten Abstriche in der Arbeitsweise von \areaset in Kauf genommen werden müssen. Siehe dazu auch das Beispiel im nächsten Abschnitt.

## 17.2. 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.

\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}
\BeforeRestoreareas{Code}
\BeforeRestoreareas*{Code}
\AfterRestoreareas{Code}
\AfterRestoreareas*{Code}
```

Mit Hilfe von `\storeareas` wird eine *Anweisung* definiert, über die alle aktuellen Seitenspiegeleinstellungen wiederhergestellt 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:** Immer wieder fragen Anwender nach der Möglichkeit, in einem A4-Dokument unter Beibehaltung der Randeinstellungen eine A3-Seite im Querformat einfügen zu können. Der Textbereich soll also lediglich um die gewonnene Papierbreite verbreitert werden. Mit `\storeareas`, `\areaset` und der zuvor bereits erklärten Option `areasetadvanced` ist das sehr einfach möglich:

```
\documentclass[areasetadvanced]{scrartcl}
\areaset[current]{\textwidth}{\textheight}

\begin{document}
\noindent\rule{\textwidth}{\textheight}

\storeareas\meinegespeichertenWerte
\KOMAoption{paper}{a3,landscape}
\areaset[current]
        {\dimexpr\textwidth+.5\paperwidth\relax}
        {\textheight}
\noindent\rule{\textwidth}{\textheight}

\clearpage
\meinegespeichertenWerte
\noindent\rule{\textwidth}{\textheight}
\end{document}
```

Die `\areaset`-Anweisung in der Dokumentpräambel dient dabei zusammen mit Option `areasetadvanced` dazu, die Unterschiede zwischen der `\typearea`-Voreinstellung und der späteren Verwendung von `\areaset` auszugleichen, wie dies bereits auf Seite 478 erklärt wurde. Ohne diese Maßnahme könnte es insbesondere bei Verwendung von Optionen wie `headinclude` und `headheight` zu deutlichen Abweichungen bei den Randeinstellungen kommen.

Wichtig ist auch die Anweisung `\clearpage` vor dem Aufruf von `\meinegespeichertenWerte`, damit die Wiederherstellung erst auf der nächsten Seite erfolgt. Bei doppelseitigen Dokumenten sollte bei Änderungen am

Papierformat stattdessen sogar \cleardoubleoddpage oder – wenn keine KOMA-Script-Klasse zum Einsatz kommt – \cleardoublepage verwendet werden.

Außerdem wird \noindent verwendet, um den normalen Absatzeinzug vor den schwarzen Kästen zu verhindern. Sie würden sonst kein korrektes Bild des Seitenlayouts wiedergeben.

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 definierten *Anweisung* wird eine entsprechende Fehlermeldung ausgegeben.

v3.18 Oftmals ist auch erwünscht, vor der Wiederherstellung der Einstelllungen per *Anweisung* grundsätzlich bestimmte Aktionen wie beispielsweise ein \cleardoubleoddpage auszuführen. Dies kann man mit Hilfe von \BeforeRestoreareas und \BeforeRestoreareas* erreichen. Entsprechend kann man mit \AfterRestoreareas und \AfterRestoreareas* *Code* nach der Wiederherstellung der Einstellungen ausführen lassen. Die Formen mit und ohne Stern unterscheiden sich insoweit, als die Sternform nur für noch nicht per \storeareas gespeicherte Einstellungen gilt, während sich die Variante ohne Stern auch auf die zukünftige Verwendung bereits früher gespeicherter Einstellungen auswirkt.

```
\AfterCalculatingTypearea{Anweisungen}
\AfterCalculatingTypearea*{Anweisungen}
\AfterSettingArea{Anweisungen}
\AfterSettingArea*{Anweisungen}
```

Diese Anweisungen dienen der Verwaltung zweier Haken (engl. *hooks*). Die ersten beiden, \AfterCalculatingTypearea und deren Sternform, ermöglichen 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. Entsprechendes leisten \AfterSettingArea und dessen Stern-Form für die Ausführung von \areaset. Die Normalformen arbeiten dabei global, während die Änderungen durch die Sternformen nur lokal wirksam sind. Die *Anweisungen* werden jeweils unmittelbar vor \activateareas ausgeführt.

v3.11

## 17.3. Lokale Einstellungen durch die Datei typearea.cfg

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

## 17.4. Mehr oder weniger obsolete Optionen und Anweisungen

Im KOMA-Script-Buch [Koh20a] 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 [Koh20a] vorbehalten. Dies sollte kein Problem sein, denn der normale Anwender, der die Klassen einfach nur verwenden will, wird diese Informationen eher selten benötigen. Ein Teil der Informationen richtet sich an Anwender, die ausgefallene Aufgaben lösen oder eigene Klassen schreiben wollen, die auf einer KOMA-Script-Klasse basieren. Da sich die entsprechenden Erklärungen ausdrücklich nicht an LaTeX-Anfänger richten, sind sie teilweise deutlich kürzer gefasst und setzen ein vertieftes Wissen über LaTeX voraus. Anderes existiert nur aus Gründen der Kompatibilität zu den Standardklassen oder früheren Versionen von KOMA-Script.

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

## 18.1. Ergänzungen zu Benutzeranweisungen

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

```
\ExecuteDoHook{footnote/text/begin}
\ExecuteDoHook{footnote/text/end}
```

v3.36 Für Experten gibt es außerdem zwei Haken vom Typ *do-hook* (siehe Abschnitt 10.8 ab Seite 349). Der erste davon wird ganz zu Beginn von `\@makefntext` noch vor `\raggedfootnote` ausgeführt. Der zweite am Ende noch bevor der Absatz beendet ist. Derzeit werden beide Haken von KOMA-Script selbst nicht verwendet.

## 18.2. Zusammenspiel von KOMA-Script und anderen Paketen

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

## 18.3. Erkennung von KOMA-Script-Klassen

Für Paketautoren besteht manchmal die Notwendigkeit, eine KOMA-Script-Klasse zu erkennen. Für Anwender besteht diese Notwendigkeit dagegen eher nicht. Bezüglich der verwendeten KOMA-Script-Version sei auf `\KOMAScriptVersion` in Abschnitt 10.5, Seite 347 verwiesen.

\KOMAClassName
\ClassName

In \KOMAClassName ist der Name der aktuell verwendeten KOMA-Script-Klasse abgelegt. Will man also wissen, ob eine KOMA-Script-Klasse verwendet wird, so kann man einfach mit \@ifundefined auf diese Anweisung testen. Tests auf eine konkrete KOMA-Script-Klasse, beispielsweise mit \Ifstr, sind ebenfalls möglich. Demgegenüber gibt \ClassName Auskunft, welche Standardklasse durch diese KOMA-Script-Klasse ersetzt wird.

## 18.4. Einträge ins Inhaltsverzeichnis

KOMA-Script-Klassen bieten erweiterte Möglichkeiten zur Erstellung und Manipulation von Einträgen in das Inhaltsverzeichnis. Einige davon basieren auf der Verwendung von tocbasic (siehe Abschnitt 13.3 ab Seite 378). Andere sind direkt in den Klassen implementiert.

\raggedchapterentry

v3.21 Bei früheren Version von KOMA-Script gab es die Möglichkeit, das Makro \raggedchapterentry als \raggedright zu definieren, um den Text der Kapiteleinträge ins Inhaltsverzeichnis im linksbündigen Flattersatz zu setzen. Offiziell existiert diese Möglichkeit seit KOMA-Script-Version 3.21 nicht mehr.

Tatsächlich ist aber die Eigenschaft raggedentrytext für den Eintrags-Stil tocline im Paket tocbasic so implementiert, dass sie das Makro \ragged*Eintragsebene*entry entweder auf \relax oder auf \raggedright setzt. Bei der Auswertung der Eigenschaft wird dann getestet, ob das entsprechende Makro entweder \raggedright ist oder als \raggedright definiert ist. In beiden Fällen wird Flattersatz verwendet. In allen anderen Fällen wird kein Flattersatz verwendet.

Da schon früher dokumentiert war, dass \raggedchapterentry nicht als etwas anderes als \raggedright definiert werden sollte, ist damit Kompatibilität zum dokumentierten Verhalten früherer Versionen erreicht. Wie in früheren Versionen gewarnt, führen andere Definitionen von \raggedchapterentry – nun aber auch von \raggedsectionentry und entsprechend für die anderen Eintragsebenen – möglicherweise zu unerwarteten Ergebnissen.

Empfohlen wird, eventuell gewünschten Flattersatz für Verzeichniseinträge stattdessen über die genannte Eigenschaft des Verzeichniseintragsstils tocline zu wählen.

\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 an, also part, chapter, section, subsection, subsubsection, paragraph oder subparagraph.

Die formatierte Gliederungsnummer wird über das zweite Argument *Nummer* übergeben. Dieses Argument darf auch leer sein. Der Text des Eintrags wird mit *Überschrift* angegeben. Zerbrechliche Befehle in diesem Argument sind 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}{%
  \protect\nonumberline 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 dann:

```
\addcontentsline{toc}{Ebene}{%
  \protect\numberline{Nummer}Überschrift%
}
```

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]{%
  \IfArgIsEmpty{#3}{%
  }{%
    \IfArgIsEmpty{#2}{%
      \addcontentsline{toc}{#1}{\protect\nonumberline#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 107.

```
\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 Anweisung 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 weitergeben.

## 18.5. Schrifteinstellungen

KOMA-Script-Klassen verfügen nicht nur über erweiterte Möglichkeiten zur Auswahl der Grundschriftgröße. Sie erlauben auch die Definition von Elementen mit eigenen Schrifteinstellungen, sowie deren Manipulation und dedizierter Anwendung.

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

```
\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 60) 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 60) für dieses Element in seine Definitionen einbaut. Der Aufruf von \newkomafont für ein bereits existierendes Element führt zu Fehlermeldungen.

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. Als Urheber der Warnung wird das Paket scrkbase angegeben.

Mit \aliaskomafont kann für ein bereits existierendes *Element* ein *Aliasname* definiert werden. KOMA-Script informiert den Benutzer in der log-Datei 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 regen Gebrauch.

\addtokomafontrelaxlist{*Makro*}
\addtokomafontonearglist{*Makro*}
\addtokomafontgobblelist{*Makro*}

Wie bereits in Teil I der Anleitung erklärt, dürfen in den Schrifteinstellungen der Elemente nur Befehle zur Wahl der Größe, Familie, Codierung, Strichstärke, Form und Farbe enthalten sein. Dabei erfolgt schon die Änderung der Farbe bei LaTeX nicht transparent und kann damit unerwünschte Effekte hervorrufen, wenn man \usekomafont an ungünstiger Stelle verwendet.

Nun neigen Anwender dazu, in die Schrifteinstellungen auch ganz andere, teilweise sehr kritische Dinge zu packen, beispielsweise ein \MakeUppercase ganz am Ende der Einstellung. Bei der internen Verwendung der Schrifteinstellungen wurde daher möglichst so vorgegangen, dass viele dieser eigentlich verbotenen Einstellungen trotzdem keinen Schaden anrichten und es meist sogar funktioniert, wenn der letzte Befehl in der Schrifteinstellung ein Argument erwartet, also beispielsweise \textbf anstelle von \bfseries verwendet wird. Eine Garantie gibt es dafür jedoch nicht.

In Einzelfällen war es innerhalb von KOMA-Script notwendig, die Umschaltung wirklich auf Schrifteinstellungen zu beschränken. Dies erfolgt dann beispielsweise mit \usefontofkomafont statt \usekomafont (siehe Abschnitt 3.6, Seite 66).

Die Anweisung \usefontofkomafont und ihre Geschwister haben allerdings ihre Grenzen. Deshalb darf die vermeintliche Schrifteinstellung eines Elements keinesfalls ein voll expandierbares Argument erwarten. Genau das ist aber beispielsweise bei \MakeUppercase der Fall. Daher verwaltet KOMA-Script eine interne Liste von Makros, die innerhalb von \usefontofkomafont und ihren Geschwistern zu \relax werden sollen. In der Voreinstellung ist das seit KOMA-Script 3.24 nur noch \normalcolor.

v3.17

v3.24

Es ist zu beachten, dass das angegebene *Makro* wirklich stur auf \relax gesetzt wird. Irgendwelche Argumente innerhalb der Schrifteinstellung werden also gegebenenfalls lokal ausgeführt. Daher dürfen Anweisungen wie \setlength keinesfalls zu dieser Liste hinzugefügt werden. Für alle Fehler, die durch die Verwendung von \addtokomafontrelaxlist entstehen, ist der Anwender selbst verantwortlich. Außerdem sollte diese Möglichkeit nicht als Legitimation dafür missverstanden werden, den Schrifteinstellungen alle möglichen Anweisungen hinzuzufügen!

v3.24

Für Befehle, deren erstes Argument noch ohne zusätzliche Gruppe ausgeführt werden soll, gibt es \addtokomafontonearglist. Das angegebene *Makro* wird dabei auf \@firstofone gesetzt. In der Voreinstellung wird dies für \MakeUppercase und \MakeLowercase verwendet.

v3.19

Soll hingegen ein *Makro* innerhalb von \usefontofkomafont und ihren Geschwistern zusammen mit seinem ersten Argument ignoriert werden, so ist stattdessen \addtokomafontgobblelist zu verwenden. Ein Beispiel dafür ist die Anweisung \color, die einschließlich des Namens der Farbe ignoriert werden muss und deshalb bereits in der Voreinstellung Teil dieser Liste ist.

Es ist zu beachten, dass sich die hier genannten Voreinstellungen in zukünftigen Versionen ändern können. Wenn Sie bestimmte Befehle in einer der Listen zwingend benötigen, sollten Sie diese also selbst explizit hinzufügen.

\IfExistskomafont{*Element*}{*Dann-Code*}{*Sonst-Code*}
\IfIsAliaskomafont{*Element*}{*Dann-Code*}{*Sonst-Code*}

v3.15 Da die Schrift mancher Elemente erst ab bestimmten Versionen von KOMA-Script geändert werden kann, ist es manchmal sinnvoll, vorher testen zu können, ob ein *Element* mit dieser Möglichkeit überhaupt existiert. Die Anweisung \IfExistskomafont führt den *Dann-Code* genau dann aus, wenn das *Element* über \newkomafont oder \aliaskomafont definiert wurde und daher auch mit \setkomafont oder \addtokomafont geändert und mit den \use...komafont-Anweisungen abgefragt werden kann. Anderenfalls wird der *Sonst-Code* ausgeführt.

v3.25 Im Unterschied dazu führt \IfIsAliaskomafont den *Dann-Code* nur aus, wenn *Element* über \aliaskomafont als Alias für ein anderes Element definiert wurde. Sowohl für nicht definierte Elemente als auch für mit \newkomafont definierte Elemente wird hingegen der *Sonst-Code* ausgeführt.

## 18.6. Absatzmarkierung

Nicht nur, aber insbesondere aufgrund der erweiterten Möglichkeiten der Absatzmarkierung bei den KOMA-Script-Klassen sollte auf die direkte Änderung der Standardlängen \parskip, \parindent und \parfillskip weitgehend verzichtet werden.

\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.

v3.17 Eine Neuberechnung oder Reaktivierung der Einstellungen für den Satzspiegel und die Ränder (siehe Kapitel 2) führt seit KOMA-Script 3.17 übrigens immer auch zu einer Neueinstellung der via \setparsizes gesetzten Werte, falls die Werte nicht zwischenzeitlich geändert wurden. Dies sollte ein Grund mehr sein, nicht an KOMA-Script vorbei die Einstellungen zu ändern.

Die Neuberechnung wird bei einer Kompatibilitätseinstellung zu einer früheren Version (siehe Abschnitt 3.2, Seite 56, Option `version`) deaktiviert.

## 18.7. Zähler

Zähler sind bei LaTeX eine durchaus komplexere Angelegenheit, als dies auf den ersten Blick erscheinen mag. Das gilt insbesondere, wenn die Referenzierung von Zählern über `\refstepcounter`, `\label` und `\ref` mit betrachtet wird.

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

## 18.8. Gliederung

Die KOMA-Script-Klassen bieten weitreichende Möglichkeiten, um Einfluss auf die Gliederungsebenen und die zugehörigen Überschriften zu nehmen. Selbst die Definition neuer Ebenen ist damit möglich.

```
\DeclareSectionCommand[Einstellungen]{Name}
\DeclareNewSectionCommand[Einstellungen]{Name}
\RedeclareSectionCommand[Einstellungen]{Name}
\ProvideSectionCommand[Einstellungen]{Name}
```

v3.15 Diese Anweisungen dienen dazu, einen neuen Gliederungsbefehl `\Name` zu definieren beziehungsweise einen vorhandenen Gliederungsbefehl `\Name` zu verändern. Dazu werden über das optionale Argument *Einstellungen* vorgenommen. Die *Einstellungen* sind dabei eine durch Komma separierte Liste von *Schlüssel*=*Wert*-Zuweisungen. Neben den vom Stil der Überschrift unabhängigen Eigenschaften, die Tabelle 18.1, Seite 489 zu entnehmen sind, gibt es auch Eigenschaften, die vom jeweiligen Stil abhängig sind. Derzeit stehen die folgenden Stile zur Verfügung:

v3.18 `part` ist der Stil für Teileüberschriften. Dieser Stil wird in der Voreinstellung für `\part` und indirekt für `\addpart` verwendet. Neue Überschriften in diesem Stil können definiert werden, haben dann aber nicht automatisch auch eine `\add...`-Variante. Für die Konfiguration der vorhandenen oder neuer Überschriften stehen zusätzlich die Eigenschaften aus Tabelle 18.2, Seite 490 zur Verfügung. Die Anweisung `\addpart` wird ebenso wie die Sternformen automatisch zusammen mit `\part` umkonfiguriert und kann nicht unabhängig davon verändert werden.

v3.18 `chapter` ist der Stil für Kapitelüberschriften. Dieser Stil wird in der Voreinstellung für `\chapter` und indirekt für `\addchap` verwendet. Neue Überschriften in diesem Stil können definiert werden, haben dann aber nicht automatisch auch eine `\add...`-Variante. Für die Konfiguration der vorhandenen oder neuer Überschriften stehen zusätzlich die

Eigenschaften aus Tabelle 18.3, Seite 491 zur Verfügung. Die Anweisung `\addchap` wird ebenso wie die Sternformen automatisch zusammen mit `\chapter` umkonfiguriert und kann nicht unabhängig davon verändert werden. Es ist zu beachten, dass dieser Stil von scrartcl nicht bereitgestellt wird.

scrbook, scrreprt

`section` ist der Stil für Abschnittsüberschriften und tieferer Ebenen. Dieser Stil wird derzeit sowohl für `\section`, `\subsection`, `\subsubsection` als auch `\paragraph` und `\subparagraph` verwendet. Neue Überschriften in diesem Stil können definiert werden. Für die Konfiguration der vorhandenen oder neuer Überschriften stehen zusätzlich die Eigenschaften aus Tabelle 18.4, Seite 492 zur Verfügung. Bei der Neudefinition sind die *Schlüssel* `style`, `afterskip`, `beforeskip` und `level` zwingend. Die *Schlüssel* `afterindent`, `font`, `indent` und `runin` sind empfohlen. `tocindent` und `tocnumwidth` können abhängig vom Namen der Gliederungsebene ebenfalls zwingend sein. Das gilt auch, falls ein Befehl, der bisher kein Gliederungsbefehl war, mit `\RedeclareSectionCommand` zu einem Gliederungsbefehl umdefiniert wird. Die Anweisung `\addsec` wird ebenso wie die Sternformen zusammen mit `\section` umkonfiguriert und kann nicht unabhängig davon verändert werden.

v3.24

v3.26

Bei der Definition eines Gliederungsbefehls wird ein gleichnamiges Element angelegt, falls es noch nicht existiert. Bei `chapter` und `part` werden ebenso Elemente für die Präfixzeile erzeugt. Die Schrifteinstellung der Elemente kann mit `\setkomafont` und `\addtokomafont` (siehe Abschnitt 3.6, Seite 60) geändert werden.

`\DeclareNewSectionCommand` dient der Definition eines neuen Gliederungsbefehls. Ist derselbe *Name* von TEX bereits anderweitig belegt, so wird ein Fehler ausgegeben und es findet keine Umdefinierung statt.

`\ProvideSectionCommand` verhält sich ähnlich, gibt aber keine Fehlermeldung aus, wenn der Gliederungsbefehl bereits existiert. Es findet dann nur keine Umdefinierung statt.

`\RedeclareSectionCommand` kann hingegen nur verwendet werden, um eine existierende Anweisung zu einem Gliederungsbefehl mit den angegebenen *Eigenschaften* zu ändern. Dabei wird nicht überprüft, ob `\`*Name* bereits zuvor ein Gliederungsbefehl war. Es muss nur ein von TEX bereits belegter *Name* sein.

Bei `\DeclareSectionCommand` findet keinerlei Überprüfung statt, ob *Name* von TEX bereits anderweitig belegt ist. Stattdessen wird der Gliederungsbefehl `\`*Name* unbedingt entsprechend der angegebenen *Eigenschaften* definiert.

Zu jeder Gliederungsanweisung gehört außerdem ein Zähler: *Name*, der bei Bedarf von allen vier Befehlen mit `\newcounter` neu angelegt wird. Dasselbe gilt für die Ausgabe des Zählers: `\the`*Name*, die Formatierung des Zählers: `\`*Name*`format`, die Anweisung zur Erstellung eines Kolumnentitels: `\`*Name*`mark`, die dabei verwendete Formatierung des Zählers: `\`*Name*`markformat`, die oben erwähnten Elemente: *Name* und gegebenenfalls *Name*`prefix`, die numerische Gliederungsebene: `\`*Name*`numdepth`. Die Anweisung `\`*Name*`mark` wird gegebenenfalls so vordefiniert, dass kein Kolumnentitel erzeugt wird. Die Ausgabe des Zählers, `\the`*Name*,

Tabelle 18.1.: Mögliche vom Stil der Überschrift unabhängige *Schlüssel* und *Werte* für die *Eigenschaften* bei der Konfiguration von Gliederungsbefehlen

| | *Schlüssel* | *Wert* | Bedeutung |
|---|---|---|---|
| | counterwithin | *Zählername* | Der zur Gliederungsebene gehörende Zähler soll vom als Wert angegebenen Zähler abhängig sein. Wird *Zählername* über \stepcounter oder \refstepcounter erhöht, so wird der zur Gliederungsebene gehörende Zähler auf 0 zurückgesetzt. Darüber hinaus wird \the*Zählername*, gefolgt von einem Punkt in der Ausgabe des zur Gliederungsebene gehörenden Zählers vorangestellt. |
| v3.19 | counterwithout | *Zählername* | Hebt eine früher vorgenommene counterwithin-Einstellung auf und ist daher nur bei Änderung vorhandener Gliederungsbefehle sinnvoll. |
| | expandtopt | *Schalter* | Ist dieser Schalter aktiv, so werden alle in den *Einstellungen* nachfolgend angegebenen Werte für Längen vollständig expandiert, ausgewertet und in pt umgerechnet gespeichert. Ist der Schalter nicht aktiv, so werden alle nachfolgend angegebenen Werte für Längen nur testweise expandiert und ausgewertet, aber lediglich expandiert gespeichert. Es werden die Werte für einfache Schalter aus Tabelle 2.5, Seite 41 verstanden. |
| | level | *Ganzzahl* | Numerischer Wert der Gliederungsebene (siehe Zähler secnumdepth, Abschnitt 3.16, Seite 121); der Wert sollte eindeutig sein und ist für neue Ebenen zwingend. |
| | style | *Name* | Legt den Stil der Überschrift fest und ist für neue Ebenen zwingend. |
| v3.20 | tocstyle | *Name* | Legt den Stil des zur Überschrift gehörenden Verzeichniseintrags fest. Es können alle bisher definierten Verzeichniseintragsstile (siehe Abschnitt 13.3) verwendet werden. Ein leerer *Name* verhindert die Umdefinierung des Befehls \l@... für die Verzeichniseinträge. |
| v3.20 | toc*Option* | *Wert* | Weitere Optionen in Abhängigkeit vom via tocstyle gewählten Verzeichniseintragsstil. Siehe dazu Abschnitt 13.3 ab Seite 378. Für die von tocbasic vordefinierten Verzeichniseintragsstile finden sich die als *Option* verwendbaren Attribute in Tabelle 13.1, ab Seite 383. |

Tabelle 18.2.: Zusätzliche *Schlüssel* und *Werte* für die *Eigenschaften* bei der Konfiguration von Gliederungsbefehlen des Stils `part`

| *Schlüssel* | *Wert* | Bedeutung |
|---|---|---|
| v3.26 `afterindent` | *Schalter* | Es wird bestimmt, ob auf die erste Zeile nach der Überschrift der aktuelle Absatzeinzug angewendet wird. Bei der Einstellung `bysign` bestimmt das Vorzeichen von `beforeskip` das Verhalten. Ein negativer Wert für `beforeskip` bewirkt dann, dass der Absatzeinzug entfällt. Mit den Werten für einfache Schalter (siehe Tabelle 2.5, Seite 41) kann die Anwendung des aktuellen Absatzeinzugs explizit aktiviert oder deaktiviert werden. Aus Kompatibilitätsgründen ist die Voreinstellung für scrartcl `false`, für scrbook und scrreprt `true`. |
| `afterskip` | *Länge* | Der Betrag gibt den vertikalen Abstand nach der Überschrift an. |
| `beforeskip` | *Länge* | Gibt den vertikalen Abstand vor der Überschrift an. Ist `afterindent=bysign`, so wird für den Abstand der Betrag von *Länge* verwendet. Es wird also trotzdem ein positiver Abstand eingefügt. Negative Werte bedeuten in diesem Fall, dass ein Absatzeinzug nach der Überschrift entfällt. |
| `font` | *Befehle* | Die Schrifteinstellungen, die zusätzlich zum Element `disposition` bei der Ausgabe des Textes der Überschrift verwendet werden sollen. Hier sind alle *Befehle* erlaubt, die auch über `\setkomafont` und `\addtokomafont` für das Element des Gliederungsbefehls erlaubt sind. |
| `innerskip` | *Länge* | Der vertikale Abstand zwischen Präfixzeile und Text der Überschrift bei scrbook und scrreprt. |
| `pagestyle` | *Seitenstil* | Der Name des Seitenstils, der für die Seite mit der Überschrift verwendet werden soll. Es findet keine Überprüfung statt, ob der angegebene *Seitenstil* gültig ist. Fehlerhafte Angaben führen daher zu Fehlermeldungen bei Verwendung des Gliederungsbefehls. Diese Möglichkeit existiert nur bei scrbook und scrreprt. |
| `prefixfont` | *Befehle* | Die Schrifteinstellungen, die zusätzlich zum Element `disposition` ab der Ausgabe einer Präfixzeile oder Nummer in der Überschrift verwendet werden sollen. Hier sind alle *Befehle* erlaubt, die auch über `\setkomafont` und `\addtokomafont` für das Element der Präfixzeile des Gliederungsbefehls erlaubt sind. |

Tabelle 18.3.: Zusätzliche *Schlüssel* und *Werte* für die *Eigenschaften* bei der Konfiguration von Gliederungsbefehlen des Stils `chapter`

| | *Schlüssel* | *Wert* | Bedeutung |
|---|---|---|---|
| v3.26 | `afterindent` | *Schalter* | Es wird bestimmt, ob auf die erste Zeile nach der Überschrift der aktuelle Absatzeinzug angewendet wird. Bei der Voreinstellung `bysign` bestimmt das Vorzeichen von `beforeskip` das Verhalten. Ein negativer Wert für `beforeskip` bewirkt dann, dass der Absatzeinzug entfällt. Mit den Werten für einfache Schalter (siehe Tabelle 2.5, Seite 41) kann die Anwendung des aktuellen Absatzeinzugs explizit aktiviert oder deaktiviert werden. |
| v3.26 | `afterskip` | *Länge* | Gibt den vertikalen Abstand nach der Überschrift an. |
| | `beforeskip` | *Länge* | Gibt den vertikalen Abstand vor der Überschrift an. Ist `afterindent=bysign`, so wird für den Abstand der Betrag von *Länge* verwendet. Es wird also trotzdem ein positiver Abstand eingefügt. Negative Werte bedeuten in diesem Fall, dass ein Absatzeinzug nach der Überschrift entfällt. |
| | `font` | *Befehle* | Die Schrifteinstellungen, die zusätzlich zum Element `disposition` bei der Ausgabe der Überschrift verwendet werden sollen. Hier sind alle *Befehle* erlaubt, die auch über `\setkomafont` und `\addtokomafont` für das Element des Gliederungsbefehls erlaubt sind. |
| | `innerskip` | *Länge* | Der vertikale Abstand zwischen Präfixzeile und Text der Überschrift, falls eine Präfixzeile verwendet wird. |
| | `pagestyle` | *Seitenstil* | Der Name des Seitenstils, der für die Seite mit der Überschrift verwendet werden soll. Es findet keine Überprüfung statt, ob der angegebene *Seitenstil* gültig ist. Fehlerhafte Angaben führen daher zu Fehlermeldungen bei Verwendung des Gliederungsbefehls. |
| | `prefixfont` | *Befehle* | Die Schrifteinstellungen, die zusätzlich zum Element `disposition` und dem Element des Gliederungsbefehls bei der Ausgabe einer Präfixzeile in der Überschrift verwendet werden sollen. Hier sind alle *Befehle* erlaubt, die auch über `\setkomafont` und `\addtokomafont`für das Element der Präfixzeile des Gliederungsbefehls erlaubt sind. |

Tabelle 18.4.: Zusätzliche *Schlüssel* und *Werte* für die *Eigenschaften* bei der Konfiguration von Gliederungsbefehlen des Stils `section`

| *Schlüssel* | *Wert* | Bedeutung |
|---|---|---|
| v3.26 `afterindent` | *Schalter* | Es wird bestimmt, ob auf die erste Zeile nach einer frei stehenden Überschrift (siehe `runin`) der aktuelle Absatzeinzug angewendet wird. Bei der Voreinstellung `bysign` bestimmt das Vorzeichen von `beforeskip` das Verhalten. Ein negativer Wert für `beforeskip` bewirkt dann, dass der Absatzeinzug entfällt. Mit den Werten für einfache Schalter (siehe Tabelle 2.5, Seite 41) kann die Anwendung des aktuellen Absatzeinzugs explizit aktiviert oder deaktiviert werden. |
| `afterskip` | *Länge* | Im Fall einer Spitzmarke (siehe `runin`) ist der Betrag der *Länge* der horizontale Abstand nach der Überschrift. Es wird in diesem Fall also immer ein positiver Abstand eingefügt. Im Fall einer frei stehenden Überschrift ist *Länge* der vertikale Abstand nach der Überschrift. Ist `runin=bysign`, so führt ein positiver Wert zu einer frei stehenden Überschrift, während ein negativer Wert oder Null zu einer Spitzmarke führt. |
| `beforeskip` | *Länge* | Gibt den vertikalen Abstand vor der Überschrift an. Ist `afterindent=bysign`, so wird für den Abstand der Betrag von *Länge* verwendet. Es wird also trotzdem ein positiver Abstand eingefügt. Negative Werte bedeuten in diesem Fall, dass ein Absatzeinzug nach der Überschrift entfällt. |
| `font` | *Befehle* | Die Schrifteinstellungen, die zusätzlich zum Element `disposition` bei der Ausgabe der Überschrift verwendet werden sollen. Hier sind alle *Befehle* erlaubt, die auch über `\setkomafont` und `\addtokomafont` für das Element des Gliederungsbefehls erlaubt sind. |
| `indent` | *Länge* | Einzug vom linken Rand vor der Ausgabe der Nummer und des Textes der Überschrift. |
| v3.26 `runin` | *Schalter* | Es wird bestimmt, ob die Überschrift als Spitzmarke (am Zeilenanfang) oder frei stehend gesetzt wird. Bei der Voreinstellung `bysign` bestimmt das Vorzeichen von `afterskip` das Verhalten. Ein positiver Wert für `afterskip` bewirkt dann eine frei stehende Überschrift. Darüber hinaus kann mit den Werten für einfache Schalter (siehe Tabelle 2.5, Seite 41) eine Spitzmarke explizit aktiviert oder deaktiviert werden. |

wird als arabische Zahl vordefiniert. Wird über den *Schlüssel* `counterwithin` der Zähler als von einem anderen Zähler abhängig definiert, so wird in der Ausgabe dieser andere Zähler mit einem Punkt getrennt vorangestellt.

v3.20 Neben dem Gliederungsbefehl selbst wird auch ein Befehl für einen Eintrag ins Inhaltsverzeichnis definiert. Dafür wird auf das Paket tocbasic zurückgegriffen. Der Stil des Verzeichniseintrags wird über die Eigenschaft `tocstyle` festgelegt. Wird hier mit `tocstyle=` oder `tocstyle={}` ein leerer *Name* angegeben, so erfolgt keine Umdefinierung des Befehls für den Verzeichniseintrag. Das ist beispielsweise dann wichtig, wenn Sie ein zusätzliches Paket zur Modifikation des Inhaltsverzeichnisses verwenden. Fehlt die Eigenschaft `tocstyle`, so wird bei der Umdefinierung der bisherige Stil erneut verwendet.

v3.20 Unterschiedliche Stile für Verzeichniseinträge haben unterschiedliche zusätzliche Eigenschaften. Diese können, mit dem Präfix `toc` versehen, direkt mit angegeben werden. So kann beispielsweise die Ebene des Verzeichniseintrags, die bei allen von den KOMA-Script-Klassen und tocbasic definierten Stilen als Eigenschaft `level` bekannt ist, mit `toclevel` gesetzt werden, der Einzug des Eintrags, `indent`, über `tocindent` und die für die Nummer reservierte Breite, `numwidth`, mit `tocnumwidth`. Für weitere Eigenschaften der Verzeichniseinträge siehe Abschnitt 13.3 ab Seite 378.

**Beispiel:** Aus unerfindlichen Gründen sollen die Überschriften von `\paragraph` nicht mehr als Spitzmarken, sondern als Überschriften ähnlich `\subsubsection` umdefiniert werden. Dabei soll über der Überschrift ein kleiner Abstand von 10 pt und unter der Überschrift kein zusätzlicher Abstand eingefügt werden. Das wäre bereits mit

```
\RedeclareSectionCommand[%
  beforeskip=-10pt,%
  afterskip=1sp%
]{paragraph}
```

möglich. Durch den negativen Wert bei `beforeskip` wird der vertikale Abstand über der Überschrift erzeugt und gleichzeitig der Einzug des ersten Abschnitts nach der Überschrift abgeschaltet. Obwohl eigentlich nach der Überschrift kein vertikaler Abstand gewünscht wird, wurde als Wert hier 1 sp angegeben. Der Grund ist einfach: Einen Wert von 0 pt betrachtet TEX nicht als positiven Wert und `\RedeclareSectionCommand` erzeugt damit unter Berücksichtigung der Voreinstellung `runin=bysign` eine Überschrift in Form einer Spitzmarke. Der kleinste positive Wert ist 1 sp. Natürlich könnte man stattdessen auch explizit die Verwendung von Spitzmarken aktivieren und dann wirklich einen Wert von 0 pt einstellen:

```
\RedeclareSectionCommand[%
  beforeskip=-10pt,%
  runin=true,%
  afterskip=0pt%
]{paragraph}
```

In der Regel ist es für den vertikalen Ausgleich (siehe \flushbottom, Abschnitt 3.4, Seite 58) besser, wenn man die Abstände mit etwas Spielraum, dem sogenannten Leim, versieht:

```
\RedeclareSectionCommand[%
  beforeskip=-10pt plus -2pt minus -1pt,%
  runin=true,%
  afterskip=0pt%
]{paragraph}
```

Dabei ist zu beachten, dass natürlich auch der Leim bei der Anwendung als vertikaler Abstand das Vorzeichen wechselt, also bei beforeskip im Beispiel negativ angegeben wird. Auch diese Überlegung kann man sich jedoch sparen, indem man den Erstzeileneinzug nach der Überschrift nicht über das Vorzeichen von beforeskip steuert, sondern über afterindent:

```
\RedeclareSectionCommand[%
  afterindent=false,%
  beforeskip=10pt plus 2pt minus 1pt,%
  runin=true,%
  afterskip=0pt%
]{paragraph}
```

Dass im Beispiel nur die Schlüssel verwendet werden mussten, deren Einstellung geändert werden soll, liegt daran, dass seit KOMA-Script 3.15 \paragraph intern bereits mit \DeclareSectionCommand definiert wird. Alle anderen Einstellungen aus der Originaldefinition bleiben so enthalten. Bei scrartcl entspräche dies der Definition:

scrartcl

```
\DeclareSectionCommand[%
  level=4,
  indent=0pt,
  afterindent=bysign,
  runin=bysign,
  beforeskip=3.25ex plus 1ex minus .2ex,
  afterskip=-1em,
  font={},
  tocindent=7em,
  tocnumwidth=4.1em,
  counterwithin=subsubsection
]{paragraph}
```

scrbook, scrreprt

Die Werte für scrreprt und scrbook weichen teilweise ab. Für \chapter sind einige Einstellungen für die Überschriften von Option headings (siehe Abschnitt 3.16, Seite 102) abhängig. Diese abhängigen Einstellungen sind in Tabelle 18.5 zu finden. Eine Übersicht über alle Voreinstellungen bietet Tabelle 18.6. Es ist zu beachten, dass dabei 1ex und \baselineskip von der voreingestellten Größe der Überschrift beziehungsweise des Inhaltsverzeichniseintrags ab-

Tabelle 18.5.: Voreinstellungen für die Kapitelüberschriften von scrbook und scrreprt in Abhängigkeit von Option headings

**Mit headings=big:**

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | 1.725\baselineskip plus .115\baselineskip minus .192\baselineskip |
| beforeskip | -3.3\baselineskip-\parskip |
| font | \huge |

**Mit headings=normal:**

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | 1.5\baselineskip plus .1\baselineskip minus .167\baselineskip |
| beforeskip | -3\baselineskip-\parskip |
| font | \LARGE |

**Mit headings=small:**

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | 1.35\baselineskip plus .09\baselineskip minus .15\baselineskip |
| beforeskip | -2.8\baselineskip-\parskip |
| font | \Large |

v3.20 hängig sind. Weitere Voreinstellungen für die Stile der Verzeichniseinträge sind Abschnitt 13.3 ab Seite 378 zu entnehmen.

Das in den Einstellungen für \subparagraph verwendete interne Makro \scr@parindent ist übrigens, der per Option parskip oder Befehl \setparsizes eingestellte Absatzeinzug.

Tabelle 18.6.: Voreinstellungen für die Formatierung der Überschriften von scrbook und scrreprt

\part:

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | 0pt plus 1fil |
| beforeskip | 0pt plus 1fil + \baselineskip |
| font | siehe part, Tabelle 3.15, Seite 111 |
| innerskip | 20pt |
| level | -1 |
| prefixfont | siehe partnumber, Tabelle 3.15, Seite 111 |
| tocindent | 0pt |
| toclevel | -1 |
| tocnumwidth | 2em |
| tocstyle | part |

\chapter:

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | siehe Tabelle 18.5 |
| beforeskip | siehe Tabelle 18.5 |
| font | siehe chapter, Tabelle 3.15, Seite 111 |
| innerskip | 0.5\baselineskip |
| level | 0 |
| prefixfont | siehe chapterprefix, Tabelle 3.15, Seite 111 |
| tocindent | 0pt |
| toclevel | 0 |
| tocnumwidth | 1.5em |
| tocstyle | chapter |

...

Tabelle 18.6.: Voreinstellungen für die Formatierung der Überschriften von scrbook und scrreprt *(Fortsetzung)*

`\section`:

| Einstellung | voreingestellter Wert |
|---|---|
| `afterskip` | `2.3ex plus .2ex` |
| `beforeskip` | `-3.5ex plus -1ex minus -.2ex` |
| `font` | siehe `section`, Tabelle 3.15, Seite 111 |
| `indent` | `0pt` |
| `level` | 1 |
| `tocindent` | `1.5em` |
| `toclevel` | 1 |
| `tocnumwidth` | `2.3em` |
| `tocstyle` | `section` |

`\subsection`:

| Einstellung | voreingestellter Wert |
|---|---|
| `afterskip` | `1.5ex plus .2ex` |
| `beforeskip` | `-3.25ex plus -1ex minus -.2ex` |
| `font` | siehe `subsection`, Tabelle 3.15, Seite 111 |
| `indent` | `0pt` |
| `level` | 2 |
| `tocindent` | `3.8em` |
| `toclevel` | 2 |
| `tocnumwidth` | `3.2em` |
| `tocstyle` | `section` |

...

Tabelle 18.6.: Voreinstellungen für die Formatierung der Überschriften von scrbook und scrreprt *(Fortsetzung)*

\subsubsection:

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | 1.5ex plus .2ex |
| beforeskip | -3.25ex plus -1ex minus -.2ex |
| font | siehe subsubsection, Tabelle 3.15, Seite 111 |
| indent | 0pt |
| level | 3 |
| tocindent | 7.0em |
| toclevel | 3 |
| tocnumwidth | 4.1em |
| tocstyle | section |

\paragraph:

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | -1em |
| beforeskip | 3.25ex plus 1ex minus .2ex |
| font | siehe paragraph, Tabelle 3.15, Seite 111 |
| indent | 0pt |
| level | 4 |
| tocindent | 10em |
| toclevel | 4 |
| tocnumwidth | 5em |
| tocstyle | section |

...

Tabelle 18.6.: Voreinstellungen für die Formatierung der Überschriften von scrbook und scrreprt *(Fortsetzung)*

\subparagraph:

| Einstellung | voreingestellter Wert |
|---|---|
| afterskip | -1em |
| beforeskip | 3.25ex plus 1ex minus .2ex |
| font | siehe subparagraph, Tabelle 3.15, Seite 111 |
| indent | \scr@parindent |
| level | 5 |
| tocindent | 12em |
| toclevel | 5 |
| tocnumwidth | 6em |
| tocstyle | section |

```
\DeclareSectionCommands[Einstellungen]{Namensliste}
\DeclareNewSectionCommands[Einstellungen]{Namensliste}
\RedeclareSectionCommands[Einstellungen]{Namensliste}
\ProvideSectionCommands[Einstellungen]{Namensliste}
```

v3.15 Diese Anweisungen können gleich eine ganze Reihe von Gliederungsbefehlen definieren oder ändern. Dabei ist *Namensliste* eine durch Komma separierte Liste von Namen der Gliederungsbefehle.

Die Befehle unterscheiden sich in zwei weiteren Punkten von den zuvor erklärten Anweisungen zur Definition oder Änderung eines einzelnen Gliederungsbefehls. Zum einen wird im Fehlerfall, also wenn eine Anweisung bei \DeclareNewSectionCommands bereits zuvor existierte oder bei \RedeclareSectionCommands noch nicht existierte, die Definition dennoch vorgenommen. Ein entsprechender Fehler wird natürlich trotzdem gemeldet. Zum anderen gibt es eine weitere Einstellung: increaselevel=*Ganzzahl*. Damit ändert sich die Bedeutung von level und toclevel (siehe Tabelle 18.1, Seite 489) dahingehend, dass deren Werte lediglich als *Einstellungen* des ersten Gliederungsbefehls aus der *Namensliste* dienen. Für alle weiteren Gliederungsbefehle werden die Werte von level und toclevel um den Wert von increaselevel erhöht. Wurde die Einstellung increaselevel ohne Wertzuweisung verwendet, so wird der Wert 1 angenommen.

```
\IfSectionCommandStyleIs{Name}{Stil}{Dann-Code}{Sonst-Code}
```

v3.27 In seltenen Fällen ist es nützlich, testen zu können, ob ein Gliederungsbefehl einem bestimmten *Stil* angehört. Ist der mittels *Name* bestimmte Gliederungsbefehl aus KOMA-Script-Sicht derzeit mit dem angebenen *Stil* definiert, so wird der *Dann-Code* ausgeführt, anderenfalls

der *Sonst-Code*. Ist *Name* nicht definiert oder kein mit KOMA-Script-Mitteln definierter Gliederungsbefehl, so wird dies als Fehler gemeldet.

```
\chapterheadstartvskip
\chapterheadmidvskip
\chapterheadendvskip
\partheadstartvskip
\partheadmidvskip
\partheadendvskip
\partheademptypage
```

Diese Anweisungen werden innerhalb von Überschriften der zuvor erklärten Stile `chapter` und `part` und damit für die Definition der Überschriften \chapter, \part, \addchap, \addpart und deren Sternvarianten \chapter*, \part*, \addchap*, \addpart* 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. Bei Kapitelüberschriften mit eigener Nummernzeile (siehe Option chapterprefix in Abschnitt 3.16, Seite 101) wird zwischen der Nummernzeile und der eigentlichen Überschrift außerdem `\chapterheadmidvskip` ausgeführt.

v3.15

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 vertikalen Abstandes interpretiert. Ein solcher Seitenumbruch ist in der Voreinstellung sowohl bei scrbook als auch scrreprt in der Definition 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 der sieben Anweisungen sind seit KOMA-Script 3.15 von der Einstellung von Option headings (siehe Abschnitt 3.16, Seite 102) unabhängig. Die Original-Definitionen für die Kapitelüberschriften ab KOMA-Script 3.17 entsprechen:

v3.15

v3.17

```
\newcommand*{\chapterheadstartvskip}{\vspace{\@tempskipa}}
\newcommand*{\chapterheadmidvskip}{\par\nobreak
                                   \vskip\@tempskipa}
\newcommand*{\chapterheadendvskip}{\vskip\@tempskipa}
```

Diese werden auch bei jeder Verwendung von Option headings=big, headings=normal oder headings=small reaktiviert. Als Seiteneffekt haben diese Optionen also gegebenenfalls nicht nur Auswirkungen auf Kapitelüberschriften, sondern auf alle Überschriften im Stil `chapter`.

Der Stil `chapter` setzt die interne Länge `\@tempskipa` vor Aufruf von `\chapterheadstartvskip` automatisch auf den Wert, der sich aus der \DeclareSectionCommand-Einstellung `beforeskip` ergibt. Vor dem Aufruf von

\chapterheadendvskip geschieht entsprechendes mit dem Wert, der sich aus der Einstellung afterskip ergibt, und vor dem Aufruf von \chapterheadmidvskip mit dem Wert, der sich aus der Einstellung innerskip ergibt.

v3.17 Da die Voreinstellungen für die Abstände von \part nicht von Option headings abhängen, werden auch die zugehörigen Anweisungen von dieser Option nicht neu definiert. Ihre Original-Definitionen entsprechen bei scrbook und scrreprt:

```
\newcommand*{\partheadstartvskip}{%
  \null\vskip-\baselineskip\vskip\@tempskipa
}
\newcommand*{\partheadmidvskip}{%
  \par\nobreak\vskip\@tempskipa
}
\newcommand*{\partheadendvskip}{%
  \vskip\@tempskipa\newpage
}
```

und bei scrartcl:

```
\newcommand*{\partheadstartvskip}{%
  \addvspace{\@tempskipa}%
}
\newcommand*{\partheadmidvskip}{%
  \par\nobreak
}
\newcommand*{\partheadendvskip}{%
  \vskip\@tempskipa
}
```

Auch hier wird vom Stil part die interne Länge \@tempskipa vor der Verwendung der Befehle entsprechend der Einstellungen von \DeclareSectionCommand gesetzt.

v3.17 Wird eine der Anweisungen, die im Original \@tempskipa für den vertikalen Abstand verwendet, umdefiniert und sollen die Abstände weiterhin beispielsweise mit \RedeclareSectionCommand konfigurierbar sein, so sollte man in der neuen Definition ebenfalls \@tempskipa verwenden. Da die Abstände über, innerhalb und unter den Überschriften einfacher mit \RedeclareSectionCommand eingestellt werden können, wird generell nicht empfohlen, zu diesem Zweck stattdessen die hier beschriebenen Anweisungen umzudefinieren. Sie sollten für tiefgreifendere Änderungen reserviert bleiben, die nicht über \RedeclareSectionCommand zu erreichen sind. Auf [KDP] findet sich dazu ein Beispiel, bei dem durch Umdefinierung von \chapterheadstartvskip und \chapterheadendvskip Linien über und unter der Kapitelüberschrift gesetzt werden.

\partlineswithprefixformat{*Ebene*}{*Nummer*}{*Text*}

v3.25

Diese Anweisung wird von Überschriften des Stils `part` verwendet. Die beiden Argumente *Nummer* und *Text* sind dabei einschließlich Einstellung der Fonts für sich bereits fertig formatiert. Letztlich regelt die Anweisung also die Anordnung der beiden Teile der Überschrift. Bei nicht nummerierten Überschriften ist *Nummer* ein komplett leeres Argument, enthält also auch keine Formatierungsanweisungen.

Vordefiniert ist die Anweisung mit:

```
\newcommand{\partlineswithprefixformat}[3]{#2#3}
```

**Beispiel:** Sie wollen die Teile-Überschriften in eine hellblaue Box mit blauer Umrandung stellen, die nur etwa drei Viertel der Breite des Textbereichs einnimmt. Hierzu verwenden Sie

```
\documentclass{scrbook}
\usepackage{xcolor}
\renewcommand*{\partlineswithprefixformat}[3]{%
  \fcolorbox{blue}{blue!25}{%
    \parbox{.75\linewidth}{#2#3}%
  }%
}
\begin{document}
\part{Umrahmte Teile}
\end{document}
```

Allerdings fällt Ihnen auf, dass die Überschrift dabei nicht wie sonst für Teile üblich zentriert wird – weder die Box selbst noch der Text innerhalb der Box.

Die Ursache für die fehlende Zentrierung der Box liegt darin, dass aufgrund der Änderung das in Argument 3 versteckte Absatzende nur noch den Absatz innerhalb der Box beendet, aber nicht mehr den Absatz mit der `\parbox`. Also ergänzen Sie ein `\par` am Ende der Definition.

Die Ursache der fehlenden Zentrierung innerhalb der Box ist, dass in der `\parbox`-Anweisung die Ausrichtung von `\raggedpart` nicht automatisch gültig ist. Daher ergänzen Sie diese Anweisung innerhalb der Box.

Mit

```
\documentclass{scrbook}
\usepackage{xcolor}
\renewcommand*{\partlineswithprefixformat}[3]{%
  \fcolorbox{blue}{blue!25}{%
    \parbox{.75\linewidth}{\raggedpart #2#3}%
  }%
  \par
}
```

```
\begin{document}
\part{Umrahmte Teile}
\end{document}
```

erhalten Sie die gewünschte Formatierung.

Wie im Beispiel gezeigt, ist der Anwender bei der Umdefinierung der Anweisung für einige Dinge selbst verantwortlich. Dazu gehört neben dem Erhalt der voreingestellten Ausrichtung auch, dass innerhalb der Überschrift kein Seitenumbruch, beispielsweise an zusätzlich eingefügten Absätzen oder Abständen, erfolgen kann. Das gezeigte Beispiel ist diesbezüglich unproblematisch. Nicht nur, dass die beiden Boxen ohnehin keinen Seitenumbruch erlauben. KOMA-Script verändert außerdem `\interlinepenalty` als Teil von *Text* so, dass darin kein Seitenumbruch stattfinden darf. Gleichzeitig endet *Text* immer mit einem internen Absatz, `\@@par`.

Das Argument *Ebene* wird von `\partlineswithprefixformat` in der Voreinstellung nicht verwendet und ist auch im Beispiel nicht erforderlich. Erst wenn der Anwender mehrere Anweisungen im Stil `part` definiert und man innerhalb der Definition nach den Anweisungen unterscheiden will, kann dies über *Ebene* erfolgen. Dabei ist *Ebene* der vom Namen der Anweisung abgeleitete Name der Gliederungsebene. Bei `\part`, `\part*`, `\addpart` und `\addpart*` ist *Ebene* daher einheitlich `part`.

`\chapterlineswithprefixformat{`*Ebene*`}{`*Nummer*`}{`*Text*`}`
`\chapterlinesformat{`*Ebene*`}{`*Nummer*`}{`*Text*`}`

v3.19

Diese Anweisungen werden von Überschriften des Stils `chapter` in Abhängigkeit von Option `chapterprefix` (siehe Abschnitt 3.16, Seite 101) verwendet. Dabei gibt `\chapterlineswithprefixformat` die Ausgabe bei aktivierter Option vor. Bei deaktivierter Option bestimmt dagegen `\chapterlinesformat` die Ausgabe.

Die beiden Argumente *Nummer* und *Text* sind dabei einschließlich Einstellung der Fonts für sich bereits fertig formatiert. Letztlich regeln die Anweisungen also die Anordnung der beiden Teile der Überschrift. Bei nicht nummerierten Überschriften ist *Nummer* ein komplett leeres Argument, enthält also auch keine Formatierungsanweisungen.

Vordefiniert sind die beiden Anweisungen mit:

```
\newcommand{\chapterlinesformat}[3]{%
  \@hangfrom{#2}{#3}%
}
\newcommand{\chapterlineswithprefixformat}[3]{#2#3}
```

**Beispiel:** Sie wollen Überschriften im Kapitelstil gelb hinterlegen. Für Überschriften ohne Präfixzeile definieren Sie daher in der Präambel des Dokument:

```
\makeatletter
\renewcommand{\chapterlinesformat}[3]{%
```

```
    \colorbox{yellow}{%
      \parbox{\dimexpr\linewidth
                      -2\fboxrule-2\fboxsep}{%
        \@hangfrom{#2}#3%
      }%
    }%
  }
  \makeatother
```

und für Überschriften mit Präfixzeile:

```
  \renewcommand{\chapterlineswithprefixformat}[3]{%
    \colorbox{yellow}{%
      \parbox{\dimexpr\linewidth
                      -2\fboxrule-2\fboxsep}{%
        #2#3%
      }%
    }%
  }
```

Allerdings fällt Ihnen nach einiger Zeit auf, dass mit dieser Umdefinierung die Überschriften wieder im Blocksatz gesetzt werden. Das liegt daran, dass `\parbox` sein Argument so setzt. Um dies zu korrigieren, fügen Sie die Anweisung `\raggedchapter` (siehe Abschnitt 3.16, Seite 115), die automatisch bereits vor `\chapterlinesformat` und `\chapterlineswithprefixformat` aufgerufen wird, in die Definitionen ein:

```
  \makeatletter
  \renewcommand{\chapterlinesformat}[3]{%
    \colorbox{yellow}{%
      \parbox{\dimexpr\linewidth
                      -2\fboxrule-2\fboxsep}{%
        \raggedchapter
        \@hangfrom{#2}#3%
      }%
    }%
  }
  \makeatother
  \renewcommand{\chapterlineswithprefixformat}[3]{%
    \colorbox{yellow}{%
      \parbox{\dimexpr\linewidth
                      -2\fboxrule-2\fboxsep}{%
        \raggedchapter
        #2#3%
      }%
    }%
  }
```

Es sei daran erinnert, dass die Klammerung mit den Anweisungen `\makeatletter`

und \makeatother nur in der Dokumentpräambel zu verwenden ist. In einer eigenen Wrapper-Klasse oder einem Paket haben sie zu entfallen. Sie werden auch nur wegen \@hangfrom in der Definition von \chapterlinesformat benötigt.

Wie im Beispiel gezeigt, ist der Anwender bei der Umdefinierung der Anweisungen für einige Dinge selbst verantwortlich. Dazu gehört neben dem Erhalt der voreingestellten Ausrichtung auch, dass innerhalb der Überschrift kein Seitenumbruch, beispielsweise an zusätzlich eingefügten Absätzen oder Abständen, erfolgen kann. Das gezeigte Beispiel ist diesbezüglich unproblematisch. Nicht nur, dass die beiden Boxen ohnehin keinen Seitenumbruch erlauben. KOMA-Script verändert außerdem \interlinepenalty als Teil von *Text* so, dass darin kein Seitenumbruch stattfinden darf. Gleichzeitig endet *Text* immer mit einem internen Absatz, \@@par.

Die Anweisung \raggedchapter ist übrigens nicht Bestandteil von *Text*, weil anderenfalls beispielsweise die Verwendung von \MakeUppercase innerhalb der Umdefinierung der beiden Anweisungen erheblich erschwert wäre. Dennoch sei darauf hingewiesen, dass typografischen Regeln zufolge Versalsatz mit gängigen Fonts immer der Sperrung und des Ausgleichs bedarf. Die LaTeX-Anweisung \MakeUppercase leistet dies jedoch nicht.

Das erste Argument, *Ebene*, wird von den Anweisungen in der Voreinstellung nicht verwendet und ist auch im Beispiel nicht erforderlich. Erst wenn der Anwender mehrere Anweisungen im Stil chapter definiert und man innerhalb der Definition nach den Anweisungen unterscheiden will, kann dies über *Ebene* erfolgen. Dabei ist *Ebene* der vom Namen der Anweisung abgeleitete Name der Gliederungsebene. Bei \chapter, \chapter*, \addchap und \addchap* ist *Ebene* daher einheitlich chapter.

\sectionlinesformat{*Ebene*}{*Einzug*}{*Nummer*}{*Text*}
\sectioncatchphraseformat{*Ebene*}{*Einzug*}{*Nummer*}{*Text*}

v3.19

Diese Anweisungen werden von Überschriften des Stils section in Abhängigkeit davon verwendet, ob die jeweilige Überschrift eine Spitzmarke erzeugt oder eine frei stehende Überschrift. Frei stehende Überschriften werden dabei per \sectionlinesformat ausgegeben, während \sectioncatchphraseformat für Spitzmarken zuständig ist.

In beiden Fällen gibt *Einzug* den Wert eines horizontalen Einzugs der Überschrift gegenüber dem Textbereich an. Durch Angabe eines negativen Wertes soll es auch möglich sein, die Überschrift in den linken Rand zu rücken.

Die beiden Argumente *Nummer* und *Text* sind einschließlich Einstellung der Fonts für sich bereits fertig formatiert. Letztlich regeln die Anweisungen also die Anordnung der beiden Teile der Überschrift. Bei nicht nummerierten Überschriften ist *Nummer* ein komplett leeres Argument, enthält also auch keine Formatierungsanweisungen.

Vordefiniert sind die beiden Anweisungen als:

```
\newcommand{\sectionlinesformat}[4]{%
  \@hangfrom{\hskip #2#3}{#4}%
```

```
}
\newcommand{\sectioncatchphraseformat}[4]{\hskip #2#3#4}
```

Bei Umdefinierung einer der beiden Anweisungen ist wiederum der Anwender selbst dafür verantwortlich, Seitenumbrüche innerhalb der Ausgabe zu verhindern. KOMA-Script hilft hier lediglich mit entsprechend gesetztem `\interlinepenalty`.

**Beispiel:** Wie schon im Beispiel der Kapitelüberschriften sollen nun die frei stehenden Überschriften der Ebene `section` mit einer Farbe hinterlegt werden. Die frei stehenden Überschriften tieferer Ebenen sollen nicht verändert werden:

```
\makeatletter
\renewcommand{\sectionlinesformat}[4]{%
  \Ifstr{#1}{section}{%
    \hspace*{#2}%
    \colorbox{yellow}{%
      \parbox{\dimexpr\linewidth
                      -2\fboxrule-2\fboxsep-#2}{%
        \raggedsection
        \@hangfrom{#3}{#4}%
      }%
    }%
  }{%
    \@hangfrom{\hskip #2#3}{#4}%
  }%
}
\makeatother
```

Mit dem gezeigten Code wird im Falle eines Einzugs der Überschrift der Bereich des Einzugs nicht mit gefärbt. Wird hingegen die Überschrift in den linken Rand gestellt, so wird dieser Bereich des Randes ebenfalls farbig hinterlegt. Durch Verschiebung der `\hspace*`-Anweisung in die `\colorbox` kann dieses Verhalten verändert werden.

`\makeatletter` und `\makeatother` werden in der Dokumentpräambel erneut wegen `\@hangfrom` benötigt.

Das erste Argument, *Ebene*, wird von den Anweisungen in der Voreinstellung nicht verwendet. Wie das Beispiel zeigt, kann es aber sehr gut dazu verwendet werden, nach unterschiedlichen Gliederungsebenen im gemeinsamen Stil `section` zu unterscheiden.

```
\ExecuteDoHook{heading/preinit/Name}
\ExecuteDoHook{heading/postinit/Name}
\ExecuteDoHook{heading/branch/star/Name}
\ExecuteDoHook{heading/branch/nostar/Name}
\ExecuteDoHook{heading/begingroup/Name}
\ExecuteDoHook{heading/endgroup/Name}
```

v3.27 Abgesehen von den übrigen Einstellmöglichkeiten bieten sämtliche mit \DeclareSectionCommand, \DeclareNewSectionCommand und \ProvideSectionCommand definierten und ebenso auch die mit \RedeclareSectionCommand umdefinierten Gliederungsanweisungen eine ganze Reihe an Haken, die per \AddtoDoHook manipuliert werden können. Zur Funktionsweise dieser *do-hook* genannten Haken sei auf Abschnitt 10.8 ab Seite 349 verwiesen. Der letzte Teil des Spezifikators ist dabei der *Name* der Gliederungsanweisung, wie er auch bei den oben genannten Befehlen als letztes Argument anzugeben ist.

Wichtig ist, dass keiner dieser Haken für Anweisungen verwendet werden sollte, die Auswirkungen auf den Seitenumbruch oder die Positionierung der Überschriften haben. Daher sollten nur wirklich erfahrene Anwender Gebrauch von diesen Haken machen. Im Zweifelsfall ist außerdem der Quellcode der Klasse bezüglich des genauen Ausführungszeitpunktes des jeweiligen Hakens zu konsultieren. Diese Haken dienen als letzter Notanker vor einer tatsächlichen Umdefinierung von Gliederungsbefehlen außerhalb der von KOMA-Script gebotenen Mittel.

Der Haken `heading/preinit/`*Name* wird unmittelbar vor der Initialisierung der Anweisungen ausgeführt. Zu diesem Zeitpunkt wurden noch keine Einstellungen vorgenommen. Selbst der Absatz, der einer Überschrift vorausgeht, ist noch nicht zwingend beendet.

Der Haken `heading/postinit/`*Name* wird etwas später ausgeführt. Zu diesem Zeitpunkt wurden einige Einstellungen bereits vorgenommen und auch ein vorausgehender Absatz wurde bereits beendet.

Der Haken `heading/branch/nostar/`*Name* wird ausgeführt, sobald feststeht, dass keine Sternform der Gliederungsanweisung vorliegt. Für die Sternform wird hingegen `heading/branch/star/`*Name* ausgeführt. Zu diesem Zeitpunkt sind gegebenenfalls auch bereits vertikale Abstände über der jeweiligen Überschrift eingefügt.

Der Haken `heading/begingroup/`*Name* wird am Anfang der Gruppe ausgeführt, in der schließlich die Überschrift gesetzt wird. Das ist also die letzte Eingriffsmöglichkeit vor der Ausgabe der Überschrift.

Entsprechend wird `heading/endgroup/`*Name* vor dem Ende der Gruppe ausgeführt, in der die Überschrift gesetzt wurde. Derzeit ist das auch der letzte Haken innerhalb einer Gliederungsanweisung.

Zu beachten ist, dass \minisec keine echte Gliederungsanweisung ist und die genannten Haken daher auf diese Anweisung keine Anwendung finden.

\IfUseNumber{*Dann-Code*}{*Sonst-Code*}

v3.27

Streng genommen handelt es sich hier um eine interne Anweisung. Sie ist nur innerhalb von Überschriften zwischen den beiden Haken .../begingroup/... und .../endgroup/... spezifiziert. In diesem Fall wird der *Dann-Code* ausgeführt, wenn die aktuelle Überschrift gemäß Einstellung von secnumdepth und aufgrund der Nichtverwendung einer Sternform nummeriert werden soll. Soll die Überschrift aufgrund der Verwendung einer Sternform oder der Einstellung von secnumdepth nicht nummeriert werden, so wird der *Sonst-Code* ausgeführt. Bei Überschriften im Stil chapter wird für die Entscheidung außerdem berücksichtigt, ob die Überschrift im Hauptteil gesetzt wird.

Wird die Anweisung außerhalb von Überschriften verwendet, so ist ihr Ergebnis nicht spezifiziert. In der Regel gibt sie dann eine Fehlermeldung aus und führt weder den *Dann-Code* noch den *Sonst-Code* aus.

\SecDef{*Sternanweisung*}{*Normalanweisung*}
\scr@startsection{*Name*}{*Ebene*}{*Einzug*}{*Abstand davor*}{*Abstand danach*}
{*Stilanweisungen*}[*Kurzform*]{*Überschrift*}
\scr@startsection{*Name*}{*Ebene*}{*Einzug*}{*Abstand davor*}{*Abstand danach*}
{*Stilanweisungen*}*{*Überschrift*}

v3.15

Wie bereits in Abschnitt 3.16 bei der Beschreibung zu den Gliederungsbefehlen ab Seite 107 erklärt, verfügt KOMA-Script bezüglich des optionalen Arguments der Gliederungsbefehle über erweiterte Möglichkeiten. Um dies zu erreichen, war es notwendig, einige Anweisungen des LaTeX-Kerns zu ersetzen:

- Statt \@startsection wird \scr@startsection verwendet. Die Definition von \@startsection wird jedoch geprüft. Entspricht diese beim Laden der Klasse nicht den Erwartungen, so wird eine Warnung ausgegeben, diverse Möglichkeiten von KOMA-Script werden deaktiviert und \scr@startsection stützt sich auf eine Kopie von \@startsection, während \@startsection selbst \scr@startsection aufruft.
- Statt \@dblarg wird von den Gliederungsbefehlen eine eigene, interne Anweisung verwendet.
- Statt \secdef wird \SecDef verwendet, um die erwähnte Änderung bezüglich \@dblarg zu erreichen. Sollte die Definition von \secdef nicht den Erwartungen entsprechen, wird eine Warnung ausgegeben.
- \@sect wird umdefiniert, um diverse Erweiterungen durch KOMA-Script zu realisieren.
- \@ssect wird umdefiniert, um diverse Erweiterungen von KOMA-Script zu realisieren.
- \@xsect wird umdefiniert, um diverse Erweiterungen von KOMA-Script zu realisieren.

Es gibt Überlegungen, in künftigen Versionen von KOMA-Script die erwähnten Anweisungen aus dem LaTeX-Kern nicht mehr anzutasten, sondern intern komplett durch Eigenentwicklungen zu ersetzen. Bei Verwendung eigentlich inkompatibler Pakete würden dadurch automatisch Erweiterungen von KOMA-Script deaktiviert und die Gliederungsbefehle in die Hände jener Pakete gelegt. Gleichzeitig müssten aber zum Erhalt der Kompatibilität mit anderen Paketen zusätzliche Maßnahmen ergriffen werden.

Die erwähnten Ersatzanweisungen können von Paketautoren genau wie die LaTeX-Kern-Anweisungen verwendet werden, bieten dann aber automatisch die erweiterte Funktionalität von KOMA-Script. Jedoch sollten sie nicht umdefiniert werden, da sie sich jederzeit ändern können und dann die Funktionalität von KOMA-Script durch diese Umdefinierung erneut beeinträchtigt werden könnte. Die Bedeutung der Parameter ist der Anleitung zum LaTeX-Kern [BCJ+05] zu entnehmen. Als Ersatz für die Umdefinierung von Anweisungen bietet KOMA-Script die zuvor dokumentierten Haken.

v3.27

\At@startsection{*Code*}
\Before@ssect{*Code*}
\Before@sect{*Code*}

v3.27 Bis KOMA-Script 3.26b dienten diese Anweisungen ebenfalls als Ersatz für die Umdefinierung von \scr@startsection und \SecDef. Seit KOMA-Script 3.27 gelten sie jedoch als veraltet.

Intern wird \At@startsection seit KOMA-Script 3.27 über den Haken heading/postinit realisiert. \Before@ssect ist mit Hilfe von heading/branch/star implementiert und \Before@sect über heading/branch/nostar. *Code* wird den Haken per \AddtoDoHook hinzugefügt. Es ist nicht vorgesehen, einmal eingefügten *Code* wieder zu entfernen.

\appendixmore

Bei den KOMA-Script-Klassen gibt es innerhalb der Anweisung \appendix eine Besonderheit. Ist \appendixmore definiert, so wird diese Anweisung von \appendix ebenfalls ausgeführt. Intern wird das von den KOMA-Script-Klassen scrbook und scrreprt für die Realisierung der Layoutoption appendixprefix genutzt (siehe Abschnitt 3.16, Seite 101). Dies sollten Sie unbedingt beachten, falls Sie selbst das Makro \appendixmore definieren oder umdefinieren wollen. Wurde diese Option bereits verwendet, so erhalten Sie bei \newcommand{\appendixmore}{...} eine Fehlermeldung. Dadurch wird verhindert, dass Sie die Option 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 in Abschnitt 3.16, Seite 101). Damit die Konsistenz gewahrt bleibt, wollen Sie auch nicht, dass eine solche Zeile im Anhang verwendet wird. Stattdessen soll 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}}
```

Sollten Sie doch noch entscheiden, dass Sie die Option appendixprefix bei \documentclass setzen 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. Schreiben Sie dazu beispielsweise Folgendes in die Präambel Ihres Dokuments:

```
\newcommand*{\appendixmore}{%
  \renewcommand*{\sectionformat}{%
    \appendixname~\thesection\autodot\enskip}%
  \renewcommand*{\sectionmarkformat}{%
    \appendixname~\thesection\autodot\enskip}}
```

Die Erklärungen zu den in diesem Beispiel umdefinierten Anweisungen finden Sie in Abschnitt 3.16, Seite 117 und Seite 120.

## 18.9. Literaturverzeichnis

Die Erklärungen in diesem Abschnitt verlieren mit der Verwendung von Paketen wie biblatex zunehmend an Bedeutung. Die weitreichenden Möglichkeiten solcher Pakete ersetzen dann die hier beschriebenen Erweiterungen der KOMA-Script-Klassen.

```
\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 werden am Ende der Standardklasse die Anweisungen \@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 158) unterbrochenen Literaturverzeichnisses werden diese Anweisungen außerdem am Anfang und Ende jedes Teils, 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 die vier Anweisungen innerhalb von *Anweisung* bei Bedarf mit `\renewcommand` umzudefinieren sind.

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 [Mit11]) 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[openstyle]{twocolumopenstyle}{%
  \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.

Wie schon \BreakBibliography verlieren auch diese Anweisungen ganz oder teilweise ihre Wirkung, wenn thebibliography beispielsweise durch Verwendung von biblatex umdefiniert wird.

## 18.10. Mehr oder weniger obsolete Optionen und Anweisungen

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

# Zusätzliche Informationen zur Klasse scrlttr2 und Paket scrletter

In diesem Kapitel finden Sie zusätzliche Informationen zu der KOMA-Script-Klasse scrlttr2. Einige Teile des Kapitels sind dabei dem KOMA-Script-Buch [Koh20a] 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 genügen. So befasst sich beispielsweise der erste Abschnitt ausführlich mit der Definition und Verwendung von Variablen.

v3.15 Darüber hinaus gibt es seit KOMA-Script 3.15 das Paket scrletter, das zusammen mit einer der KOMA-Script-Klassen scrartcl, scrreprt oder scrbook verwendet werden kann. Es stellt nahezu die komplette Funktionalität von scrlttr2 für die drei genannten Klassen zur Verfügung. Einige wenige Unterschiede gibt es jedoch, die ebenfalls in diesem Kapitel genannt werden.

## 19.1. Variablen für fortgeschrittene Anwender

Neben der Möglichkeit, vordefinierte Variablen zu verwenden, bietet KOMA-Script auch Anweisungen, um neue Variablen zu definieren oder deren automatische Verwendung innerhalb der Geschäftszeile zu beeinflussen.

```
\newkomavar[Bezeichnung]{Name}
\newkomavar*[Bezeichnung]{Name}
\addtoreffields{Name}
\removereffields
\defaultreffields
```

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 zu *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 224) 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 `refline` ein, ob das Datum links, rechts oder gar nicht in der Geschäftszeile erscheinen soll. Diese Einstellungen 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 scrlayer-scrpage auch für `\markleft` (siehe Abschnitt 5.5, Seite 291). 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.

```
\foreachkomavar{Variablenliste}{Befehl}
\foreachnonemptykomavar{Variablenliste}{Befehl}
\foreachemptykomavar{Variablenliste}{Befehl}
\foreachkomavarifempty{Variablenliste}{Dann-Befehl}{Sonst-Befehl}
```

v3.27 Mit der Anweisung \foreachkomavar wird der angegebene *Befehl* für jede Variable aus der durch Komma separierten *Variablenliste* ausgeführt. Dabei wird der Name der jeweiligen Variablen als Argument an den *Befehl* angehängt.

Die Anweisung \foreachnonemptykomavar führt im Unterschied dazu *Befehl* nur aus, wenn \Ifkomavarempty sie als nicht leer erkennt. Leere Variablen in der *Variablenliste* haben dagegen keine Auswirkungen.

Dagegen führt \foreachemptykomavar den *Befehl* aus, wenn die Variable im Sinne von \Ifkomavarempty leer ist. Nicht leere Variablen in der *Variablenliste* haben entsprechend keine Auswirkungen.

Die Anweisung \foreachkomavarifempty ist quasi eine Verschmelzung beider vorgenannten. Sie führt *Dann-Befehl* für alle leeren Variablen aus, während *Sonst-Befehl* für die nicht leeren Variablen zur Anwendung kommt. Wie bei *Befehl* wird in beiden Fällen der Name der jeweiligen Variable als Argument angehängt.

## 19.2. Ergänzende Informationen zu den Seitenstilen

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

## 19.3. 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. Leider gibt es keine allgemein gültigen Regeln, um die Position von Anschriftfeldern und Ähnlichem für beliebige Papierformate zu berechnen. Vielmehr werden für unterschiedliche Papierformate unterschiedliche Parameter benötigt.

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. Dies gilt umso mehr, wenn scrletter verwendet wird, da die Einstellung der Papiergröße dann in erster Linie von der verwendeten Klasse abhängt.

\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 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:** Angenommen, Sie schreiben Briefe auf A5-Papier und stecken diese in Fensterbriefumschläge im Format C6. Damit wäre prinzipiell die Position des Adressfeldes die gleiche wie bei einem Brief in A4. Der Unterschied besteht 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{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.

visualize.lco

v3.04

Wenn man selbst lco-Dateien entwickelt, um die Positionen von Feldern des Briefbogens an eigene Wünsche anzupassen, ist es hilfreich, wenn einige Elemente sichtbar gemacht werden können. Zu diesem Zweck existiert die lco-Datei visualize.lco. Allerdings ist das Laden dieser lco-Datei auf die Dokumentpräambel beschränkt und seine Auswirkungen können nicht wieder rückgängig gemacht werden. Die lco-Datei benötigt die Pakete eso-pic und graphicx, die nicht zu KOMA-Script gehören.

\showfields{*Feldliste*}

Mit dieser Anweisung kann bei Verwendung von `visualize.lco` die Visualisierung von Feldern des Briefbogens aktiviert werden. Das Argument *Feldliste* ist dabei eine durch Komma separierte Liste der Felder, 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 den Anweisungen \setkomafont und \addtokomafont (siehe Abschnitt 4.9, Seite 192) für das Element `field` kann die Farbe der Visualisierung geändert werden. Voreingestellt ist `\normalcolor`.

\setshowstyle{*Stil*}
\edgesize

In der Voreinstellung werden von `visualize.lco` die einzelnen Felder durch Rahmen markiert. Dies entspricht dem *Stil* `frame`. Nach unten offene Felder werden nicht komplett umrahmt, sondern unten offen mit kleinen Pfeilen dargestellt. Als Alternative hierzu steht auch der *Stil* `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 *Stil* ist `edges`. Dabei werden die Ecken der Felder markiert. Bei nach unten offenen Feldern entfallen 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 von `visualize.lco` dienen dazu, eine Seite mit einer um 90° gedrehten Zeichnung eines Umschlags im Maßstab 1:1 auszugeben. 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 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 192) für das Element `letter` verändert werden. Voreingestellt ist `\normalcolor`.

Die Umschlagzeichnung wird automatisch bemaßt. Die Farbe der Bemaßung und die Größe von deren Beschriftung kann mit Hilfe der Anweisungen `\setkomafont` und `\addtokomafont` (siehe Abschnitt 4.9, Seite 192) 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. Des Weiteren 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.

## 19.4. Unterstützung verschiedener Sprachen

Die Klasse scrlttr2 und das Paket scrletter sind multilingual angelegt. Sie unterstützen also die Verwendung von Paketen wie babel oder polyglossia. Derzeit verfügbar sind Deutsch, Englisch, Finnisch, Französisch, Italienisch, Kroatisch, Niederländisch, Norwegisch, Polnisch, Schwedisch, Slowakisch, Spanisch und Tschechisch in den Varianten, die durch die auf Seite 521 und Seite 522 dokumentieren Befehle bestimmt sind.

v3.03
v3.08

Es gibt leider auch inkompatible Sprachpakete wie french (siehe [Gau07]). Dieses definiert beispielsweise `\opening` in einer Weise um, die nicht für scrlttr2 oder scrletter geeignet ist. Das ist ein vor Jahrzehnten gemeldeter Fehler von french.

Wird das Paket babel für die Umschaltung auf die Sprache verwendet, können vereinzelt ebenfalls Probleme auftreten. Bei babel lassen sich problematische Änderungen einer Sprache jedoch meist gezielt abschalten.Für Deutsch sind solche Probleme jedoch nicht bekannt und treten weder mit den Paketen german oder ngerman noch mit babel auf.

```
\yourrefname
\yourmailname
\myrefname
\customername
\invoicename
\subjectname
\ccname
\enclname
\headtoname
\headfromname
\datename
\pagename
\mobilephonename
\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, wie in Abschnitt 10.4 erklärt, angepasst werden. Von KOMA-Script 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 19.1, Seite 523 sind die Voreinstellungen für `english` und `ngerman` zu finden.

```
\captionsamerican
\captionsaustralien
\captionsaustrian
\captionsbritish
\captionscanadian
\captionscroatian
\captionsczech
\captionsdutch
\captionsenglish
\captionsfinnish
\captionsfrench
\captionsgerman
\captionsitalian
\captionsnaustrian
\captionsnewzealand
\captionsngerman
\captionsnorsk
\captionsnswissgerman
\captionspolish
\captionsslovak
\captionsspanish
\captionsswedish
\captionsswissgerman
\captionsUKenglish
\captionsUSenglish
```

Wird die Sprache eines Briefes gewechselt, so werden über diese Anweisungen die Begriffe aus Tabelle 19.1, Seite 523 umdefiniert. Sollte das verwendete Sprachumschaltpaket dies nicht unterstützen, so können die Anweisungen notfalls auch direkt verwendet werden.

```
\dateamerican
\dateaustralien
\dateaustrian
\datebritish
\datecanadian
\datecroatian
\dateczech
\datedutch
\dateenglish
\datefinnish
\datefrench
\dategerman
\dateitalian
\datenaustrian
\datenewzealand
\datengerman
\datenorsk
\datenswissgerman
\datepolish
\dateslovak
\datespanish
\dateswedish
\dateswissgerman
\dateUKenglish
\dateUSenglish
```

Je nach verwendeter Sprache werden die Datumsangaben des numerischen Datums (siehe Option `numericaldate`, Abschnitt 4.10, Seite 223) in unterschiedlicher Form umgesetzt. Die genauen Angaben sind Tabelle 19.2, Seite 524 zu entnehmen.

## 19.5. Obsolete Anweisungen

Im KOMA-Script-Buch [Koh20a] finden sich an dieser Stelle weitere Informationen.

Tabelle 19.1.: Voreinstellung für die sprachabhängigen Begriffe bei den 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 |
| `\mobilephonename` | Mobile phone | Mobiltelefon |
| `\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 KOMA-Script nicht überschrieben. Abweichungen im Wortlaut sind daher möglich und der Anleitung des verwendeten Sprachpakets zu entnehmen.

Tabelle 19.2.: Sprachabhängige Ausgabeformate für das Datum

| Anweisung | Ausgabebeispiel |
|---|---|
| \dateamerican | 12/24/1993 |
| \dateaustralien | 24/12/1993 |
| \dateaustrian | 24. 12. 1993 |
| \datebritish | 24/12/1993 |
| \datecanadian | 1993/12/24 |
| \datecroatian | 24. 12. 1993. |
| \dateczech | 24. 12. 1993 |
| \datedutch | 24. 12. 1993 |
| \dateenglish | 24/12/1993 |
| \datefinnish | 24.12.1993. |
| \datefrench | 24. 12. 1993 |
| \dategerman | 24. 12. 1993 |
| \dateitalian | 24. 12. 1993 |
| \datenaustrian | 24. 12. 1993 |
| \datenewzealand | 24/12/1993 |
| \datengerman | 24. 12. 1993 |
| \datenorsk | 24.12.1993 |
| \datenswissgerman | 24. 12. 1993 |
| \datepolish | 24. 12. 1993 |
| \dateslovak | 24. 12. 1993 |
| \datespanish | 24. 12. 1993 |
| \dateswedish | 24/12 1993 |
| \dateswissgerman | 24. 12. 1993 |
| \dateUKenglish | 24/12/1993 |
| \dateUSenglish | 12/24/1993 |

# Änderungsliste

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.

scrartcl


v2.8p ..... 60, 70, 85, 110, 113, 129, 141, 252
v2.8q ..... 43, 75, 97, 142, 148, 151
v2.96a ..... 32, 56, 114, 164, 302
v2.97c ..... 70, 79, 309
v3.00 31, 55–57, 67, 74, 75, 81, 84, 91, 92, 94–96, 102, 142, 143, 151, 152, 156, 158, 159, 162, 163, 268, 301, 418, 419, 462, 463
v3.01a ..... 32, 57, 164, 302
v3.02 ..... 130
v3.06 ..... 98, 151, 153
v3.07 ..... 63, 98
v3.08 ..... 81, 82, 482, 484
v3.09 ..... 138–140
v3.10 ..... 103, 104, 107, 124, 145
v3.12 .... 61, 62, 65–67, 71, 73, 77, 102, 104, 121, 122, 153, 157, 158, 196, 272, 306, 465
v3.15 ..... 65, 78, 79, 151, 486, 487, 499, 500, 508
v3.17 ..... 58, 117, 485, 486, 500, 501
v3.18 ..... 75, 77, 159, 160, 487
v3.19 ..... 485, 489, 505
v3.20 ..... 148, 489, 493, 495
v3.22 ..... 107, 109
v3.23 ..... 98
v3.24 ..... 485, 488
v3.25 ..... 57, 69, 149, 486, 502
v3.26 ..... 488, 490, 492
v3.27 ..... 107, 109, 499, 507–509
v3.28 ..... 83, 122
v3.33 ..... 63, 64, 96, 126, 251
v3.36 ..... 99, 137, 250, 316, 481
v3.39 ..... 60, 110, 124, 129
v3.44 ..... 62, 63, 128
v3.46 ..... 138–140

v3.49.3 ..... 107, 109, 112
scrbase
v3.05a ..... 348
v3.08 ..... 336
v3.12 ..... 325, 331, 335–337, 339–341, 344, 345
v3.15 ..... 326, 335, 336
v3.18 ..... 327
v3.19 ..... 339
v3.20 ..... 328, 336
v3.23 ..... 331
v3.25 ..... 337
v3.27 ..... 330, 331, 333, 342, 349–352
v3.28 ..... 339–341
v3.37 ..... 343
v3.49.3 ..... 348
scrbook
v2.8o ..... 119
v2.8p ..... 60, 70, 85, 110, 113, 123, 129, 141, 252
v2.8q ..... 43, 75, 97, 142, 148, 151
v2.96a ..... 32, 56, 102, 106, 114, 164, 302
v2.97c ..... 70, 79, 309
v2.97e ..... 99
v3.00 31, 55–57, 67, 75, 81, 84, 91, 92, 94–96, 100, 102, 142, 143, 151, 152, 156, 158, 159, 162, 163, 268, 301, 418, 419, 462, 463
v3.01a ..... 32, 57, 164, 302
v3.02 ..... 130, 500
v3.06 ..... 98, 151, 153
v3.07 ..... 63, 98
v3.08 ..... 81, 82, 482, 484
v3.09 ..... 138–140
v3.10 ..... 103, 104, 107, 124, 145
v3.12 .... 61, 62, 65–67, 71, 73, 77, 102, 104, 121, 122, 153, 157, 158, 196, 272, 306, 465
v3.15 ..... 61, 76, 78, 79, 116, 151, 486, 487, 499, 500, 508
v3.17 ..... 58, 117, 118, 485, 486, 500, 501
v3.18 ..... 75, 77, 102, 159, 160, 487
v3.19 ..... 485, 489, 503, 505
v3.20 ..... 148, 489, 493, 495
v3.21 ..... 482
v3.22 ..... 107, 109
v3.23 ..... 98

v3.24 ..... 485, 488
v3.25 ..... 57, 69, 149, 486, 502
v3.26 ..... 488, 490–492
v3.27 ..... 107, 109, 499, 507–509
v3.28 ..... 83, 122
v3.33 ..... 63, 64, 96, 126, 251
v3.36 ..... 99, 137, 250, 316, 481
v3.39 ..... 60, 110, 124, 129
v3.44 ..... 62, 63, 128
v3.46 ..... 138–140
v3.49 ..... 99, 100
v3.49.3 ..... 107, 109, 112
screxend
v3.39 ..... 304
scrextend
v3.01a ..... 32, 57, 164, 302
v3.02 ..... 318
v3.10 ..... 317
v3.12 ..... 61, 62, 65, 66, 196, 272, 306, 307, 310, 311, 465
v3.23 ..... 316
v3.25 ..... 303, 308
v3.28 ..... 311
v3.33 ..... 96
v3.36 ..... 99, 250, 316, 481
scrlayer
v3.12 ..... 418
v3.16 ..... 423, 424, 429, 432, 434, 446, 451
v3.18 ..... 425, 426, 428, 430, 440
v3.19 ..... 428, 431
v3.24 ..... 440
v3.26 ..... 430
v3.28 ..... 292, 448, 450
scrlayer-notecolumn
v0.1.2583 ..... 471
scrlayer-scrpage
v3.12 ..... 267, 451
v3.14 ..... 277, 280, 282, 292
v3.24 ..... 440
v3.25 ..... 269
v3.41 ..... 288

scrletter
v3.15 ... 162, 513
v3.17 ... 195, 215, 216, 228, 229
v3.19 ... 182, 240
v3.26 ... 176, 178
v3.27 ... 176, 181, 208, 515
v3.28 ... 171
v3.44 ... 232, 252
v3.48 ... 235
scrlfile
v2.96 ... 360, 361
v3.03 ... 360
v3.08 ... 362, 363
v3.09 ... 357
v3.12 ... 362–364
v3.32 ... 355, 357, 360, 361
v3.35 ... 355
scrlogo
v3.36 ... 321
scrlttr2
v2.9i ... 171
v2.9t ... 32, 56, 164, 237, 302
v2.95 ... 181
v2.96 ... 215
v2.97 ... 257
v2.97c ... 199, 201, 207, 215, 217, 224, 228
v2.97d ... 222
v2.97e ... 198, 200–202, 235, 237
v3.00 ... 165, 244–248
v3.01 ... 230, 231
v3.01a ... 32, 57, 164, 302
v3.02 ... 253, 519
v3.03 ... 171, 215, 217–220
v3.04 ... 256, 516
v3.05 ... 202, 237
v3.06 ... 250
v3.07 ... 193, 250
v3.08 ... 167, 168, 235, 238, 244, 519
v3.09 ... 224
v3.12 ... 167, 195, 208, 225

v3.13 ..... 519
v3.14 ..... 256
v3.15 ..... 486
v3.17 ..... 195, 215, 216, 228, 229
v3.19 ..... 182, 240
v3.23 ..... 250
v3.25 ..... 165, 486
v3.26 ..... 176, 178
v3.27 ..... 176, 181, 208, 515
v3.28 ..... 171, 239
v3.33 ..... 193, 194, 248
v3.39 ..... 191, 216, 217, 226, 228, 252
v3.44 ..... 193, 194, 232, 252
v3.48 ..... 235

scrreprt

v2.8o ..... 119
v2.8p ..... 60, 70, 85, 110, 113, 123, 129, 141, 252
v2.8q ..... 43, 75, 97, 142, 148, 151
v2.96a ..... 32, 56, 102, 106, 114, 164, 302
v2.97c ..... 70, 79, 309
v3.00 31, 55–57, 67, 74, 75, 81, 84, 91, 92, 94–96, 100, 102, 142, 143, 151, 152, 156, 158, 159, 162, 163, 268, 301, 418, 419, 462, 463
v3.01a ..... 32, 57, 164, 302
v3.02 ..... 130, 500
v3.06 ..... 98, 151, 153
v3.07 ..... 63, 98
v3.08 ..... 81, 82, 482, 484
v3.09 ..... 138–140
v3.10 ..... 103, 104, 107, 124, 145
v3.12 .... 61, 62, 65–67, 71, 73, 77, 102, 104, 121, 122, 153, 157, 158, 196, 272, 306, 465
v3.15 ..... 61, 76, 78, 79, 116, 151, 486, 487, 499, 500, 508
v3.17 ..... 58, 117, 118, 485, 486, 500, 501
v3.18 ..... 75, 77, 102, 159, 160, 487
v3.19 ..... 485, 489, 503, 505
v3.20 ..... 148, 489, 493, 495
v3.21 ..... 482
v3.22 ..... 107, 109
v3.23 ..... 98
v3.24 ..... 485, 488
v3.25 ..... 57, 69, 149, 486, 502

v3.26 ..... 488, 490–492
v3.27 ..... 107, 109, 499, 507–509
v3.28 ..... 83, 122
v3.33 ..... 63, 64, 96, 126, 251
v3.36 ..... 99, 137, 250, 316, 481
v3.39 ..... 60, 110, 124, 129
v3.44 ..... 62, 63, 128
v3.46 ..... 138–140
v3.49.3 ..... 107, 109, 112

tocbasic
v3.01 ..... 377
v3.06 ..... 400
v3.09 ..... 400
v3.10 ..... 376
v3.12 ..... 371, 372, 376, 396
v3.17 ..... 376
v3.20 ..... 377–379, 382, 390, 392–394, 396, 400, 402–404
v3.21 ..... 382, 389, 400, 402
v3.25 ..... 403, 404
v3.26 ..... 383
v3.27 ..... 376, 381–383, 385–387, 389–391, 400
v3.28 ..... 366, 378
v3.31 ..... 382–385
v3.35 ..... 401
v3.39 ..... 385–387
v3.42 ..... 410, 411
v3.46 ..... 406, 412
v3.49 ..... 384, 385

typearea
v3.00 . 31, 33, 34, 36, 37, 40, 41, 43, 44, 47, 48, 55, 56, 162, 163, 268, 301, 418, 419, 462, 463
v3.01b ..... 32, 48, 56, 164, 302
v3.02c ..... 48
v3.05a ..... 51
v3.11 ..... 478, 480
v3.12 ..... 41, 45
v3.17 ..... 50, 477
v3.18 ..... 480
v3.20 ..... 51

v3.22 ..... 48

v0.1.2583
scrlayer-notecolumn ..... 471
v2.8o
scrbook ..... 119
scrreprt ..... 119
v2.8p
scrartcl ..... 60, 70, 85, 110, 113, 129, 141, 252
scrbook ..... 60, 70, 85, 110, 113, 123, 129, 141, 252
scrreprt ..... 60, 70, 85, 110, 113, 123, 129, 141, 252
v2.8q
scrartcl ..... 43, 75, 97, 142, 148, 151
scrbook ..... 43, 75, 97, 142, 148, 151
scrreprt ..... 43, 75, 97, 142, 148, 151
v2.9i
scrlttr2 ..... 171
v2.9t
scrlttr2 ..... 32, 56, 164, 237, 302
v2.95
scrlttr2 ..... 181
v2.96
scrlfile ..... 360, 361
scrlttr2 ..... 215
v2.96a
scrartcl ..... 32, 56, 114, 164, 302
scrbook ..... 32, 56, 102, 106, 114, 164, 302
scrreprt ..... 32, 56, 102, 106, 114, 164, 302
v2.97
scrlttr2 ..... 257
v2.97c
scrartcl ..... 70, 79, 309
scrbook ..... 70, 79, 309
scrlttr2 ..... 199, 201, 207, 215, 217, 224, 228
scrreprt ..... 70, 79, 309
v2.97d
scrlttr2 ..... 222
v2.97e
scrbook ..... 99
scrlttr2 ..... 198, 200–202, 235, 237

v3.00
scrartcl ... 31, 55–57, 67, 74, 75, 81, 84, 91, 92, 94–96, 102, 142, 143, 151, 152, 156, 158, 159, 162, 163, 268, 301, 418, 419, 462, 463
scrbook .. 31, 55–57, 67, 75, 81, 84, 91, 92, 94–96, 100, 102, 142, 143, 151, 152, 156, 158, 159, 162, 163, 268, 301, 418, 419, 462, 463
scrlttr2 ... 165, 244–248
scrreprt ... 31, 55–57, 67, 74, 75, 81, 84, 91, 92, 94–96, 100, 102, 142, 143, 151, 152, 156, 158, 159, 162, 163, 268, 301, 418, 419, 462, 463
typearea ... 31, 33, 34, 36, 37, 40, 41, 43, 44, 47, 48, 55, 56, 162, 163, 268, 301, 418, 419, 462, 463
v3.01
scrlttr2 ... 230, 231
tocbasic ... 377
v3.01a
scrartcl ... 32, 57, 164, 302
scrbook ... 32, 57, 164, 302
scrextend ... 32, 57, 164, 302
scrlttr2 ... 32, 57, 164, 302
scrreprt ... 32, 57, 164, 302
v3.01b
typearea ... 32, 48, 56, 164, 302
v3.02
scrartcl ... 130
scrbook ... 130, 500
scrextend ... 318
scrlttr2 ... 253, 519
scrreprt ... 130, 500
v3.02c
typearea ... 48
v3.03
scrlfile ... 360
scrlttr2 ... 171, 215, 217–220
v3.04
scrlttr2 ... 256, 516
v3.05
scrlttr2 ... 202, 237
v3.05a
scrbase ... 348
typearea ... 51
v3.06

scrartcl . . . . . 98, 151, 153
scrbook . . . . . 98, 151, 153
scrlttr2 . . . . . 250
scrreprt . . . . . 98, 151, 153
tocbasic . . . . . 400
v3.07
scrartcl . . . . . 63, 98
scrbook . . . . . 63, 98
scrlttr2 . . . . . 193, 250
scrreprt . . . . . 63, 98
v3.08
scrartcl . . . . . 81, 82, 482, 484
scrbase . . . . . 336
scrbook . . . . . 81, 82, 482, 484
scrlfile . . . . . 362, 363
scrlttr2 . . . . . 167, 168, 235, 238, 244, 519
scrreprt . . . . . 81, 82, 482, 484
v3.09
scrartcl . . . . . 138–140
scrbook . . . . . 138–140
scrlfile . . . . . 357
scrlttr2 . . . . . 224
scrreprt . . . . . 138–140
tocbasic . . . . . 400
v3.10
scrartcl . . . . . 103, 104, 107, 124, 145
scrbook . . . . . 103, 104, 107, 124, 145
scrextend . . . . . 317
scrreprt . . . . . 103, 104, 107, 124, 145
tocbasic . . . . . 376
v3.11
typearea . . . . . 478, 480
v3.12
scrartcl . . . 61, 62, 65–67, 71, 73, 77, 102, 104, 121, 122, 153, 157, 158, 196, 272, 306, 465
scrbase . . . . . 325, 331, 335–337, 339–341, 344, 345
scrbook . . 61, 62, 65–67, 71, 73, 77, 102, 104, 121, 122, 153, 157, 158, 196, 272, 306, 465
scrextend . . . . . 61, 62, 65, 66, 196, 272, 306, 307, 310, 311, 465
scrlayer . . . . . 418
scrlayer-scrpage . . . . . 267, 451
scrlfile . . . . . 362–364

scrlttr2 ..... 167, 195, 208, 225
scrreprt .. 61, 62, 65–67, 71, 73, 77, 102, 104, 121, 122, 153, 157, 158, 196, 272, 306, 465
tocbasic ..... 371, 372, 376, 396
typearea ..... 41, 45
v3.13
scrlttr2 ..... 519
v3.14
scrlayer-scrpage ..... 277, 280, 282, 292
scrlttr2 ..... 256
v3.15
scrartcl ..... 65, 78, 79, 151, 486, 487, 499, 500, 508
scrbase ..... 326, 335, 336
scrbook ..... 61, 76, 78, 79, 116, 151, 486, 487, 499, 500, 508
scrletter ..... 162, 513
scrlttr2 ..... 486
scrreprt ..... 61, 76, 78, 79, 116, 151, 486, 487, 499, 500, 508
v3.16
scrlayer ..... 423, 424, 429, 432, 434, 446, 451
v3.17
scrartcl ..... 58, 117, 485, 486, 500, 501
scrbook ..... 58, 117, 118, 485, 486, 500, 501
scrletter ..... 195, 215, 216, 228, 229
scrlttr2 ..... 195, 215, 216, 228, 229
scrreprt ..... 58, 117, 118, 485, 486, 500, 501
tocbasic ..... 376
typearea ..... 50, 477
v3.18
scrartcl ..... 75, 77, 159, 160, 487
scrbase ..... 327
scrbook ..... 75, 77, 102, 159, 160, 487
scrlayer ..... 425, 426, 428, 430, 440
scrreprt ..... 75, 77, 102, 159, 160, 487
typearea ..... 480
v3.19
scrartcl ..... 485, 489, 505
scrbase ..... 339
scrbook ..... 485, 489, 503, 505
scrlayer ..... 428, 431
scrletter ..... 182, 240
scrlttr2 ..... 182, 240

scrreprt ..... 485, 489, 503, 505
v3.20
scrartcl ..... 148, 489, 493, 495
scrbase ..... 328, 336
scrbook ..... 148, 489, 493, 495
scrreprt ..... 148, 489, 493, 495
tocbasic ..... 377–379, 382, 390, 392–394, 396, 400, 402–404
typearea ..... 51
v3.21
scrbook ..... 482
scrreprt ..... 482
tocbasic ..... 382, 389, 400, 402
v3.22
scrartcl ..... 107, 109
scrbook ..... 107, 109
scrreprt ..... 107, 109
typearea ..... 48
v3.23
scrartcl ..... 98
scrbase ..... 331
scrbook ..... 98
scrextend ..... 316
scrlttr2 ..... 250
scrreprt ..... 98
v3.24
scrartcl ..... 485, 488
scrbook ..... 485, 488
scrlayer ..... 440
scrlayer-scrpage ..... 440
scrreprt ..... 485, 488
v3.25
scrartcl ..... 57, 69, 149, 486, 502
scrbase ..... 337
scrbook ..... 57, 69, 149, 486, 502
scrextend ..... 303, 308
scrlayer-scrpage ..... 269
scrlttr2 ..... 165, 486
scrreprt ..... 57, 69, 149, 486, 502
tocbasic ..... 403, 404
v3.26

scrartcl ... 488, 490, 492
scrbook ... 488, 490–492
scrlayer ... 430
scrletter ... 176, 178
scrlttr2 ... 176, 178
scrreprt ... 488, 490–492
tocbasic ... 383
v3.27
scrartcl ... 107, 109, 499, 507–509
scrbase ... 330, 331, 333, 342, 349–352
scrbook ... 107, 109, 499, 507–509
scrletter ... 176, 181, 208, 515
scrlttr2 ... 176, 181, 208, 515
scrreprt ... 107, 109, 499, 507–509
tocbasic ... 376, 381–383, 385–387, 389–391, 400
v3.28
scrartcl ... 83, 122
scrbase ... 339–341
scrbook ... 83, 122
scrextend ... 311
scrlayer ... 292, 448, 450
scrletter ... 171
scrlttr2 ... 171, 239
scrreprt ... 83, 122
tocbasic ... 366, 378
v3.31
tocbasic ... 382–385
v3.32
scrlfile ... 355, 357, 360, 361
v3.33
scrartcl ... 63, 64, 96, 126, 251
scrbook ... 63, 64, 96, 126, 251
scrextend ... 96
scrlttr2 ... 193, 194, 248
scrreprt ... 63, 64, 96, 126, 251
v3.35
scrlfile ... 355
tocbasic ... 401
v3.36
scrartcl ... 99, 137, 250, 316, 481

scrbook ..... 99, 137, 250, 316, 481
scrextend ..... 99, 250, 316, 481
scrlogo ..... 321
scrreprt ..... 99, 137, 250, 316, 481
v3.37
scrbase ..... 343
v3.39
scrartcl ..... 60, 110, 124, 129
scrbook ..... 60, 110, 124, 129
screxend ..... 304
scrlttr2 ..... 191, 216, 217, 226, 228, 252
scrreprt ..... 60, 110, 124, 129
tocbasic ..... 385–387
v3.41
scrlayer-scrpage ..... 288
v3.42
tocbasic ..... 410, 411
v3.44
scrartcl ..... 62, 63, 128
scrbook ..... 62, 63, 128
scrletter ..... 232, 252
scrlttr2 ..... 193, 194, 232, 252
scrreprt ..... 62, 63, 128
v3.46
scrartcl ..... 138–140
scrbook ..... 138–140
scrreprt ..... 138–140
tocbasic ..... 406, 412
v3.48
scrletter ..... 235
scrlttr2 ..... 235
v3.49
scrbook ..... 99, 100
tocbasic ..... 384, 385
v3.49.3
scrartcl ..... 107, 109, 112
scrbase ..... 348
scrbook ..... 107, 109, 112
scrreprt ..... 107, 109, 112

# 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 https://mirror.ctan.org/ 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 neuste 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/.

[BB13] Johannes Braams und Javier Bezos:
*Babel*, Dezember 2013.
CTAN://macros/latex/required/babel/.

[BCJ+05] Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley und Rainer Schöpf:
*The LaTeX2e Source*, Dezember 2005.

[BJ04] Johannes L. Braams und Theo Jurriens:
*The supertabular environment*, Februar 2004.
CTAN://macros/latex/contrib/supertabular.

[Car99a] David Carlisle:
*The keyval package*, März 1999.
CTAN://macros/latex/required/graphics/.

[Car99b] David Carlisle:
*The tabularx package*, Januar 1999.
CTAN://macros/latex/required/tools/.

[Car04] David Carlisle:
*The longtable package*, Februar 2004.
CTAN://macros/latex/required/tools/.

[Car17] David P. Carlisle:
*Packages in the 'graphics' bundle*, Juni 2017.
CTAN://macros/latex/required/graphics/.

[CRSR22] François Charette, Arthur Reutenauer, Jürgen Spitzmüller und Bastien Roucariès:
*Polyglossia: Modern multilingual typesetting with XeLaTeX and LuaLaTeX*, April 2022.
CTAN://macros/unicodetex/latex/polyglossia.

[Dal10] Patrick W. Daly:
*Natural sciences citations and references*, September 2010.
CTAN://macros/latex/contrib/natbib/.

[DGS+12] Marco Daniel, Patrick Gundlach, Walter Schmidt, Jörg Knappen, Hubert Partl und Irene Hyna:
*LaTeX 2ε-Kurzbeschreibung*, Juli 2012.
CTAN://info/lshort/german/.

[DUD96] DUDEN:
*Die deutsche Rechtschreibung*. DUDENVERLAG, Mannheim, 21. Auflage, 1996.

[Fai11] Robin Fairbairns:
*footmisc — a portmanteau package for customising footnotes in LaTeX*, Juni 2011.
CTAN://macros/latex/contrib/footmisc/.

[Gau07] Bernard Gaulle:
*Les distributions de fichiers de francisation pour latex*, Mai 2007.
CTAN://language/french/.

[Gre12] Enrico Gregorio:
*The xpatch package, extending etoolbox patching commands*, Oktober 2012.
CTAN://macros/latex/contrib/xpatch/.

[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/.

[Kie10] Axel Kielhorn:
*adrconv*, April 2010.
CTAN://macros/latex/contrib/adrconv/.

[Knu90] Donald E. Knuth:
*The TEXbook*, Band A der Reihe *Computers and Typesetting*. Addison-Wesley Publishing Company, Reading, Mass., 19. Auflage, 1990.

[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.

[Koh12] Markus Kohm:
*Non-floating margin notes with* marginnote, März 2012.
CTAN://macros/latex/contrib/marginnote/.

[Koh14] Markus Kohm:
*Creating more than one index using* splitidx *and* splitindex, April 2014.
CTAN://macros/latex/contrib/splitindex/.

[Koh15] Markus Kohm:
*chapteratlists=entry automatisch nur für Kapitel mit Einträgen*, Juli 2015.
https://komascript.de/comment/5070#comment-5070.

[Koh20a] Markus Kohm:
*KOMA-Script*. Edition DANTE. Lehmanns Media, Berlin, 7. Auflage, 2020, ISBN 978-3-96543-097-6. Print-Ausgabe.

[Koh20b] Markus Kohm:
*KOMA-Script*. Edition DANTE. Lehmanns Media, Berlin, 7. Auflage, 2020, ISBN 978-3-96543-103-4. eBook-Ausgabe.

[Koh23a] Markus Kohm:
*Emulating Former KOMA-Script Package* scrhack, 3.42 Auflage, August 2023.
CTAN://macros/latex/contrib/scrhack/scrhack.pdf.

[Koh23b] Markus Kohm:
scrhack *— an enhancement to package* float *using package* tocbasic, August 2023.
CTAN://macros/latex/contrib/floatbytocbasic.

[Koh23c] Markus Kohm:
scrwfile *– use LaTeX* .aux *file in place of* \newfile *files*, April 2023.
https://www.ctan.org/pkg/scrwfile.

[Koh24] Markus Kohm:
*Typeset formalized legal documents such as contracts, statutes, etc*, Februar 2024.
CTAN://macros/latex/contrib/contract.

[Lam87] Leslie Lamport:
*MakeIndex: An index processor for LaTeX*, Februar 1987.
CTAN://indexing/makeindex/doc/makeindex.pdf.

[Lap08] Olga Lapko:
*The floatrow package*, August 2008.
CTAN://macros/latex/contrib/floatrow/.

[Leh11] Philipp Lehman:
*The etoolbox package*, Januar 2011.
CTAN://macros/latex/contrib/etoolbox/.

[Lin01] Anselm Lingnau:
*An improved environment for floats*, November 2001.
CTAN://macros/latex/contrib/float/.

[Mit11] Frank Mittelbach:
*An environment for multicolumn output*, Juni 2011.
CTAN://macros/latex/required/tools/.

[Nie15] Rolf Niepraschk:
*The eso-pic package*, Juli 2015.
CTAN://macros/latex/contrib/eso-pic/.

[Pac] Jean Marie Pacquet:
*KomaLetter2; Example by Jean-Marie Pacquet (French style)*. Wiki.
http://wiki.lyx.org/Examples/KomaLetter2#toc6.

[Pro26] The LaTeX Project:
*The LaTeX3 Interface*, August 2026.
CTAN://macros/latex/required/l3kernel/interface3.pdf.

[Sch09] Martin Schröder:
*The ragged2e package*, Juni 2009.
CTAN://macros/latex/contrib/ms/.

[Sch13] R Schlicht:
*The microtype package: An interface to the micro-typographic extensions of pdfTeX*, Mai 2013.
CTAN://macros/latex/contrib/microtype/.

[Sch18] Martin Scharrer:
*The mwe package*, März 2018.
CTAN://macros/latex/contrib/mwe/.

[Som13] Axel Sommerfeldt:
*Anpassen der Abbildungs- und Tabellenbeschriftungen mit Hilfe des caption-Paketes*, Mai 2013.
CTAN://macros/latex/contrib/caption/.

[Tea98] The $\mathcal{NTS}$ Team:
*The ε-TEX manual*, Februar 1998.
CTAN://systems/e-tex/v2/doc/etex_man.pdf.

[Tea05] LaTeX3 Project Team:
*LaTeX 2ε font selection*, November 2005.
CTAN://macros/latex/doc/fntguide.pdf.

[Tea06] LaTeX3 Project Team:
*LaTeX 2ε for class and package writers*, Februar 2006.
CTAN://macros/latex/doc/clsguide.pdf.

[Tea23] LaTeX Project Team:
*LaTeX 2ε for authors — historic version*, November 2023.
CTAN://macros/latex/doc/usrguide-historic.pdf.

[Tea26] LaTeX Project Team:
*LaTeX for authors — current version*, Februar 2026.
CTAN://macros/latex/doc/usrguide.pdf.

[TF11] Geoffrey Tobin und Robin Fairbairns:
*setspace LaTeX package*, Dezember 2011.
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.

[Ume10] Hideo Umeki:
*The geometry package*, September 2010.
CTAN://macros/latex/contrib/geometry/.

[vO04] Piet van Oostrum:
*Page layout in LaTeX*, März 2004.
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


`@everystyle@` (Seitenstil) ..... 436, **439**

**A**

Abbildungen ..... **136–150**
  Nummern ..... 99
  Verzeichnis ..... 99, 151, 154
Absatz ..... **80**
  Abstand ..... 80
  Auszeichnung ..... **80–82**, **238–239**
Abschnitt ..... **107**
  Nummer ..... 99
Absender ..... 202, 208
Absenderergänzung ..... 202, **221–222**
Adressdatei ..... **262–266**, 296
Adressverzeichnis ..... **265**
Alias
  Seitenstil ..... 434, **435–436**
Anhang ..... 101, 119, **155–156**
Anlagen ..... 187
Anrede ..... 182
Anschrift ..... **180–181**, 182, **215–218**, **220**
Anweisung
  → Index der Befehle etc. ..... 547
Ausgabe
  erzwungene ..... 474
Ausgleich ..... 58
Autor ..... **70**, **309**

**B**

Bankverbindung ..... 237
Befehl
  → Index der Befehle etc. ..... 547
Betreff ..... 182, **228–230**, **232**
BibTeX ..... 156
Bindeanteil ..... *siehe auch* Bindekorrektur
Bindekorrektur ..... **26**, 27, 29, 33
Bindung ..... **26**, *siehe auch* Bindekorrektur
`boxed` (float-Stil) ..... 145
Briefbogen ..... **196–237**
Briefbogenfuß ..... **234–237**
Briefe ..... **162–266**
  Aufbau ..... **178–188**
Briefkopf ..... 182, **201–202**, 203, **213**
Bundsteg ..... 26, 27

**C**

CM-Fonts ..... 112

**D**

Datei
  → Index der Dateien etc. ..... 561
Dateiendung ..... *siehe* Dateierweiterung
Dateierweiterung ..... **366–417**
Datum ..... 70, 223, 225, 309, 522
doppelseitig ..... 26, 85
Durchschuss ..... **28**, 38
DVI ..... 49

**E**

Ebenen ..... **418–450**, 467
  Seitenstil ..... 434, **436–443**
EC-Fonts ..... 112
einseitig ..... 26
Einzug ..... **80**
Element ..... 488
  → Index der Elemente ..... 560
`empty` (Seitenstil) . **84–86**, 87, 91, 92, **241–244**, 245, 246, 312, 313, **439**, 465, 470
Endfassung ..... 57, 165, 303
Entwurf ..... **57–58**, **165**, **303**

## F

Faltmarke ... 182, 198, **196–201**
Fax ... 208
float-Stile
  `boxed` ... 145
  `komaabove` ... **145**
  `komabelow` ... **145**
  `plain` ... 145
  `ruled` ... 145
Folgeseite ... 196, 244
Formeln ... **136**, **254**
Fuß ... 182, 234, 244
  Breite ... 292
  Höhe ... **45–46**, **269**, **443–444**
Fußnoten .. 70, **93–99**, **247–250**, 310, **313–316**

## G

Gedichte ... 131, 253
Geschäftszeile ... 182, **223–227**, 513
Gleichungen ... **136**, **254**
  Ausrichtung ... 136
  Nummern ... 99, 136
Gleitumgebungen ... **136–150**, 154
Gliederung 105, **100–123**, *siehe auch* Abschnitt, Kapitel, Überschriften
  Stil
    `chapter` ... 487, 491, 503
    `part` ... 487, 490, 502
    `section` ... 488, 492, 505
Gruß ... 183, 232, 234, **232–234**

## H

Haken .. **99**, **181**, **250**, **316**, 327, **349–352**, 394, 395, **434**, 435–439, 442, 480, **481**, **507**
Haken
  → Index der Haken ... 567
Handy ... 208
Hauptteil ... 99
`headings` (Seitenstil)
  ... **84–86**, 87, 120, 122, 168, **241–244**, 244, **273**, 273, 275, 284, 342, 446, 452
*hook* ... 181, 480

## I

Inhaltsverzeichnis ... **74–80**, 88, 99, 112, 292

## K

Kapitel ... **107**
  Anfang ... 100
  Präambel ... 122
  Seitenstil ... 87, 88
  Überschrift ... 101, 119
Klasse
  → Index der Dateien etc. ... 561
Kolumnentitel ... 42, 85
  automatisch ... **84**, 112, 120
  lebend ... 284
  manuell ... **85**, 291, 448
  statisch ... 284
`komaabove` (float-Stil) ... **145**
`komabelow` (float-Stil) ... **145**
Kompatibilität .. **32–33**, **56–57**, **163–164**, **302**
Kopf ... 182, 244
  Breite ... 292
  Höhe ... **44–45**, **269**, **443–444**

## L

Länge
  → Index der Befehle etc. ... 547
  → Index der Längen etc. ... 559
`lco`-Datei 171, 176, 200–202, 218, 219, 226, 227, 236, 237, **255–262**, **515–519**
`letter` (Seitenstil) ... 240, 244
Letter-Class-Option ... **255–262**
Linie ... 240
Listen ... **126–135**, **250–254**, **318–319**
Literaturverzeichnis ... 100, **156–159**
LM-Fonts ... 112
Lochermarke ... 200
Logo ... 211

## M

Makro
  → Index der Befehle etc. ... 547
Maßlinien ... 287
Mathematik ... **136**, **254**
Mobiltelefon ... 208
`myheadings` (Seitenstil) ... **85–86**, 87, 168, **241–244**, 244, 284

## N

Nachspann ... 99

Norm ..... 256
Notiz
anhängige ..... 474
irrtümliche ..... 474
Nummerierung ..... 105, 112, **121**, 128, 252

**O**

Option
→ Index der Optionen ..... 563
Optionen ..... **30–32**, **55–56**, **162–163**, 256, **267–268**, **300–302**, **418–419**, **462–463**
Ort ..... 225

**P**

Paginierung ..... 42, 85
Paket
→ Index der Dateien etc. ..... 561
Papier ..... 27
Ausrichtung ..... 48
Format ..... **26**, **515–516**
PDF ..... 49
`plain` (float-Stil) ..... 145
`plain` (Seitenstil) **85–86**, 87, 92, **241–244**, 246, **273**, 273, 275, 311, 313
`plain.letter` (Seitenstil) ..... 240
`plain.scrheadings` (Seitenstil) . **273**, 275, 283, 289
PostScript ..... 49
Postskriptum ..... 184
Pseudolänge
→ Index der Längen etc. ..... 559
Pseudolängen ..... **171–178**

**R**

Rand ..... **27**, 27, 42, 134, 254, 318
Randnotizen ..... **154–155**, **255**, **319**
Redewendung ..... **124–126**, **316–317**
Rücksendeadresse ..... 215, 220
`ruled` (float-Stil) ..... 145

**S**

Satzspiegel ..... **26**, 27, 30, 42, 46
Schlussgruß ..... 183, **232–234**
Schmutztitel ..... **68**, 69, **308**, 308
Schrift
Art ..... **60–66**, 85–86, 110–112, 126, 128–130, 141, **190–196**, 228, 243–244, 251–253, **270–272**, **304–306**, 318, **463–465**
Größe .... **59**, 110–112, **188–190**, **303–304**
`scrheadings` (Seitenstil) 244, **273**, 275, 283, 284
Seite ..... 27
Seiten
Aufteilung ..... **58–59**, **165–166**
Format ..... **26**
Fuß ..... 42, 240, 241
gerade ..... **83**, **239**, **311**
Kopf ..... 42, 84, 240, 241
Stil ..... **83–93**, 107, **240–246**, **284–292**, **312–313**, 434, **433–454**, **515**
Umbruch ..... 470
ungerade ..... **83**, **239**, **311**
Zahl ..... 85, 89
Seitenstil
Alias- ..... 434, **435–436**
Ebenen- ..... 434, **436–443**
Seitenstile
`@everystyle@` ..... 436, **439**
`empty` .... **84–86**, 87, 91, 92, **241–244**, 245, 246, 312, 313, **439**, 465, 470
`headings` ..... **84–86**, 87, 120, 122, 168, **241–244**, 244, **273**, 273, 275, 284, 342, 446, 452
`letter` ..... 240, 244
`myheadings` **85–86**, 87, 168, **241–244**, 244, 284
`plain` .. **85–86**, 87, 92, **241–244**, 246, **273**, 273, 275, 311, 313
`plain.letter` ..... 240
`plain.scrheadings` .... **273**, 275, 283, 289
`scrheadings` ..... 244, **273**, 275, 283, 284
Serienbriefe ..... **262–266**
Serifen ..... **27**
Signatur ..... 232, 234
Sprachdefinition ..... **343–346**
Sprachen ..... **519–522**
Spruch ..... **124–126**, **316–317**
Stichwortverzeichnis ..... 100
Seitenstil ..... 87
Synchronisation ..... 472

**T**

Tabellen ..... **136–150**
  Nummern ..... 99
  Überschrift ..... 407
  Verzeichnis ..... 99, 151, 154
Tafeln ..... 136
Teil ..... **107**
  Präambel ..... 122
  Seitenstil ..... 87
Telefon ..... 208
Telefonliste ..... **265**
Text
  Auszeichnung . **60–66**, **190–196**, **270–272**, **304–306**, **463–465**
  Bereich ..... 29, 84, 235, 241
Titel ..... **67–73**, 182, 228, **306–311**
  Kopf ..... 67, 74, 307, 311
  Rückseite ..... 72, 310
  Seite ..... 67, 307
  Seitenkopf ..... **70**, **309**
  Seitenstil ..... 87, **311**
  Zeilen ..... 67
Trennlinie ..... 84, 240
Trennzeichen ..... 185, 187, 228
Typisierung ..... **70**, **309**

**U**

Überschriften ..... 102, 114, 120, 122, *siehe auch* Abschnitt, Gliederung, Kapitel
Umgebung
  → Index der Befehle etc. ..... 547
Umschlag ..... 67, 72, 307, 310
Unterschrift ..... 232, 234

**V**

Vakatseite . 85, **90–93**, 101, 103, 104, **245–246**, **312–313**
Variablen ..... **166–171**, **513–515**
  Bezeichnung ..... 166
  Inhalt ..... 166
  Name ..... 166
Verlag ..... **70**, **309**
Versalsatz ..... 287
Versandart ..... 216, 220
Verteiler ..... 185
Verzeichnis ..... **366–417**
  Eintrag ..... 376, **378–394**, 398, 402
Vorspann ..... 99
Vorwort ..... 99

**W**

Widmung ..... 73, 311

**Z**

Zähler
  → Index der Befehle etc. ..... 547
  → Index der Längen etc. ..... 559
Zeilenlänge ..... **28**
Zitat ..... **124–126**, 133, 253, **316–317**
Zusammenfassung ..... **73–74**


## Befehle, Umgebungen und Variablen


`\@addtoplength` ..... **178**
`\@currext` ..... 369, 370, 372, 374, 394, 396
`\@dblarg` ..... 508
`@everystyle@`
  → Allgemeiner Index ..... 544
`\@firstofone` ..... 374
`\@makechapterhead` ..... 412
`\@mkboth` ..... 342, **449–450**, **454**
`\@mkdouble` ..... **449–450**, **454**
`\@mkleft` ..... **449–450**, **454**
`\@mkright` ..... **449–450**, **454**
`\@newplength` ..... **176**
`\@openbib@code` ..... **510–512**
`\@seccntformat` ..... 449, 452
`\@seccntmarkformat` ..... 449, 452
`\@sect` ..... 508
`\@setplength` ..... **178**
`\@ssect` ..... 508
`\@startsection` ..... 508
`\@starttoc` ..... 370, 372, 394, 412
`\@tempskipa`
  → Index der Längen etc. ..... 559
`\@xsect` ..... 508
`\\` ..... 131, 253

* ..... 131, 253

A

abstract (Umgebung) ..... **74**, 122
\abstractname ..... 345
\activateareas ..... **478**, 486
\addchap ..... 74, **113**, 120, 487, 500
\addchap* ..... 74, **113**, 500
\addchapmark ..... **120–121**
\addchaptertocentry ..... **484**
\addcontentsline ..... 371, 390, 482
\addcontentslinetoeachtocfile ..... **372**
\AddLayersAtBeginOfPageStyle ..... **441**
\AddLayersAtEndOfPageStyle ..... **441**
\AddLayersToPageStyle ..... **441**
\AddLayersToPageStyleAfterLayer ..... **441**
\AddLayersToPageStyleBeforeLayer ..... **441**
addmargin (Umgebung) ..... **134–135**, **254**, **318–319**, 471
addmargin* (Umgebung) ..... **134–135**, **254**, **318–319**
\addparagraphtocentry ..... **484**
\addpart ..... **113**, 487, 500
\addpart* ..... **113**, 500
\addparttocentry ..... **484**
\addrchar ..... **265–266**, **296**
\addrentry ..... **263**, **296**, 297–298
\Address ..... **297**, 298
addresseeimage (Variable) ..... **166**, **215–218**
\addsec ..... **113**, 120
\addsec* ..... **113**
\addsecmark ..... **120–121**
\addsectiontocentry ..... **484**
\addsubparagraphtocentry ..... **484**
\addsubsectiontocentry ..... **484**
\addsubsubsectiontocentry ..... **484**
\addtocentrydefault ..... **482–483**, 484
\addtocontents ..... 370
\AddtoDoHook ..... **350–351**
\addtoeachtocfile ..... **370–371**
\AddToHook ..... 355–357
\addtokomafont ... **60–66**, **192–195**, **270–272**, **305**, **463–465**
\addtokomafontgobblelist ..... **485**
\addtokomafontonearglist ..... **485**
\addtokomafontrelaxlist ..... **485**
\AddToLayerPageStyleOptions ..... **442**
\addtolength ..... 171
\addtolengthplength ..... **178**
\AddtoOneTimeDoHook ..... **350–351**
\addtoplength ..... **178**
\addtoreffields ..... **513–514**
\addtotoclist ..... **367–368**
\addxcontentsline ..... **371**, 390
\addxcontentslinetoeachtocfile ..... **372**
\adrchar ..... **265–266**, **296**
\adrentry ..... **262–263**, **296**, 297–298
\AfterAtEndOfClass ..... **356–358**
\AfterAtEndOfClass* ..... **356–358**
\AfterAtEndOfPackage ..... **356–358**
\AfterAtEndOfPackage* ..... **356–358**
\AfterBibliographyPreamble ..... **159**, 511
\AfterCalculatingTypearea ..... 468, **480**
\AfterCalculatingTypearea* ..... **480**
\AfterClass ..... **356–358**
\AfterClass* ..... **356–358**
\AfterFile ..... **355–356**
\AfterPackage ..... **356–358**
\AfterPackage* ..... **356–358**
\AfterReadingMainAux ..... **359–360**
\AfterRestoreareas ..... **479–480**
\AfterRestoreareas* ..... **479–480**
\AfterSelectAnyPageStyle ..... **435**
\AfterSettingArea ..... **480**
\AfterSettingArea* ..... **480**
\AfterStartingTOC ..... **374**
\AfterTOCHead ..... 39, **374**
\aliaskomafont ..... **484**
\alsoname ..... 345
\and ..... **70–72**, **309–310**
\appendix ..... **156**, 509
\appendixmore ..... **509–510**
\appendixname ..... 345
\areaset ..... 38, **46–47**, **478**, 479
\At@startsection ..... **509**
\AtAddToTocList ..... **368–369**
\AtBeginDocument ..... 181
\AtBeginLetter ..... **181–182**
\AtEndBibliography ..... **159**, 511
\AtEndLetter ..... **181–182**
\AtEndOfClass ..... 181, 357

\AtEndOfFamilyOptions ..... 331
\AtEndOfFamilyOptions* ..... 331
\AtEndOfPackage ..... 357
\author ..... 70–72, 309–310
\autodot ..... 117–119
\automark ..... 284–287, 444–445
\automark* ..... 284–287, 444–445

## B

backaddress (Variable) ..... 166, 215–218
backaddressseparator (Variable) ..... 166, 215–218
\backmatter ..... 99–100, 401, 420
\backmatter* ..... 99–100
\bankname ..... 520
\Before@sect ..... 509
\Before@ssect ..... 509
\BeforeClass ..... 356
\BeforeClosingMainAux ..... 359–360
\BeforeFamilyProcessOptions ..... 327–328
\BeforeFile ..... 355–356
\BeforePackage ..... 356
\BeforeRestoreareas ..... 479–480
\BeforeRestoreareas* ..... 479–480
\BeforeSelectAnyPageStyle ..... 435
\BeforeStartingTOC ..... 374
\BeforeTOCHead ..... 374
\bib@beginhook ..... 510–512
\bib@endhook ..... 510–512
\bibname ..... 345
\bigskip ..... 125, 131, 253
\blinddocument ..... 289
\botmark ..... 446, 451
boxed
→ Allgemeiner Index ..... 544
\BreakBibliography ..... 158

## C

\caption .... 137, 141, 400, 405, 407–408, 412
\captionabove ..... 141, 407–408
\captionaboveof ..... 141, 409–410
\captionbelow ..... 141, 407–408
\captionbelowof ..... 141, 409–410
captionbeside (Umgebung) ..... 139, 142–144
\captionformat ..... 145–146
\captionof ..... 141, 409–410, 412
captionofbeside (Umgebung) ..... 145
\captionsamerican ..... 521
\captionsaustralien ..... 521
\captionsaustrian ..... 521
\captionsbritish ..... 521
\captionscanadian ..... 521
\captionscroatian ..... 521
\captionsczech ..... 521
\captionsdutch ..... 521
\captionsenglish ..... 521
\captionsfinnish ..... 521
\captionsfrench ..... 521
\captionsgerman ..... 521
\captionsitalian ..... 521
\captionsnaustrian ..... 521
\captionsnewzealand ..... 521
\captionsngerman ..... 521
\captionsnorsk ..... 521
\captionsnswissgerman ..... 521
\captionspolish ..... 521
\captionsslovak ..... 521
\captionsspanish ..... 521
\captionsswedish ..... 521
\captionsswissgerman ..... 521
\captionsUKenglish ..... 521
\captionsUSenglish ..... 521
\cc ..... 185–186
\ccname ..... 345, 520
ccseparator (Variable) ..... 167, 185–186
\cefoot ..... 277–280
\cefoot* ..... 280–281
\cehead ..... 274–276
\cehead* ..... 277
\centering ..... 98, 250, 316
\cfoot ..... 281–282
\cfoot* ..... 282–283
\chapapp ..... 119
\chapappifchapterprefix ..... 119
chapter
→ Index der Längen etc. ..... 559
\chapter . 78, 107–112, 113, 121, 401, 412, 487, 500
\chapter* ..... 74, 112–113, 292, 500
\chapterformat ..... 104, 105, 117–119
\chapterheadendvskip ..... 500–501

\chapterheadmidvskip ..... 500–501
\chapterheadstartvskip ..... 500–501
\chapterlinesformat ..... 119, 503–505
\chapterlineswithprefixformat ..... 119, 503–505
\chaptermark ..... 120–121, 122, 290, 448
\chaptermarkformat . 120–121, 290, 447, 453
\chaptername ..... 345
\chapternumdepth ..... 121
\chapterpagestyle ..... 87–89
\chead ..... 281–282
\chead* ..... 282–283
\ClassInfoNoLine ..... 347
\ClassName ..... 482
\Clause ..... 320
\cleardoubleemptypage 92–93, 246, 312–313
\cleardoubleevenemptypage ..... 92–93, 246, 312–313
\cleardoubleevenpage . 92–93, 246, 312–313
\cleardoubleevenpageusingstyle ..... 92–93, 246, 312–313
\cleardoubleevenplainpage ..... 92–93, 246, 312–313
\cleardoubleevenstandardpage .. 92–93, 246, 312–313
\cleardoubleoddemptypage ..... 92–93, 246, 312–313
\cleardoubleoddpage ..... 89, 92–93, 246, 312–313, 480
\cleardoubleoddpageusingstyle 92–93, 246, 312–313
\cleardoubleoddplainpage ..... 92–93, 246, 312–313
\cleardoubleoddstandardpage ... 92–93, 246, 312–313
\cleardoublepage . 92–93, 246, 312–313, 480
\cleardoublepageusingstyle .... 92–93, 246, 312–313
\cleardoubleplainpage 92–93, 246, 312–313
\cleardoublestandardpage ..... 92–93, 246, 312–313
\clearmainofpairofpagestyles ..... 456
\clearnotecolumn ..... 474
\clearnotecolumns ..... 475
\clearpage ..... 92–93, 246, 312–313
\clearpairofpagestyles ..... 289, 456
\clearplainofpairofpagestyles ..... 456
\CloneTOCEntryStyle ..... 392
\closing ..... 179, 183–184, 232
\cofoot ..... 277–280
\cofoot* ..... 280–281
\cohead ..... 274–276
\cohead* ..... 277
\Comment ..... 297, 298
\contentsline ..... 390
\contentsname ..... 346
\coverpagebottommargin ..... 67, 307
\coverpageleftmargin ..... 67, 307
\coverpagerightmargin ..... 67, 307
\coverpagetopmargin ..... 67, 307
\currentpagestyle ..... 434–435
customer (Variable) ..... 167, 224–226
\customername ..... 520

## D

\date ..... 70–72, 223, 309–310
date (Variable) ..... 167, 223–224, 225
\dateamerican ..... 522
\dateaustralien ..... 522
\dateaustrian ..... 522
\datebritish ..... 522
\datecanadian ..... 522
\datecroatian ..... 522
\dateczech ..... 522
\datedutch ..... 522
\dateenglish ..... 522
\datefinnish ..... 522
\datefrench ..... 522
\dategerman ..... 522
\dateitalian ..... 522
\datename ..... 520
\datenaustrian ..... 522
\datenewzealand ..... 522
\datengerman ..... 522
\datenorsk ..... 522
\datenswissgerman ..... 522
\datepolish ..... 522
\dateslovak ..... 522
\datespanish ..... 522
\dateswedish ..... 522
\dateswissgerman ..... 522

\dateUKenglish ..... 522
\dateUSenglish ..... 522
\DeclareLayer ..... 423–430
\DeclareNewLayer ..... 423–430
\DeclareNewNoteColumn ..... 466–469
\DeclareNewPageStyleAlias ..... 435
\DeclareNewPageStyleByLayers 432, 436–439
\DeclareNewSectionCommand 75, 382, 487–499
\DeclareNewSectionCommands ..... 499
\DeclareNewTOC ..... 382, 393, 400–406
\DeclareNoteColumn ..... 466–469
\DeclareOption ..... 326
\DeclarePageStyleAlias ..... 435
\DeclarePageStyleByLayers .... 436–439, 457
\DeclareSectionCommand ... 75, 382, 487–499, 500, 501
\DeclareSectionCommands ..... 499
\DeclareSectionNumberDepth ..... 421
\DeclareTOCEntryStyle ..... 390–392, 393
\DeclareTOCStyleEntries ..... 379–390
\DeclareTOCStyleEntry ..... 75, 379–390, 400
\dedication ..... 73, 311
\defaultreffields ..... 513–514
\defcaptionname ..... 344–346
\defcaptionname* ..... 344–346
\deffootnote ..... 96–98, 249, 315–316, 316
\deffootnotemark ..... 96–98, 249, 315–316
\DefineFamily ..... 323–324
\DefineFamilyKey ..... 324–326
\DefineFamilyMember ..... 323–324
\DefineTOCEntryBooleanOption ..... 390–392
\DefineTOCEntryCommandOption ..... 390–392
\DefineTOCEntryIfOption ..... 390–392
\DefineTOCEntryLengthOption ..... 390–392
\DefineTOCEntryNumberOption ..... 390–392
\DefineTOCEntryOption ..... 390–392
\defpagestyle ..... 457–459
\defpairofpagestyles ..... 455–456
\deftocheading ..... 375, 377
description (Umgebung) ..... 129–130, 252
\DestroyLayer ..... 433
\DestroyPageStyleAlias ..... 436
\DestroyRealLayerPageStyle ..... 443
\dictum ..... 123, 124–126, 317
\dictumauthorformat ..... 124–126, 317
\dictumrule ..... 124–126, 317
\dictumwidth ..... 124–126, 317
\dimexpr ..... 340
displaymath (Umgebung) ..... 136
\documentclass .. 31, 55, 162–163, 267–268, 301, 358, 418–419, 462
\doforeachtocfile ..... 369–370
\dospecials ..... 471

## E

\edgesize ..... 517
\emailname ..... 520
emailseparator (Variable) ..... 167, 208–211
empty
→ Allgemeiner Index ..... 544
\encl ..... 187–188
\enclname ..... 346, 520
enclseparator (Variable) ..... 167, 187–188
\enlargethispage ..... 235, 237
enumerate (Umgebung) ..... 128–129, 252
eqnarray (Umgebung) ..... 136
equation (Umgebung) ..... 136
\evensidemargin
→ Index der Längen etc. ..... 559
\ExecuteDoHook ..... 349–350
\ExpandArgsCS ..... 348
\extratitle ..... 69, 308–309

## F

\FamilyBoolKey ..... 332–333
\FamilyCounterKey ..... 335
\FamilyCounterMacroKey ..... 336
\FamilyCSKey ..... 336–337
\FamilyExecuteOptions ..... 328, 331
\FamilyInverseBoolKey ..... 333
\FamilyKeyState ..... 324–326
\FamilyKeyStateNeedValue ..... 324–326
\FamilyKeyStateProcessed ..... 324–326
\FamilyKeyStateUnknown ..... 324–326
\FamilyKeyStateUnknownValue ..... 324–326
\FamilyLengthKey ..... 336
\FamilyLengthMacroKey ..... 336
\FamilyNumericalKey ..... 334–335
\FamilyOption ..... 330–331
\FamilyOptions ..... 329–330, 331
\FamilyProcessOptions ..... 326–327, 331

\FamilySetBool ..... 332–333
\FamilySetCounter ..... 335
\FamilySetCounterMacro ..... 336
\FamilySetInverseBool ..... 333
\FamilySetLength ..... 336
\FamilySetLengthMacro ..... 336
\FamilySetNumerical ..... 334–335
\FamilySetUseLengthMacro ..... 336
\FamilyStringKey ..... 336–337
\FamilyUnknownKeyValue ..... 337–339
\FamilyUseLengthMacroKey ..... 336
\faxname ..... 520
faxseparator (Variable) ..... 167, 208–211
figure (Umgebung) ..... 137
figure* (Umgebung) ..... 137
figure- (Umgebung) ..... 137
\figureformat ..... 146
\figurename ..... 346
firstfoot (Variable) ..... 167, 235–236
firsthead (Variable) ..... 167, 213
\firstmark ..... 446, 451
\FirstName ..... 297, 298
\flushbottom ..... 28, 58–59, 82, 166, 181
\footheight
→ Index der Längen etc. ..... 559
\footnote ..... 94, 94–95, 247, 248, 314, 314
\footnotemark .. 94, 94–95, 247, 248, 314, 314
\footnotetext ..... 94–95, 248, 314
\footref ..... 96, 248, 315
\footskip
→ Index der Längen etc. ..... 559
\ForDoHook ..... 351
\foreachemptykomavar ..... 515
\foreachkomavar ..... 515
\foreachkomavarifempty ..... 515
\ForEachLayerOfPageStyle ..... 440–441
\ForEachLayerOfPageStyle* ..... 440–441
\foreachnonemptykomavar ..... 515
\FreeI ..... 297, 298
\FreeII ..... 297, 298
\FreeIII ..... 297, 298
\FreeIV ..... 297, 298
fromaddress (Variable) ..... 167, 203–206
frombank (Variable) ..... 167, 235–237
fromemail (Variable) ..... 167, 208–211
fromfax (Variable) ..... 167, 208–211
fromlogo (Variable) ..... 167, 211–213
frommobilephone (Variable) .... 167, 208–211
fromname (Variable) ... 168, 203–206, 209, 242
fromphone (Variable) ..... 168, 208–211
fromurl (Variable) ..... 168, 208–211
fromzipcode (Variable) ..... 168, 215–218
\frontispiece ..... 69, 308–309
\frontmatter ..... 90, 99–100, 401, 420
\frontmatter* ..... 99–100

## G

\g@addto@macro ..... 395
\GenericMarkFormat ..... 449, 452–454
\GetLayerContents ..... 432
\GetRealPageStyle ..... 436
\glossaryname ..... 346
\glueexpr ..... 341

## H

\headfromname ..... 520
\headheight
→ Index der Längen etc. ..... 559
headings
→ Allgemeiner Index ..... 544
\headmark ..... 288–289, 447
\headtoname ..... 346, 520

## I

\if@atdocument ..... 343
\if@chapter ..... 420
\if@dvips ..... 340
\if@mainmatter ..... 420
\IfActiveMkBoth ..... 342
\IfArgIsEmpty ..... 339, 483
\Ifattoclist ..... 366–367
\ifcase ..... 334
\IfChapterUsesPrefixLine ..... 101–102
\Ifdimen ..... 340
\Ifdvioutput ..... 340
\IfExistskomafont ..... 486
\iffalse ..... 420
\IfIsAliaskomafont ..... 486
\Ifiscount ..... 341
\Ifiscounter ..... 341
\Ifisdimen ..... 340

\Ifisdimension ..... 340
\Ifisdimexpr ..... 340
\Ifisglue ..... 341
\Ifisglueexpr ..... 341
\Ifisinteger ..... 341
\Ifisnumexpr ..... 341
\Ifisskip ..... 340
\Ifkomavar ..... 171
\Ifkomavarempty ..... 171, 514–515
\Ifkomavarempty* ..... 171, 514–515
\IfLayerAtPageStyle ..... 443
\IfLayerExists ..... 433
\IfLayerPageStyleExists ..... 442
\IfLayersAtPageStyle ..... 443
\Ifnotundefined ..... 339
\Ifnumber ..... 341
\Ifnumbered ..... 122
\ifoot ..... 281–282
\ifoot* ..... 282–283
\Ifpdfoutput ..... 339
\Ifplength ..... 176
\Ifpsoutput ..... 340
\IfRealLayerPageStyleExists ..... 442
\IfSectionCommandStyleIs ..... 499–500
\IfSomeLayersAtPageStyle ..... 443
\Ifstr ..... 338, 339
\Ifstrstart ..... 339
\Ifthispageodd ..... 83, 239, 311
\Iftocfeature ..... 378
\iftrue ..... 420
\Ifundefinedorrelax ..... 339
\Ifunnumbered ..... 122
\IfUseNumber ..... 508
\IfUsePrefixLine ..... 117–119
\ihead ..... 281–282, 289
\ihead* ..... 282–283
\indexname ..... 346
\indexpagestyle ..... 87–89, 160
\InputAddressFile ..... 296, 297
\interlinepenalty ..... 132
invoice (Variable) ..... 168, 224–226
\invoicename ..... 520
\item ..... 126–131, 251–253, 318
itemize (Umgebung) ..... 126–128, 251

## K

komaabove
→ Allgemeiner Index ..... 544
komabelow
→ Allgemeiner Index ..... 544
\KOMAClassName ..... 482
\KOMAoption .. 31–32, 56, 163, 268, 301–302, 419, 463
\KOMAoptions . 31–32, 56, 163, 268, 301–302, 400, 419, 463
\KOMAScript ..... 321
\KOMAScriptVersion ..... 347

## L

\l@addto@macro ..... 348
\label ..... 96, 182, 248, 315
\labelenumi ..... 128–129, 252
\labelenumii ..... 128–129, 252
\labelenumiii ..... 128–129, 252
\labelenumiv ..... 128–129, 252
labeling (Umgebung) ..... 130–131, 253, 318
\labelitemi ..... 126–128, 251
\labelitemii ..... 126–128, 251
\labelitemiii ..... 126–128, 251
\labelitemiv ..... 126–128, 251
\LastName ..... 297, 298
\lastpenalty ..... 393
\LastTOCLevelWasHigher ..... 393
\LastTOCLevelWasLower ..... 393
\LastTOCLevelWasSame ..... 393
\layercontentsmeasure ..... 433
\layerhalign ..... 431
\layerheight ..... 431, 459
\layervalign ..... 431
\layerwidth ..... 431, 459
\layerxoffset ..... 431
\layeryoffset ..... 431
\lefoot ..... 277–280
\lefoot* ..... 280–281
\leftbotmark ..... 446–447, 451–452
\leftfirstmark ..... 446–447, 451–452
\leftmark ..... 288–289, 291, 446–447, 448, 451–452
\lefttopmark ..... 446–447, 451–452
\lehead ..... 274–276
\lehead* ..... 277

\LenToUnit ..... 431
letter
→ Index der Längen etc. ..... 559
letter
→ Allgemeiner Index ..... 544
letter (Umgebung) ..... 180–181
\letterlastpage ..... 182
\LetterOptionNeedsPapersize ..... 516
\letterpagestyle ..... 240
\linespread ..... 28, 39
\lipsum ..... 276
\listfigurename ..... 346
\listof*Dateierweiterung*name ..... 372–374
\listofeachtoc ..... 372–374
\listoffigures ..... 154, 412
\listoftables ..... 154, 412
\listoftoc ..... 372–374
\listoftoc* ..... 372–374
\listtablename ..... 346
\LoadLetterOption ..... 256–262
\LoadLetterOptions ..... 256–262
location (Variable) ..... 168, 221–222
\lofoot ..... 277–280
\lofoot* ..... 280–281
\lohead ..... 274–276
\lohead* ..... 277
\lowertitleback ..... 72, 310

## M

\mainmatter ..... 90, 99–100, 420
\mainmatter* ..... 99–100
\makeatletter ..... 397
\makeatother ..... 397
\MakeMarkcase ..... 86, 287, 292, 374–375, 445–446
\makenote ..... 470–471, 472
\makenote* ..... 470–471
\maketitle ..... 67, 68–73, 307, 308–311
\MakeUppercase ..... 287, 374, 406, 453, 514
\manualmark ..... 284–287, 444–445
\marginline ..... 155, 255, 319
\marginpar ..... 155, 255, 319
\marginparsep
→ Index der Längen etc. ..... 559
\marginparwidth
→ Index der Längen etc. ..... 559
\markboth ..... 85, 87, 243, 244, 244–245, 291–292, 342, 448–449, 514
\markdouble ..... 243, 244, 291–292, 448–449
\markleft 87, 243, 244, 291–292, 448–449, 514
\markright ..... 85, 87, 243, 244, 244–245, 291–292, 446, 448–449, 452, 514
\maxdimen
→ Index der Längen etc. ..... 559
\maybesffamily ... 60, 110, 124, 129, 191, 216, 217, 226, 228, 252, 304–305
\mediaheight
→ Index der Längen etc. ..... 559
\mediawidth
→ Index der Längen etc. ..... 559
\medskip ..... 131, 253
\microtypesetup ..... 376
\minisec ..... 114–115
\mobilephonename ..... 520
mobilephoneseparator (Variable) ..... 168, 208–211
\ModifyLayer ..... 423–430
\ModifyLayerPageStyleOptions ..... 442
\ModifyLayers ..... 430
\multfootsep ... 94, 94, 95, 247, 248, 314, 314
\multiplefootnoteseparator 94–95, 248, 314
myheadings
→ Allgemeiner Index ..... 544
myref (Variable) ..... 168, 224–226
\myrefname ..... 520

## N

\Name ..... 297, 298
\newbibstyle ..... 157, 510–512
\newblock ..... 156, 158, 510–512
\newcaptionname ..... 344–346
\newcaptionname* ..... 344–346
\newcommand* ..... 480
\newkomafont ..... 484
\newkomavar ..... 513–514
\newkomavar* ..... 513–514
\newlength ..... 171
\newpage ..... 90
\newpagestyle ..... 457–459
\newpairofpagestyles ..... 455–456
\newplength ..... 176
nextfoot (Variable) ... 168, 243, 244, 244–245

nexthead (Variable) ..... **168**, 243, **244–245**
\nobreakspace ..... 95
\nofiles ..... 470
\noindent ..... 74, 133, 253
\nonumberline ..... **396**
\normalfont ..... 271, 272
\normalmarginpar ..... 466
\numberline ..... **379**, 383
\numexpr ..... 341, 348

**O**

\ofoot ..... **281–282**
\ofoot* ..... **282–283**
\ohead ..... **281–282**
\ohead* ..... **282–283**, 289
\onecolumn ..... 377
\opening 179, **182–183**, 201, 222, 226, 234, 241, 243, 257, 516
\othersectionlevelsformat ..... **117–119**

**P**

\PackageInfoNoLine ..... **347**
\pagebreak ..... 90
\pageheight
→ Index der Längen etc. ..... 559
\pagemark ..... 86, **288–289**, **447**
\pagename ..... 346, **520**
\pagenumbering ..... **89–90**
\pageref ..... 96, 182
\pagestyle .... **84–86**, **241–244**, 434, 435, 455
\pagewidth
→ Index der Längen etc. ..... 559
\paperheight
→ Index der Längen etc. ..... 559
\paperwidth
→ Index der Längen etc. ..... 559
\paragraph ..... **107–112**, 488
\paragraph* ..... **112–113**
\paragraphformat ..... **117–119**
\paragraphmark ..... **290**, **448**
\paragraphmarkformat ..... **290**, **447**
\paragraphnumdepth ..... **121**
\paragraphtocdepth ..... **79–80**
\parbox ..... 124, 317
\parfillskip
→ Index der Längen etc. ..... 559
\parindent
→ Index der Längen etc. ..... 559
\parskip
→ Index der Längen etc. ..... 559
\part ..... 78, **107–112**, 113, 121, 487, 500
\part* ..... **112–113**, 500
\partformat ..... **117–119**
\partheademptypage ..... **500–501**
\partheadendvskip ..... **500–501**
\partheadmidvskip ..... **500–501**
\partheadstartvskip ..... **500–501**
\partlineswithprefixformat ... 119, **502–503**
\partmark ..... **290**, **448**
\partmarkformat ..... **290**, **447**
\partname ..... 346
\partnumdepth ..... 80, **121**
\partpagestyle ..... **87–89**
\parttocdepth ..... **79–80**
\pdflastypos ..... 470
\pdfpageheight
→ Index der Längen etc. ..... 559
\pdfpagewidth
→ Index der Längen etc. ..... 559
\pdfsavepos ..... 470
\phonename ..... **520**
phoneseparator (Variable) ..... **168**, **208–211**
picture (Umgebung) ..... 422
place (Variable) ..... **168**, **215–218**, 225
placeseparator (Variable) . **168**, 224, **225–226**
plain
→ Allgemeiner Index ..... 544
plain.letter
→ Allgemeiner Index ..... 544
plain.scrheadings
→ Allgemeiner Index ..... 544
PPcode (Variable) ..... **169**, **215–218**
PPdatamatrix (Variable) ..... **169**, **215–218**
\prefacename ..... 346
\PreventPackageFromLoading ... **362–363**, 363
\PreventPackageFromLoading* ..... **362–363**
\ProcessOptions* ..... 326
\proofname ..... 346
\protect ..... 96, 453, 483, 514
\providecaptionname ..... **344–346**
\providecaptionname* ..... **344–346**

\ProvideLayer ..... 423–430
\ProvideNoteColumn ..... 466–469
\providepagestyle ..... 457–459
\ProvidePageStyleAlias ..... 435
\ProvidePageStyleByLayers ..... 436–439
\providepairofpagestyles ..... 455–456
\ProvideSectionCommand .... 75, 382, 487–499
\ProvideSectionCommands ..... 499
\ps ..... 184–185
\publishers ..... 70–72, 309–310
\putC ..... 431–432
\putLL ..... 431–432
\putLR ..... 431–432
\putUL ..... 431–432
\putUR ..... 431–432

## Q

quotation (Umgebung) ..... 74, 133–134, 253
quote (Umgebung) ..... 133–134, 253

## R

\raggedbottom ..... 28, 58–59, 166
\raggedchapter ..... 115–116
\raggedchapterentry ..... 482
\raggeddictum ..... 124–126, 317
\raggeddictumauthor ..... 124–126, 317
\raggeddictumtext ..... 124–126, 317
\raggedfootnote ..... 98, 250, 316
\raggedleft ..... 98, 124, 250, 316, 317
\raggedpart ..... 115–116
\raggedright ..... 98, 124, 250, 316, 317
\raggedsection ..... 115–116
\raggedsignature ..... 232–234
\raisebox ..... 143
\recalctypearea ..... 40, 59, 189, 304
\RedeclareLayer ..... 423–430
\RedeclareNoteColumn ..... 466–469
\RedeclarePageStyleAlias ..... 435
\RedeclarePageStyleByLayers ..... 436–439
\RedeclareSectionCommand . 75, 382, 487–499
\RedeclareSectionCommands ..... 499
\ref ..... 96
\refname ..... 346
\refoot ..... 277–280
\refoot* ..... 280–281
\rehead ..... 274–276
\rehead* ..... 277
\relax ..... 98, 250, 316, 484
\RelaxFamilyKey ..... 326
\removefromtoclist ..... 369
\RemoveLayersFromPageStyle ..... 441
\removereffields ..... 513–514
\renewcaptionname ..... 344–346
\renewcaptionname* ..... 344–346
\renewcommand ..... 511
\renewpagestyle ..... 457–459
\renewpairofpagestyles ..... 455–456
\ReplaceClass ..... 361–362
\ReplaceInput ..... 360–361
\ReplacePackage ..... 361–362
\RequirePackage ..... 323, 358, 361–363
\RequirePackageWithOptions ..... 323, 362
\ResetPreventPackageFromLoading . 363–364
\restylefloat ..... 138
\reversemarginpar ..... 466
\rightbotmark ..... 446–447, 451–452
\rightfirstmark ..... 446–447, 451–452
\rightmark ..... 288–289, 291, 446–447, 448, 451–452
\righttopmark ..... 446–447, 451–452
\rofoot ..... 277–280
\rofoot* ..... 280–281
\rohead ..... 274–276
\rohead* ..... 277
ruled
→ Allgemeiner Index ..... 544

## S

\scr@startsection ..... 508–509
scrheadings
→ Allgemeiner Index ..... 544
\scrlayerAddCsToInterface ..... 450
\scrlayerAddToInterface ..... 450
\scrlayerInitInterface ..... 450
\scrlayerOnAutoRemoveInterface ..... 450
\SecDef ..... 508–509
\secdef ..... 508
secnumdepth
→ Index der Längen etc. ..... 559
section
→ Index der Längen etc. ..... 559
\section ..... 78, 107–112, 113, 121, 401, 488

\section* ..... 112–113
\sectioncatchphraseformat .... 119, 505–506
\sectionformat ..... 117–119
\sectionlinesformat ..... 119, 505–506
\sectionmark ..... 120–121, 290, 448
\sectionmarkformat . 120–121, 290, 447, 452, 453
\sectionnumdepth ..... 121
\sectiontocdepth ..... 79–80
\seename ..... 346
\selectfont ..... 81, 238
\setbibpreamble ..... 157–158
\setcapdynwidth ..... 148
\setcaphanging ..... 146–147
\setcapindent ..... 146–147
\setcapindent* ..... 146–147
\setcapmargin ..... 148
\setcapmargin* ..... 148
\setcaptionalignment ..... 149–150
\setcapwidth ..... 148
\setchapterpreamble ..... 122–123
\setfootnoterule ..... 98, 250
\setindexpreamble ..... 160–161
\setkomafont 60–66, 192–195, 270–272, 305, 463–465
\setkomavar ..... 170
\setkomavar* ..... 170
\setlength ..... 171
\setlengthtoplength ..... 178
\setparsizes ..... 486–487
\setpartpreamble ..... 122–123
\setplength ..... 178
\setplengthtodepth ..... 178
\setplengthtoheight ..... 178
\setplengthtototalheight ..... 178
\setplengthtowidth ..... 178
\setshowstyle ..... 517
\setuptoc ..... 375–378
\showenvelope ..... 518–519
\showfields ..... 517
\showISOenvelope ..... 518–519
\showUScheck ..... 518–519
\showUScommercial ..... 518–519
\sigalign ..... 232–234
signature (Variable) ..... 169, 232
\sigspace ..... 232
specialmail (Variable) ..... 169, 215–218
\SplitDoHook ..... 352
\storeareas ..... 479–480
\StorePreventPackageFromLoading . 363–364
\SubClause ..... 320
\subject ..... 70–72, 309–310
subject (Variable) ..... 169, 228–230, 242
\subjectname ..... 520
subjectseparator (Variable) ... 169, 228–230
\subparagraph ..... 107–112, 488
\subparagraph* ..... 112–113
\subparagraphformat ..... 117–119
\subparagraphmark ..... 290, 448
\subparagraphmarkformat ..... 290, 447
\subparagraphnumdepth ..... 121
\subparagraphtocdepth ..... 79–80
\subsection ..... 107–112, 121, 488
\subsection* ..... 112–113
\subsectionformat ..... 117–119
\subsectionmark ..... 120–121, 290, 448
\subsectionmarkformat .. 120–121, 290, 447, 453
\subsectionnumdepth ..... 121
\subsectiontocdepth ..... 79–80
\subsubsection ..... 107–112, 121, 488
\subsubsection* ..... 112–113
\subsubsectionformat ..... 117–119
\subsubsectionmark ..... 290, 448
\subsubsectionmarkformat ..... 290, 447
\subsubsectionnumdepth ..... 121
\subsubsectiontocdepth ..... 79–80
\subtitle ..... 70–72, 309–310
\syncwithnotecolumn ..... 472–473
\syncwithnotecolumns ..... 474

## T

table (Umgebung) ..... 136, 137
table* (Umgebung) ..... 137
table- (Umgebung) ..... 137, 410
\tableformat ..... 146
\tablename ..... 346
\tableofcontents ..... 78–79, 292
tabular (Umgebung) ..... 136
\tb@*Dateierweiterung*@after@hook ..... 395
\tb@*Dateierweiterung*@before@hook ..... 395

\Telephone ..... 297
\textheight
→ Index der Längen etc. ..... 559
\textmaybesf ..... 60, 191, 304–305
\textwidth
→ Index der Längen etc. ..... 559
\thanks ..... 70–72, 309–310
\theenumi ..... 128–129, 252
\theenumii ..... 128–129, 252
\theenumiii ..... 128–129, 252
\theenumiv ..... 128–129, 252
\thefootnotemark ..... 96–98, 249, 315–316
\thepage ..... 86
\thisletter ..... 182
\thispagestyle .... 84–86, 241–244, 311, 434
\title ..... 70–72, 309–310
title (Variable) ..... 169, 228
\titlehead ..... 70–72, 309–310
titlepage (Umgebung) ..... 68, 307–308
\titlepagestyle ..... 87–89, 276, 311
toaddress (Variable) ..... 169, 215–218
\tocbasic@@after@hook ..... 395
\tocbasic@@before@hook ..... 395
\tocbasic@addxcontentsline ..... 396
\tocbasic@DependOnPenaltyAndTOCLevel
..... 396–397
\tocbasic@extend@babel ..... 394
\tocbasic@listhead ..... 396
\tocbasic@listhead@*Dateierweiterung* . 396
\tocbasic@SetPenaltyByTOCLevel ... 396–397
\tocbasic@starttoc ..... 394–395
\tocbasicautomode ..... 370
tocdepth
→ Index der Längen etc. ..... 559
\TOCEntryStyleInitCode ..... 393
\TOCEntryStyleStartInitCode ..... 393
\TOCLineLeaderFill ..... 394
\today ..... 70, 223, 223, 309
toname (Variable) ..... 169, 215–218
\toplevelpagestyle ..... 434–435
\topmargin
→ Index der Längen etc. ..... 559
\topmark ..... 446, 452
\typearea ..... 40

## U

\UnifyLayersAtPageStyle ..... 442
\unitfactor ..... 518–519
\UnPreventPackageFromLoading ..... 364
\UnPreventPackageFromLoading* ..... 364
\UnReplaceClass ..... 362
\UnReplaceInput ..... 362
\UnReplacePackage ..... 362
\unsettoc ..... 375–378
\uppertitleback ..... 72, 310
urlseparator (Variable) ..... 208–211
\useencodingofkomafont .. 66, 196, 272, 306, 465, 485
\usefamilyofkomafont .... 66, 196, 272, 306, 465, 485
\usefontofkomafont .. 66, 196, 272, 306, 465, 485
\usekomafont 60–66, 192–195, 270–272, 305, 463–465, 484
\usekomavar ..... 171, 514
\usekomavar* ..... 171, 514
\usepackage . 31, 55, 162–163, 267–268, 301, 323, 362, 418–419, 462
\useplength ..... 176
\useseriesofkomafont .... 66, 196, 272, 306, 465, 485
\useshapeofkomafont 66, 196, 272, 306, 465, 485
\usesizeofkomafont .. 66, 196, 272, 306, 465, 485
\usetocbasicnumberline ..... 379

## V

verse (Umgebung) ..... 131–133, 253
\vspace ..... 472

## W

\wwwname ..... 520

## X

\XdivY ..... 348
\XmodY ..... 348

## Y

yourmail (Variable) ..... 169, 224–226
\yourmailname ..... 520

yourref (Variable) ..... 169, 224–226
\yourrefname ..... 520

## Längen und Zähler

\@tempskipa (Länge) ..... 500, 501

B

backaddrheight ..... 172, 220
bfoldmarklength ..... 172, 200
bfoldmarkvpos ..... 172, 200

C

chapter (Zähler) ..... 401

E

\evensidemargin (Länge) ..... 67, 307

F

firstfoothpos ..... 172, 237
firstfootvpos ..... 172, 237
firstfootwidth ..... 173, 237
firstheadhpos ..... 173, 202
firstheadvpos ..... 173, 202
firstheadwidth ..... 173, 202, 237
foldmarkhpos ..... 173, 201
foldmarkthickness ..... 201
foldmarkvpos ..... 173, 201
\footheight (Länge) ..... 45–46, 269, 444
\footskip (Länge) ..... 237, 269
fromrulethickness ..... 173, 207
fromrulewidth ..... 173, 203, 207

H

\headheight (Länge) ..... 269, 444

L

letter (Zähler) ..... 182
lfoldmarkhpos ..... 173, 201
lfoldmarklength ..... 173, 201
locheight ..... 173, 222
lochpos ..... 174, 222
locvpos ..... 174, 222
locwidth ..... 174, 222

M

\marginparsep (Länge) ..... 468
\marginparwidth (Länge) ..... 44
\maxdimen (Länge) ..... 202, 237
\mediaheight (Länge) ..... 51
\mediawidth (Länge) ..... 51
mfoldmarklength ..... 174, 200
mfoldmarkvpos ..... 174, 200

P

\pageheight (Länge) ..... 51
\pagewidth (Länge) ..... 51
\paperheight (Länge) ..... 237
\paperwidth (Länge) ..... 202, 237
\parfillskip (Länge) ..... 394
\parindent (Länge) ..... 394
\parskip (Länge) ..... 394, 486
\pdfpageheight (Länge) ..... 51, 470
\pdfpagewidth (Länge) ..... 51
pfoldmarklength ..... 174, 200
PPdatamatrixvskip ..... 174, 220
PPheadheight ..... 174, 220
PPheadwidth ..... 174, 220

R

refaftervskip ..... 174, 227
refhpos ..... 174, 226–227
refvpos ..... 175, 226
refwidth ..... 175, 226–227

S

secnumdepth (Zähler) ..... 80, 121
section (Zähler) ..... 401
sigbeforevskip ..... 175, 232, 234
sigindent ..... 175, 232, 234
specialmailindent ..... 175, 220
specialmailrightindent ..... 175, 220
subjectaftervskip ..... 175, 231–232
subjectbeforevskip ..... 175, 231–232
subjectvpos ..... 175, 230–231

T

\textheight (Länge) ..... 38

Z

zipcodeseparator (Variable) ... 169, 215–218

\textwidth (Länge) ..... 38, 44
tfoldmarklength ..... **175**, **200**
tfoldmarkvpos ..... **175**, **200**
toaddrheight ..... **175**, **219**
toaddrhpos ..... **176**, **218–219**, 258
toaddrindent ..... **176**, **219**
toaddrvpos ..... **176**, **218–219**
toaddrwidth ..... **176**, **219**, 259
tocdepth (Zähler) .. **79–80**, **378–379**, 380, 386, 391
\topmargin (Länge) ..... 67, 307

## Elemente mit der Möglichkeit zur Schriftumschaltung

### A

addressee ..... **192**, 215, **215**, 217
author ..... **61**, **70**, **309**

### B

backaddress ..... **192**, **216**, 217

### C

caption ..... **61**, **141**
captionlabel ..... **61**, **141**
chapter ..... **61**, 102, 111, **110–112**
chapterentry ..... **61**, **79**
chapterentrydots ..... **61**, **78**, 79
chapterentrypagenumber ..... **61**, **79**
chapterprefix ..... **62**, **102**, 111, **110–112**

### D

date ..... **62**, **70**, **309**
dedication ..... **62**, **73**, **311**
descriptionlabel ..... **62**, **129–130**, **192**, **252**
dictum ..... **62**, **124**, 125, **317**, 317
dictumauthor ..... **62**, **125**, **317**
dictumtext ..... **62**
disposition .. **62**, 102, 111, 112, **110–112**, 116

### E

enumeratelabel ..... **62**, **128–129**, **193**, **252**

### F

field ..... **517**
foldmark ..... **193**, **199–200**
footbotline ..... **271**, **293**
footnote ..... **62**, **97–98**, **193**, **249**, **315–316**
footnotelabel **62**, **97–98**, **193**, **249**, **315–316**
footnotereference .. **63**, 97, **97–98**, **193**, 249, **249**, 316, **316**
footnoterule ..... **63**, **98**, **193**, **250**
footsepline ..... **271**, **293**
fromaddress ..... **193**, 204, **204**
fromname ..... **193**, 204, **204**
fromrule ..... **193**, 204, **204**

### G

*Gliederungsebene* ..... **488**
*Gliederungsebene*prefix ..... **488**

### H

headsepline ..... **271**, **293**
headtopline ..... **271**, **293**

### I

itemizelabel ..... **63**, **126**, **193**, **251**

### L

labelenumi ..... **63**, **128–129**, **193**, **252**
labelenumii ..... **63**, **128–129**, **194**, **252**
labelenumiii ..... **63**, **128–129**, **194**, **252**
labelenumiv ..... **63**, **128–129**, **194**, **252**
labelinglabel ... **63**, 130, **130–131**, **194**, 253, **253**, 318, **318**
labelingseparator ... **63**, 130, **130–131**, **194**, 253, **253**, 318, **318**
labelitemi ..... **63**, **126**, **194**, **251**
labelitemii ..... **64**, **126**, **194**, **251**
labelitemiii ..... **64**, **126**, **194**, **251**
labelitemiv ..... **64**, **126**, **194**, **251**
letter ..... **518**
lettersubject ..... **195**, **228–229**
lettertitle ..... **195**, **228**

### M

measure ..... **518**
minisec ..... **64**, **114**

### N

notecolumn.marginpar ..... **466**
notecolumn.*Name der Notizspalte* ..... **466**

### P

pagefoot .... **64**, 85, 86, **85–86**, **194**, **243–244**, **271**, 271, **278**
pagehead **64**, **195**, **243–244**, 271, **272**, **274**, 279
pageheadfoot ..... **64**, 85, 86, **85–86**, **195**, **243–244**, 271, **272**, 272, **274**, **278**, 279
pagenumber . **64**, 85, 86, **85–86**, **195**, **243–244**, **272**, **289**, **447**
pagination ..... **64**, **195**
paragraph ..... **64**, 102, **110–112**
parnumber ..... **110–112**
part ..... **64**, 111, **110–112**
partentry ..... **64**, **79**
partentrypagenumber ..... **65**, **79**
partnumber ..... **65**, 111
placeanddate ..... **195**, **225**
PPdata ..... 217, **218**
PPlogo ..... 217, **218**
priority ..... 217, **217**
prioritykey ..... 217, **217**
publishers ..... **65**, **70**, **309–310**

### R

refname ..... **195**, **224**
refvalue ..... **195**, **224**

### S

section ..... **65**, 102, 111, **110–112**
sectionentry ..... **65**, **79**
sectionentrydots ..... **65**, **78**, 79
sectionentrypagenumber ..... **65**, **79**
sectioning ..... **65**
specialmail ..... **195**, 217, **216–217**
subject ..... **65**, **70**, **228–229**, **309**
subparagraph ..... **65**, 102, 111, **110–112**
subsection ..... **65**, 102, 111, **110–112**
subsubsection ..... **66**, 102, 111, **110–112**
subtitle ..... **66**, **70**, **309**

### T

title ..... **66**, **70**, **228**, **309**
titlehead ..... **66**, **70**, **309**
toaddress ..... **195**, 215, **215**, 217
toname ..... **195**, 215, **215**, 217


## Dateien, Klassen und Pakete


fancyhdr (Paket) ..... **267**

### A

adrconv (Paket) ..... 265, 266
amsmath (Paket) ..... 254
article (Klasse) ..... 54

### B

babel (Paket) ... 69, 179, 223, 308, 343, 366, 376, 394, 468, 519
babelbib (Paket) ..... 156
biblatex (Paket) ..... 156, 512
blindtext (Paket) ..... 289
book (Klasse) ..... 54

### C

capt-of (Paket) ..... 409
caption (Paket) ..... 406, 409
color (Paket) ..... 305, 471
contract (Paket) ..... **320**, 468

### D

DIN.lco ..... **259**
DIN5008A.lco ..... **260**
DIN5008B.lco ..... **260**
DINmtext.lco ..... **260**

### E

eso-pic (Paket) ..... 431, 516
etoolbox (Paket) ..... 348

### F

fancyhdr (Paket) ..... 86, 243
float (Paket) ..... 75, 138, 145, 368, 373
floatbytocbasic (Paket) ..... 138
floatrow (Paket) ..... 368, 373
fontawesome (Paket) ..... 208
fontenc (Paket) ..... 37

footmisc (Paket) ..... 94, 247, 314
french (Paket) ..... 519

G

geometry (Paket) ..... 26, 52, 477
german (Paket) ..... 519
graphics (Paket) ..... 212, 218
graphicx (Paket) ..... 93, 218, 516

H

hyperref (Paket) ..... 109, 321, 406

I

index (Paket) ..... 159
isodate (Paket) ..... 223

K

KakuLL.lco ..... **260**
keyval (Paket) ..... 323, 325
KOMAold.lco ..... **260**

L

lco ..... **515–519**
letter (Klasse) ..... 54
lipsum (Paket) ..... 276
listings (Paket) ..... 385, 387
longtable (Paket) ..... 138, 148, **150**

M

marginnote (Paket) ..... 461
marvosym (Paket) ..... 208
microtype (Paket) ..... 57, 165, 303, 376
mparhack (Paket) ..... 461
multicol (Paket) ..... 377, 511

N

nameref (Paket) ..... 109
natbib (Paket) ..... 156, 157
NF.lco ..... **260**
ngerman (Paket) ..... 519
NipponEH.lco ..... **260**
NipponEL.lco ..... **261**
NipponLH.lco ..... **261**
NipponLL.lco ..... **261**
NipponRL.lco ..... **261**

P

polyglossia (Paket) ..... 105, 119, 343

R

ragged2e (Paket) ..... 98, 126, 149, 150, 250, 316
report (Klasse) ..... 54

S

scraddr (Paket) ..... **296–299**
scrartcl (Klasse) . 78, 85, 121, **54–161**, **481–512**
scrbase (Paket) ..... 32, 56, 163, 268, 300, 302, **323–353**, 419, 463
scrbook (Klasse) . 78, 84, 121, **54–161**, **481–512**
scrextend (Paket) . 247, 250, **300–319**, **481–512**
scrhack (Paket) ..... 54, 368
scrjura (Paket) ..... **320**
scrkbase (Paket) ..... 484
scrlayer (Paket) . 39, 88, 267, 284, **418–450**, 457, 461, 465, 470
scrlayer-notecolumn (Paket) ..... **461–475**
scrlayer-scrpage (Paket) ..... 45, 46, 84, 86–88, 120, 241, 243, 244, **267–295**, 444, **451–460**, 461, 465, 468, 470, 514
scrletter (Paket) ..... **162–266**, 300, **513–522**
scrlettr (Klasse) ..... 256
scrlfile (Paket) ..... 300, **354–364**
scrlfile-hook (Paket) ..... **355**
scrlfile-hook-3.34 (Paket) ..... **355**
scrlfile-patcholdkernel (Paket) ..... **355**
scrlogo (Paket) ..... **321**
scrlttr2 (Klasse) ..... **162–266**, 283, **513–522**
scrpage2 (Paket) ..... 243, 267
scrreprt (Klasse) . 78, 84, 121, **54–161**, **481–512**
scrwfile (Paket) ..... **365**, 373, 376, 395
setspace (Paket) ..... 28, 39, 52, 440
SN.lco ..... **261**
SNleft.lco ..... **261**
splitidx (Paket) ..... 159
supertabular (Paket) ..... 138

T

tabularx (Paket) ..... 181
titleref (Paket) ..... 109
tocbasic (Paket) .... 75, 255, **366–417**, 482, 493
typearea (Paket) .. **26–53**, 58, 84, 165, 166, 172, 269, 304, 340, 444, 468, **477–480**, 515
typearea.cfg ..... **480**

### U

UScommercial9.lco ..... **261**
UScommercial9DW.lco ..... **262**

### X

xcolor (Paket) ..... 98, 250, 305, 453, 468, 471
xpatch (Paket) ..... 348

### Z

zref (Paket) ..... 109
zref-titleref (Paket) ..... 109


## Klassen- und Paketoptionen


### A

abstract=*Ein-Aus-Wert* ..... **74**
addrfield ..... 219, 220
addrfield=backgroundimage ..... 219, 220
addrfield=image ..... 219
addrfield=PP ..... 219, 220
addrfield=*Modus* ..... **215–218**
adrFreeIVempty ..... **299**
adrFreeIVshow ..... **299**
adrFreeIVstop ..... **299**
adrFreeIVwarn ..... **299**
appendixprefix=*Ein-Aus-Wert* ..... **101–102**
areasetadvanced ..... 479
areasetadvanced=*Ein-Aus-Wert* ..... **478**
autoclearnotecolumns=*Ein-Aus-Wert* .... **475**
autoenlargeheadfoot=*Ein-Aus-Wert* ..... **269**
automark ..... 243, **286–287**, **444–445**
autooneside=*Ein-Aus-Wert* ..... **286–287**, **444–445**
autoremoveinterfaces ..... 450

### B

backaddress=*Wert* ..... **215–218**
BCOR ..... 47
BCOR=current ..... 40
BCOR=*Korrektur* ..... **33–34**
bibliography=leveldown ..... 157, 158
bibliography=nottotoc ..... 157, 158
bibliography=numbered ..... 157, 158
bibliography=oldstyle ..... 156, 158
bibliography=openstyle ..... 156, 158, 511
bibliography=standardlevel ..... 158
bibliography=totoc ..... 157, 158
bibliography=*Einstellung* ..... **156–157**

### C

captionabove=false ..... **407**
captionabove=true ..... **407**
captionabove=*Ein-Aus-Wert* ..... **406–407**
captionbelow=false ..... **407**
captionbelow=true ..... **407**
captionbelow=*Ein-Aus-Wert* ..... **406–407**
captionposition ..... **406–407**
captionposition=bottom ..... 140, **406–407**
captionposition=top ..... 139, **406–407**
captionposition=bottom ..... 138
captionposition=top ..... 138
captions ..... 142, 143
captions=bottombeside ..... 139, **143**
captions=centeredbeside ..... 139, **143**
captions=figureheading ..... 138, 139
captions=figuresignature ..... 138, 139
captions=heading ..... 138, 139
captions=innerbeside ..... 139, **142**
captions=leftbeside ..... 140, **142**
captions=nooneline ..... 138, 140
captions=oneline ..... 140
captions=outerbeside ..... 140, **142**
captions=rightbeside ..... 140, **142**
captions=signature ..... 138, 140
captions=tableheading ..... 138, 140
captions=tablesignature ..... 138, 140
captions=topbeside ..... 141, **143**
captions=*Einstellung* ..... **137–141**
chapteratlists ..... **106**
chapteratlists=entry ..... 106
chapteratlists=*Wert* ..... **106**
chapterentrydots=*Ein-Aus-Wert* ..... **78**, 79
chapterprefix ..... 503
chapterprefix=*Ein-Aus-Wert* ..... **101–102**
cleardoublepage ..... **91**, **245**, **312**
cleardoublepage=current ..... **91**, **245**, **312**
cleardoublepage=*Seitenstil* ... **91**, **245**, **312**

clines ..... 293, **295**

D

deactivatepagestylelayers=*Ein-Aus-Wert* ..... **440–441**
DIN ..... **259**
DIN5008A ..... **260**
DIN5008B ..... **260**
DINmtext ..... **260**
disablepatch=all ..... 411, 412
disablepatch=caption ..... 412
disablepatch=chapter ..... 412
disablepatch=listoffigures ..... 412
disablepatch=listoftables ..... 412
disablepatch=*Patchliste* ..... **411–417**
disablepatch=starttoc ..... 412
DIV ..... **36–40**, **47**
DIV=areaset ..... 38, **47**
DIV=calc ..... **36**, 38
DIV=classic ..... **36**, 38
DIV=current ..... 38, **37–40**
DIV=default ..... 38
DIV=last ..... 38, **37–40**
DIV=*Faktor* ..... **34–40**
draft=*Ein-Aus-Wert* ... **57–58**, **165**, **287**, **303**, **433**

E

enablepatch=all ..... 411, 412
enablepatch=caption ..... 412
enablepatch=chapter ..... 411, 412
enablepatch=listoffigures ..... 412
enablepatch=listoftables ..... 412
enablepatch=*Patchliste* ..... **411–417**
enablepatch=starttoc ..... 412
enablepatchcaption ..... 407, 411
enlargefirstpage=*Ein-Aus-Wert* ..... **235**
extendedfeature=*Möglichkeit* ..... **303**

F

figurecaptionabove=default ..... **407**
figurecaptionabove=*Ein-Aus-Wert* ..... **407**
figurecaptionbelow=default ..... **407**
figurecaptionbelow=*Ein-Aus-Wert* ..... **407**
figurecaptionposition=bottom ..... 139, **407**
figurecaptionposition=default ..... **407**
figurecaptionposition=top ..... 139, **407**
figurecaptionposition=bottom ..... 138
figurecaptionposition=top ..... 138
firstfoot=false ..... 237
firstfoot=*Ein-Aus-Wert* ..... **235**
firsthead=*Ein-Aus-Wert* ..... **202**
fleqn ..... **136**
foldmarks=*Einstellung* ... **198–200**, 200, 201
fontsize=*Größe* ..... **59**, **189–190**, **304**
footbotline ..... 457
footbotline=*Dicke*:*Länge* ..... **293**
footheight ..... 47
footheight=*Höhe* ..... **45–46**
footinclude=*Ein-Aus-Wert* ..... **42–43**
footlines ..... 47
footlines=*Zeilenanzahl* ..... **45–46**
footnotes=multiple ..... 94
footnotes=nomultiple ..... 94
footnotes=*Einstellung* ..... **94**, **247**, **314**
footsepline ..... 243, 457
footsepline=*Dicke*:*Länge* ..... **293**
footsepline=*Ein-Aus-Wert* ..... **84**, **240–241**
footwidth=*Breite*:*Offset*:*Offset* . **292–293**
forceoverwrite ..... 450
fromalign=*Methode* ..... **202**
fromemail=*Ein-Aus-Wert* ..... **208–211**
fromfax=*Ein-Aus-Wert* ..... **208–211**
fromlogo=*Ein-Aus-Wert* ..... **211–213**
frommobilephone=*Ein-Aus-Wert* ..... **208–211**
fromphone=*Ein-Aus-Wert* ..... **208–211**
fromrule ..... 207
fromrule=*Position* ..... **203–206**
fromurl=*Ein-Aus-Wert* ..... **208–211**

H

headheight ..... 47
headheight=*Höhe* ..... **44–45**
headinclude ..... 84
headinclude=*Ein-Aus-Wert* ..... **42–43**
headings ..... 110, 500
headings=big ..... 102, 103
headings=normal ..... 102, 103
headings=onelineappendix ..... 103
headings=onelinechapter ..... 103
headings=openany ..... 103
headings=openleft ..... 104

headings=openright ..... 104
headings=optiontoheadandtoc .. 103, 104, **107**
headings=optiontohead ..... 103, 104, **107**
headings=optiontotoc ..... 103, **107**
headings=small ..... 102, 104
headings=twolineappendix ..... 104
headings=twolinechapter ..... 105
headings=*Einstellung* ..... **102–105**
headlines ..... 47
headlines=*Zeilenanzahl* ..... **44–45**
headsepline ..... 243, 457
headsepline=*Dicke*:*Länge* ..... **293**
headsepline=*Ein-Aus-Wert* ..... **84**, **240–241**
headtopline ..... 457
headtopline=*Dicke*:*Länge* ..... **293**
headwidth=*Breite*:*Offset*:*Offset* . **292–293**
hmode=*Ein-Aus-Wert* ..... **460**

I

ilines ..... 293, **295**
index=leveldown ..... 160
index=nottotoc ..... 159, 160
index=numbered ..... 160
index=standardlevel ..... 160
index=totoc ..... 159, 160
index=*Einstellung* ..... **159–160**

K

KakuLL ..... **260**
KOMAold ..... **260**

L

leqno ..... **136**
listof=chapterentry ..... 152
listof=chaptergapline ..... 152
listof=chaptergapsmall ..... 106, 152
listof=entryprefix ..... 153
listof=flat ..... 151, 153
listof=graduated ..... 151, 153
listof=leveldown ..... 153, 375
listof=nochaptergap ..... 152, 154
listof=nonumberline ..... 377
listof=nottotoc ..... 154
listof=numbered ..... 154, 376
listof=numberline ..... 377
listof=standardlevel ..... 154
listof=totoc ..... 154, 377
listof=*Einstellung* ..... **151–154**, 368, 377
locfield=*Einstellung* ..... **221–222**

M

manualmark ..... **286–287**, **444–445**
markcase=lower ..... **288**
markcase=noupper ..... **287**, **288**, **446**
markcase=title ..... **288**
markcase=upper ..... 283, **288**
markcase=used ..... 283, **287**, **288**, **445**
markcase=*Wert* ..... **287**, **445–446**
mpinclude=*Ein-Aus-Wert* ..... **43–44**

N

NF ..... **260**
NipponEH ..... **260**
NipponEL ..... **261**
NipponLH ..... **261**
NipponLL ..... **261**
NipponRL ..... **261**
numbers=autoendperiod ..... 105, 106
numbers=endperiod ..... 105, 106
numbers=noendperiod ..... 106
numbers=*Einstellung* ..... **105**
numericaldate=*Ein-Aus-Wert* ..... **223–224**

O

olines ..... 293, **295**
onpsbackground=*Code* ..... **439–440**
onpsevenpage=*Code* ..... **439–440**
onpsfloatpage=*Code* ..... **439–440**
onpsforeground=*Code* ..... **439–440**
onpsinit=*Code* ..... **439–440**
onpsnonfloatpage=*Code* ..... **439–440**
onpsoddpage=*Code* ..... **439–440**
onpsoneside=*Code* ..... **439–440**
onpsselect=*Code* ..... **439–440**
onpstwoside=*Code* ..... **439–440**
open ..... 93
open=left ..... 101
open=right ..... 101
open=*Methode* ..... **100**
origlongtable ..... **150**
overfullrule=*Ein-Aus-Wert* **57–58**, **165**, **303**

P

pagenumber ..... 244
pagenumber=*Position* ..... **241**
pagesize ..... 48
pagesize=automedia ..... 51
pagesize=auto ..... 51
pagesize=dvipdfmx ..... 51
pagesize=dvips ..... 51
pagesize=false ..... 51
pagesize=luatex ..... 51
pagesize=pdftex ..... 51
pagesize=*Ausgabetreiber* ..... **49–50**
pagestyleset=*Einstellung* ..... **283–284**
pagestyleset=KOMA-Script ..... **283–284**
pagestyleset=standard ..... **283–284**
paper=*Ausrichtung* ..... **48–49**
paper=*Format* ..... **48–49**
parskip=false ..... 81
parskip=full* ..... 82
parskip=full+ ..... 82
parskip=full- ..... 81
parskip=full ..... 81
parskip=half* ..... 82
parskip=half+ ..... 82
parskip=half ..... 82
parskip=never ..... 82
parskip=half- ..... 82
parskip=*Methode* ..... **81–82**, **238–239**
plainfootbotline=*Ein-Aus-Wert* ..... **295**
plainfootsepline=*Ein-Aus-Wert* ..... **295**
plainheadsepline=*Ein-Aus-Wert* ..... **295**
plainheadtopline=*Ein-Aus-Wert* ..... **295**
priority=*Priorität* ..... **215–218**

R

refline ..... 225
refline=dateleft ..... 514
refline=dateright ..... 514
refline=narrow ..... 227
refline=nodate ..... 514
refline=wide ..... 227, 258
refline=*Einstellung* ..... **224–226**, 226

S

sectionentrydots=*Ein-Aus-Wert* ..... **78**, 79
sfdefaults=*Ein-Aus-Wert* . **60**, **191**, **304–305**
shrinkfirstpage=*Ein-Aus-Wert* ..... **235**
singlespacing ..... 39
singlespacing=*Ein-Aus-Wert* ..... **440**
SN ..... 258, **261**
SNleft ..... **261**
subject=*Einstellung* ..... **228–230**, 230
symbolicnames=*Wert* ..... **208–211**

T

tablecaptionabove=default ..... **407**
tablecaptionabove=*Ein-Aus-Wert* ..... **407**
tablecaptionbelow=default ..... **407**
tablecaptionbelow=*Ein-Aus-Wert* ..... **407**
tablecaptionposition=bottom ..... 140, **407**
tablecaptionposition=default ..... **407**
tablecaptionposition=top ..... 140, **407**
tablecaptionposition=bottom ..... 138
tablecaptionposition=top ..... 138
titlepage ..... **67**, **307**
titlepage=false ..... 311
titlepage=firstiscover ..... **67**, 72, **307**, 310
titlepage=*Ein-Aus-Wert* ..... **67**, **307**
toc=bibliographynumbered ..... 75, 76
toc=bibliography ..... 75, 76
toc=chapterentrywithdots ..... 76, 79
toc=chapterentrywithoutdots ..... 76
toc=flat ..... 76, 151
toc=graduated ..... 75, 76
toc=indexnumbered ..... 75
toc=index ..... 75, 77
toc=listofnumbered ..... 75, 77, 151
toc=listof ..... 75, 77, 151
toc=nobibliography ..... 77
toc=noindex ..... 77
toc=nolistof ..... 77, 151
toc=nonumberline ..... 377
toc=numberline ..... 77, 153, 377, 383, 384
toc=sectionentrywithdots ..... 78, 79
toc=sectionentrywithoutdots ..... 78
toc=*Einstellung* ..... **75–78**
twocolumn ..... 58, 122
twocolumn=*Ein-Aus-Wert* ..... **41–42**
twoside ..... **40–41**, 58, 279
twoside=semi ..... **40–41**
twoside=*Ein-Aus-Wert* ..... **40–41**

### U

UScommercial9 ..... **261**
UScommercial9DW ..... **262**
usegeometry=*Ein-Aus-Wert* ..... **477–478**

### V

version .. **32–33**, 48, **56–57**, 58, **164**, **302**, 487
version=first ..... **32–33**, **56–57**, **164**, **302**
version=last ..... **32–33**, **56–57**, **164**, **302**
version=*Wert* ..... **32–33**, **56–57**, **164**, **302**
visualize ..... **516–519**

### W

withdeprecated ..... **355**


## Haken (*do-hooks*)


### F

footnote/text/begin ..... **99**, **250**, **316**, **481**
footnote/text/end ..... **99**, **250**, **316**, **481**

### H

heading/begingroup/*Name* ..... **507**
heading/branch/nostar/*Name* ..... **507**
heading/branch/star/*Name* ..... **507**
heading/endgroup/*Name* ..... **507**
heading/postinit/*Name* ..... **507**
heading/preinit/*Name* ..... **507**