Entwicklung einer webbasierter Applikation zur Bearbeitung von PDF Dateien

Bachelorarbeit zur Erlangung des akademischen Grades
Bachelor of Science
im Studiengang Technische Informatik
an der Fakultät für Informations-, Medien- und Elektrotechnik
der Technischen Hochschule Köln

vorgelegt von:	Janina Schroeder
Matrikel-Nr.:	11132206
Adresse:	Laurentiusweg 10 50321 Brühl janina_jessika_jelena.schroeder@smail.th-koeln.de
eingereicht bei:	Prof. Dr. Chunrong Yuan
Zweitgutachter*in:	Prof. Dr. René Wörzberger

Köln, 04.03.2024

Bachelorarbeit

Titel: Entwicklung einer webbasierter Applikation zur Bearbeitung von PDF Dateien

Gutachter:

- Prof. Dr. Chunrong Yuan

- Prof. Dr. René Wörzberger

Zusammenfassung: Für die Bachelorarbeit habe ich eine Open Source offline Webseite zur Bearbeitung von PDF Dateien im Firefox Browser programmiert. Seit Adobe den PDF Standard entwickelt hat, tauchten zahlreiche meist kostenpflichtige PDF Anwendungen, um PDF Dateien zu bearbeiten auf dem Markt auf. Ich habe den Markt an PDF Programmen analysiert und diese mit meiner Webapplikation verglichen. Daraufhin beleuchte ich den aktuellen Stand der Technik des PDF Standards. Im späteren Verlauf erkläre ich die Implementierung meiner Webapp und meine Erfahrungen mit anderen Browsern, sowie auf MacOS, Linux, Android und iOS. Die Javascript Libraries PDF.js und PDF-LIB sind das tragende Fundament meiner PDF Webapp. Die PDF Webapp vereint alle Funktionalitäten, die man für gängige PDF Bearbeitung benötigt. Man kann PDFs lesen, splitten, mergen, erstellen, sowie mit Texten, Bildern, Geometrie und Zeichnungen versehen. Am Ende diskutiere ich, was man hätte besser machen können, welche Funktionalitäten fehlen und welche Features in Zukunft noch geplant sind.

Stichtwörter: PDF Bearbeitung, Adobe, Javascript, Vue JS 3, auf PDF zeichnen, Splitten, Mergen, PDF.js, PDF-LIB

Datum: 04. März 2024

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Glossar

Hund Behaartes, vierbeiniges Säugetier. Bester Freund des Menschen. 19

Abkürzungsverzeichnis

GDI Graphics Device Interface. 3

ISO International Organization for Standardization. 2, 3

PDF Portable Document Format. 2–6, II

PDL Page Description Language. 3–5

RIP Raster Image Processor. 3

SVM support vector machine. 19

Symbolverzeichnis

$\vec{F}$ Kraft, vektorielle Größe. 20

1 Einleitung

Das vorliegende Dokument kann als Muster und Anleitung für wissenschaftliche Abschlussarbeiten verwendet werden. Es beruht ursprünglich auf einem Leitfaden, den Prof. Dr. Stephan Freichel als Prüfungsausschussvorsitzender für die Studiengänge B. Sc. Logistik und M. Sc. *Supply Chain and Operations Management* an der Fakultät für Fahrzeugsysteme und Produktion erstellt hat.

Der Text in dieser Vorlage beschreibt allgemeine formale Anforderungen, insbesondere zum Inhaltsverzeichnis, zum Einfügen von Quellenverweisen und zum Erstellen eines Literaturverzeichnisses.

Die Vorlage kann fachübergreifend als Musterdatei für Abschlussarbeiten an der TH Köln verwendet werden. Allerdings müssen Sie dann unbedingt klären, ob sie den Konventionen in ihrem Studienfach entspricht.

Für die Möglichkeit eines fachunabhängigen Gebrauchs wurde das Dokument von Maria-Anna Worth (i. R.) und Susanne Neuzerling (Hochschulreferat Planung und Controlling) inhaltlich überarbeitet und modifiziert. Eine erneute Überarbeitung und Aktualisierung erfolgte durch Andreas Bissels (Schreibzentrum). Frau Katharina Bata hat die Dokumentvorlage in LaTeX erstellt. Zuletzt wurde diese Vorlage 2023 von André Ulrich und Jan Salmen überarbeitet und um diverse Hinweise speziell zur Gestaltung mit LaTeX ergänzt (siehe Kapitel 6).

Zur Vorbereitung auf Ihre Abschlussarbeit empfehlen wir Ihnen die Angebote des Schreibzentrums der Kompetenzwerkstatt[1]; hierzu gehören sowohl eine Schreibberatung als auch Schreibkurse. Das Schreibzentrum ist Ihre Anlaufstelle an der TH Köln in Fragen rund um das wissenschaftliche Schreiben.

Sichern Sie diese Dokumentvorlage bitte zweifach auf Ihrem Rechner: Einmal, um weiterhin auf den hier dargestellten Inhalt zugreifen zu können, und ein zweites Mal, um sie mit Ihrer eigenen Abschlussarbeit zu überschreiben.

Bitte beachten Sie: Die Vorlage ersetzt nicht die spezifischen Vorgaben der jeweiligen Prüfungsausschüsse. Sollte es in Ihrem Fach besondere formale Vorgaben geben, so gelten diese.

Köln, August 2023

[1] https://www.th-koeln.de/schreibzentrum

2 Grundlagen

2.1 PDF Einführung

Die Popularität von PDF (Portable Document Format) Dateien ist seit 2008 rasant angestiegen in der globalen Informationsübertragung. Täglich werden weltweit 2,5 Milliarden PDF-Dokumente erzeugt. Seine Beliebtheit verdankt PDF vor allem an der plattformübergreifenden Kompatibilität (Desktop-Computer, Tablets und Smartphones), denn PDF Dokumente ist auf mehr als 1,5 Milliarden Geräten ohne zusätzliche Software lesbar. Über 80% der geschäftlichen Dokumente werden als PDF Datei weitergegeben. [1] 90 % der Büroangestellten wollen auf das PDF Dateiformat nicht mehr verzichten. Drei Viertel aller archivierten Dokumente sind PDF Dokumente. [3] Das PDF Dateiformat steht für Plattformunabhängigkeit, Konsistenz in Formatierung und Layout und soll ein möglichst originalgetreues Druckergebnis liefern.

PDF wurde 1993 von Adobe veröffentlicht und ging aus dem 1991 von Adobe-Mitbegründer John Warnock gestarteten „Project Camelot"hervor. Ziel dieses Projektes war, ein Dateiformat für elektronische Dokumente zu kreieren, sodass diese Anwendungsprogramm, Betriebssystem und Hardware unabhängig originalgetreu wiedergegeben werden können. Anfangs war der Adobe Reader kostenpflichtig und PDF war für einen langen Zeitraum ein proprietäres Dateiformat, welches offengelegt im PDF Reference Manual von Adobe dokumentiert ist. Die International Organization for Standardization (ISO) übernahm PDF 2007 in den Standardisierungsprozess und seit der Veröffentlichung von PDF Version 1.7 am 1. Juli 2008 gilt PDF als Offener Standard. [2] Der Begriff Offener Standard bezeichnet einen Standard, der für alle Teilhaber am Markt besonders leicht zugänglich, weiterentwickelbar und einsetzbar ist. Das bedeutet, dass der Standard von einer gemeinnützigen Organisation eingeführt, veröffentlicht, weiterentwickelt und gleichmäßige Einflussnahme aller interessierten Parteien ermöglicht. [4]

2.2 PDF Funktionsumfang

Um die Navigation innerhalb eines PDF Dokuments zu erleichtern können PDFs anklickbare Inhaltsverzeichnisse und miniaturisierte Seitenvorschauen enthalten.

2.3 PDF Standards

2.3.1 PDF/X

Erleichterung des Datenaustausches in der Druckvorstufe.

2.3.2 PDF/A

Langzeitarchivierung von PDF-Dateien (als PDF/A-1 in ISO 19005-1:2005)

2.4 PDF Dateiversionen

2.4.1 PDF 1.7

Veröffentlichung am 1. Juli 2008 ist PDF in Version 1.7 als ISO 32000-1:2008 ein Offener Standard

2.5 PDF Implementierung

PDF ist eine vektorbasierte Page Description Language (PDL) (Seitenbeschreibungssprache) und basiert auf dem PostScript-Format. Eine PDL beschreibt den Seitenaufbau, wie die Seite in einem Ausgabeprogramm bzw. Ausgabegerät, z.B. einem Drucker, aussehen soll. PDLs können Seiten mit Vektoren beschreiben. Das Ausgabeformat ist normalerweise nicht zur weiteren Bearbeitung vorgesehen. An den Drucker wird durch die PDL ein Datenstrom der zu druckenden Aufgabe erzeugt und an den Drucker gesendet. Der Raster Image Processor (RIP) eines Druckers wandelt die Bildschirmausgabe in die gerasterte Druckerausgabe um. Viele APIs der Hardwareabstraktionsschicht im Computer wie Graphics Device Interface (GDI) oder OpenGL können in PDL ausgeben. Speichert ein Satzprogramm den Seitenbeschreibungscode eines Dokuments in einer Datei, müssen Drucker die PDL nicht selbst verarbeiten. Im Common Unix Printing System, der Standard-Druckersteuerung von Linux hat der PostScript und der PDF-Interpreter ghostscript die Aufgabe eines RIP, d.h. er ist für die Umwandlung in die gerasterte Druckausgabe auf dem Drucker zuständig. Zudem stellen PDLs eine Schnittstelle zum Quellcode eines Dokuments bzw. zu Programmen, die Quellcode verwalten oder das Dokument formatieren können, dar. Die PDL PDF erweitert die Funktionalität der Vorschau am Bildschirm um anklickbare

Links (Hypertextfunktionalität), die die Navigation im Dokument erleichtern oder um URLs, die sich automatisch im Browser öffnen. [5]

2.5.1 PostScript

Die PostScript PDL wurde in den 1980er Jahren von Adobe erfunden. [6] Hinzu wurden weiter PostScript Technologien entwickelt, die aus der stackorientierten, interpretierten Programmiersprache PostScript [7], Grafik-, Textformatierungsanwendungen, Treibern und Abbildungssystemen bestehen. PostScript hat sich als Industriestandard etabliert. Die letzte Version ist PostScript 3. Seine primäre Anwendung gemäß des Adobe imaging models findet sich in der Beschreibung von Textdarstellung, graphische Formen und Bildern auf gedruckten oder auf dem Bildschirm angezeigten Seiten. Dabei ist die Beschreibung des Dokuments geräteunabhängig. PostScript unterstützt unter anderem beliebige geometrische Formen, Zeichenoperationen in Graustufen, RGB, CMYK und CIE (Yxy-Farbraum) und vorinstallierte oder benutzerdefinierte Fonts und Digitalbilder jeglicher Auflösung je nach Farbmodell und ein allgemeines Koordinatensystem. Dabei werden die Textzeichen eines Fonts, gemäß des Adobe imaging models, als graphische Formen betrachtet auf denen Grafikoperationen möglich sind. Das Koordinatensystem unterstützt alle linearen Transformationen, die auf alle Seitenelemente angewandt werden können. Die Seitenbeschreibung in PostScript kann auf jedem Gerät, was einen PostScript Interpreter implementiert, gerendert werden. In diesem Prozess wird die high-level PostScript-Beschreibung in low-level Rasterdatenformate für das jeweilige Gerät übersetzt. PostScript Programme können erstellt, übertragen und als ASCII Quellcode interpretiert werden. [6]

2.5.2 Adobe imaging model

PDF und die PostScript Programmiersprache haben das Adobe imaging model als Gemeinsamkeit. Es kann nahtlos zwischen PDF und PostScript konvertiert werden und beide erzielen das gleiche Ausgabeergebnis beim Druck. Dennoch fehlt PDF das general-purpose Framework der PostScript Programmiersprache. Stattdessen stellt ein PDF Dokument eine statische Datenstruktur optimiert für den random access dar und enthält zusätzlich Seitennavigationsinformationen für interaktives Lesen. Das high-level imaging model beschreibt die Elemente, die auf der Seite dargestellt werden, also Text, Geometrie oder Bilder, als abstrakte graphische Elemente, anstatt als Pixeldefinitionen. Dadurch wird das imaging model zu einem geräteunabhängigem Modell und kann hochwertige Ausgaben auf vielen verschiedenen Druckern und Bildschirmen liefern. Die PDL beschreibt dieses imaging model. Eine Anwendung generiert zuerst die geräteunabhängige Beschreibung des gewünschten Ausgabegeräts

in der PDL. Daraufhin interpretiert eine Firmware oder Software eines spezifischen Ausgabegeräts für Rasterausgaben die Beschreibung und rendert sie im Ausgabegerät. Hierbei hat die PDL die Rolle eines Austauschstandards für die Übertragung und Speicherung von druckbarem oder auf Displays darstellbaren Dokumenten. [6]

2.6 PDF Sicherheitsaspekte

Etwa 40 % der Unternehmen setzen PDFs für geschützte Inhalte ein. In den letzten 2 Jahren ist die Nutzung der elektronischen Signaturfunktion in PDFs um mehr als 150 % gestiegen. [1]

PDF unterstützt Verschlüsselung und die Vergabe von 2 Passworttypen. Eventuell kann beim Öffnen einer Datei ein Passwort gefordert werden oder das Kopieren von Teilinhalten, jeglichem Inhalt oder das Ausdrucken ist gesperrt.

2.7 Aktueller Stand von Forschung und Technik von PDF

3 PDF Programme auf dem Markt

Bis 2025 werden über 3 Millarden Dollar jährlich für PDF Editoren ausgegeben werden. [3]

3.1 Freie PDF Programme und Onlinedienste

PDF Dateien lassen sich in vielen Programmen einfach über den Druckdialog erstellen. Apple hat das Lesen von PDF Dokumenten in seiner Apples Vorschau integriert. Viele Webbrowser stellen PDF Viewer bereit, so Google Chrome seit 2010. [2]

3.1.1 PDFCreator

PDF Dokumente und Dateien erzeugen

3.1.2 LibreOffice

PDF Dokumente und Dateien erzeugen

3.1.3 OpenOffice

PDF Dokumente und Dateien erzeugen

3.1.4 ghostscript

3.2 Kostenpflichtige PDF Programme und Onlinedienste

3.2.1 Adobe Acrobat

4 MultiPDFmin - Open Source PDF Web App

5 Formaler Aufbau

In diesem Kapitel finden Sie grundlegende Hinweise zum formalen Aufbau Ihrer Arbeit.

5.1 Reihenfolge

Eine wissenschaftliche Arbeit besteht in der Regel aus den folgenden Teilen:

1. Deckblatt
2. Kurzfassung/Abstract (optional)
3. Inhaltsverzeichnis
4. Abbildungs- und Tabellenverzeichnis (auch am Ende üblich)
5. Abkürzungsverzeichnis (auch am Ende üblich)
6. Einleitung
7. Hauptteil
8. Zusammenfassung/Fazit
9. Literaturverzeichnis
10. Anhänge (optional)
11. Erklärung

5.2 Deckblatt

Das Deckblatt beinhaltet: Titel der Arbeit, Art der Arbeit, Verfasser*in, Matrikelnummer, Abgabetermin, Betreuer*in sowie Zweitgutachter*in. Das Deckblatt wird bei Arbeiten, die länger sind als 15 Seiten, bei der Seitenanzahl zwar mitgezählt, jedoch nicht nummeriert.

5.3 Inhaltsverzeichnis

Wir empfehlen eine Dezimalgliederung wie in diesem Dokument angelegt. Werden innerhalb eines Kapitels Unterüberschriften verwendet, müssen mindestens zwei vorhanden sein: wo ein 2.1 ist, muss es ein 2.2 geben.

Das Inhaltsverzeichnis enthält immer die Seitenangaben zu den aufgelisteten Gliederungspunkten; es wird dabei aber selbst nicht im Inhaltsverzeichnis aufgelistet. Die Seiten, die das Inhaltsverzeichnis selbst einnimmt, können römisch gezählt werden.

Für eine Abschlussarbeit ist eine Gliederungstiefe von wenigstens drei Ebenen üblich. In der Regel werden nur bis zu vier Ebenen vorne im Inhaltsverzeichnis abgebildet. Hier sollten Sie aber unbedingt die Gepflogenheiten in Ihrem Fach berücksichtigen und ggf. in Erfahrung bringen.

5.4 Abbildungsverzeichnis und Tabellenverzeichnis

Abbildungen und Tabellen werden in entsprechenden Verzeichnissen gelistet. In dieser Vorlage erscheinen sie direkt nach dem Inhaltsverzeichnis. Dann können die entsprechenden Seiten römisch gezählt werden. Die Verzeichnisse können jedoch auch am Ende der Arbeit vor oder hinter dem Literaturverzeichnis stehen. Dann werden sie regulär mit Seitenzahlen versehen.

5.5 Abkürzungsverzeichnis

Die Zahl der Abkürzungen sollte übersichtlich bleiben. Das Abkürzungsverzeichnis enthält lediglich wichtige fachspezifischen Abkürzungen in alphabetischer Reihenfolge, insbesondere Abkürzungen von Organisationen, Verbänden oder Gesetzen. Gängige Abkürzungen wie „u. a.“, „z. B.“, „etc.“ werden nicht aufgenommen.

Zur technischen Umsetzung mit LaTeX vergleiche auch Abschnitt 6.18.

5.6 Literaturverzeichnis

Das Literaturverzeichnis wird alphabetisch nach Autorennamen geordnet. Es enthält alle im Text zitierten Quellen – und nur diese. Mehrere Schriften einer Person werden nach Erscheinungsjahr geordnet. Schriften derselben Person aus einem Erscheinungsjahr müssen Sie selbst unterscheidbar machen. In den Ingenieurwissenschaften wird

zusätzlich häufig ein Nummern- oder Autorenkürzel dem Namen in eckigen Klammern voran-gestellt. Mehr hierzu und weitere wichtige Regeln des Zitierens lernen Sie in den E-Learning-Kursen des Schreibzentrums[1] kennen.

Zur Verwaltung der verwendeten Literatur eigenen sich entsprechende Softwaretools wie Citavi oder Zotero, die mit verschiedenen Textverarbeitungsprogrammen kompatibel sind.

5.7 Rechtschreibung, Grammatik

Achten Sie bei der Abgabe Ihrer Arbeit auf ein einwandfreies Deutsch bzw. Englisch. Wenn Fehler die Lesbarkeit beeinträchtigen, kann sich dies durchaus negativ auf die Note auswirken. Nutzen Sie daher unbedingt die Rechtschreibprüfung Ihres Textverarbeitungsprogramms, auch wenn diese nicht alle Fehler erkennt.

Für alle, die sich bei diesem Thema unsicher fühlen, empfehlen wir die E-Learning-Kurse des Schreibzentrums[2]. Wenden Sie sich ggf. auch an die Beauftragte für Studierende mit Beeinträchtigung[3].

5.8 Umfang der Arbeit

Alle Fächer nennen verbindliche Angaben zu Unter- und Obergrenzen, die in der Regel eingehalten werden müssen. Verzeichnisse und Anhänge werden dabei in aller Regel nicht mitgezählt. In Einzelfällen – insbesondere bei empirischen Arbeiten – können abweichende Vereinbarungen mit der Betreuungsperson getroffen werden.

[1] https://ilu.th-koeln.de/goto.php?target=cat_52109&client_id=thkilu
[2] https://ilu.th-koeln.de/goto.php?target=cat_52109&client_id=thkilu
[3] https://www.th-koeln.de/studium/studieren-mit-beeintraechtigung_169.php

6 Gestaltung: Textsatz mit LaTeX

Mit LaTeX ist es verhältnismäßig einfach, Dokumente zu erstellen, die professionellen Ansprüchen genügen. Ein entscheidender Vorteil ist, dass der Nutzer fast nur den Inhalt beisteuert, während die korrekte äußere Form dann automatisch erzeugt wird. LaTeX basiert auf TeX, das von Donald Knuth entwickelt wurde [**knuth:tex**]. Einige weitere Vorteile gegenüber gängiger Textverarbeitung:

Frei/Plattformunabhägig: Bei LaTeX handelt es sich um freie Software. Es wird kein proprietärer Editor benötigt, um LaTeX-Dokumente zu schreiben. Tatsächlich können die Dokumente auf *jedem* Rechner mit *jedem* beliebigen Editor bearbeitet werden.

Reines Textformat: Der Quelltext – die `tex`-Datei – ist ein reines Textformat. Dadurch eignen sich LaTeX-Dokumente auch hervorragend zur Versionskontrolle mit beispielsweise git. Dies wiederum ermöglicht eine effiziente Zusammenarbeit mehrerer Autor*innen.

Aufteilen großer Dokumente: Der Quelltext großer Dokumente, wie beispielsweise von Projektarbeiten, kann auf mehrere Dateien aufgeteilt werden. So können beispielsweise mehrere Personen an jeweils einem eigenen Kapitel arbeiten. Aufgrund der beiden oberen Punkte wird es auch nicht zu Kompatibilitätsproblemen kommen.

Trennen von Layout/Inhalt: Mit LaTeX kann man explizit das Layout für das gesamte Dokument festlegen – oder die verwendete Dokumentklasse kümmert sich implizit darum. Zeitgemäße Textverarbeitung bietet mit Formatvorlagen zwar entsprechende Funktionalitäten; aber durch den programmatischen Ansatz mit LaTeX kann noch genauer Einfluss auf das Layout genommen werden. Anschließend kann die ganze Konzentration auf das Schreiben gelegt werden.

Professionelles Ergebnis: Ein mit LaTeX erzeugtes Dokument schaut professioneller aus, als ein entsprechendes, mit Textverarbeitung erzeugtes Dokument. Das gilt vor allem für mathematiklastige Dokumente. Aber auch andere Dokumente können von einem einheitlichen Layout, gleichmäßigem Grauwert des Fließtexts, stimmigeren Seitenumbrüchen und hochwertigen Vektorgraphiken profitieren – um nur mal einige Punkte zu nennen.

Vielseitig einsetzbar: Mit LaTeX können nicht nur „einfache“ Dokumente erzeugt werden. Es existieren unzählige Dokumentklassen, die beispielsweise auch das Erstellen von Präsentationen oder Postern ermöglichen.

In den folgenden Abschnitten 6.1 bis 6.18 wird auf diverse Aspekte eingegangen, die Sie beim Erstellen Ihres Dokuments berücksichtigen sollten.

6.1 Unter der Haube

Sie definieren in Ihren `TEX`-Dokumenten, was Ihre Inhalte sind (Text mit Gliederung, Bilder, Tabellen, Literaturverweise, ...) und wie diese jeweils grob aussehen sollen (z. B. Platzierung von Abbildungen mittels *Gleitumgebungen*, vgl. Abschnitt 6.6).

Beim Erstellen des endgültigen Dokuments wendet LaTeX „unter der Haube“ eine ganze Menge Regeln an, die festlegen, wie das alles bestmöglich umgesetzt werden kann. Diese Regeln betreffen z. B. den Anteil von Text und Bildern pro Seite, Abstände innerhalb von Zeilen, aber auch Sonderfälle wie das Vermeiden einzelner Zeilen eines Abschnitts alleine auf einer Seite (sog. „Schusterjungen“ oder „Hurenkinder“).

Im Ergebnis kann es also passieren, dass z. B. Ihre Abbildungen „springen“, während Sie an Ihrem Text arbeiten. Das hat im Zweifel alles seine Richtigkeit und kann im Notfall am Ende noch optimiert werden.

In diesem Zusammenhang ist zu vermeiden, in den Gestaltungsprozess einzugreifen, indem z. B. manuell Zeilenumbrüche („`\newline`“ oder „`\\`“) eingefügt werden oder Abstände. Ausnahmen bitte nur in begründeten Fällen wie in dieser Vorlage bei der Gestaltung des Deckblatts.

Weitere Infos dazu, wie Sie mit dieser Vorlage hier weiterarbeiten können, finden Sie in Abschnitt 6.18.

6.2 Überschriften

Wir nutzen in dieser Vorlage das Kapitel („`\chapter`“) als höchste Gliederungsebene. Danach kommen Abschnitte („`\section`“) und Unterabschnitte („`\subsection`“). Diese drei Ebenen werden nummeriert und erscheinen im Inhaltsverzeichnis. Falls Sie Ihren Text weiter gliedern wollen, gibt es noch den „`\paragraph`“-Befehl.

Bitte beachten Sie, dass im Text nie zwei Überschriften direkt aufeinander folgen sollten. Nach einer Überschrift kommt immer erst etwas Text (siehe z. B. die Kapitelanfänge hier auf Seite 8 und Seite 11). Für weitere Hinweise vgl. Abschnitt 5.3.

6.3 Absätze

Stellen im Text, an denen ein neuer Absatz beginnen soll, können im Quellcode durch „`\par`“ markiert werden. Wie diese Absätze im fertigen Dokument genau aussehen, wird durch den Parameter „`\parskip`“ in der Dokumentenklasse bestimmt – dazu mehr in Abschnitt 6.18. Das ist ein großer Vorteil von LaTeX: Der Stil kann jederzeit für das gesamte Dokument einfach verändert werden.

Hinweis: Sie erhalten das gleiche Verhalten auch, wenn Sie im Quellcode statt des „`\par`“-Befehls eine leere Zeile stehen lassen. Vielleicht gefällt Ihnen das sogar noch besser.

6.4 Silbentrennung

Die automatische Silbentrennung in LaTeX funktioniert grundsätzlich gut. Es kann aber immer mal kleinere Probleme und erwünschtes Verhalten geben. Wenn Sie die Trennung für ein bestimmtes Wort beeinflussen möchten, können Sie mit dem „`\hyphenation`“-Befehl manuell die erlaubten Trennstellen spezifizieren. So kann man insbesondere erreichen, dass bestimmte Wörter nie getrennt werden, was z. B. für Eigennamen unerwünscht sein könnte.

Zum Beispiel werden Wörter, die einen Bindestrich enthalten, *nur* dort getrennt, das kann dazu führen, dass Zeilen nicht richtig dargestellt werden können, was zu einer Warnung führt (siehe Abschnitt 6.17). In solche Fällen müssten Sie im Quellcode manuell zusätzlich Trennstellen angeben.

6.5 Aufzählungen

Nutzen Sie die Umgebungen

- „`\begin{itemize}`“ ...„`\end{itemize}`“
- „`\begin{enumerate}`“ ...„`\end{enumerate}`“
- „`\begin{description}`“ ...„`\end{description}`“
- „`\begin{labeling}`“ ...„`\end{labeling}`“

um schöne Listen zu erstellen. Auch hier gilt, dass das genaue Aussehen im Dokument global eingestellt wird, das können Sie jederzeit verändern, dazu mehr in Abschnitt 6.18.

6.6 Abbildungen

Wenn jemand Ihre fertige Arbeit in die Hände bekommt, kann es gut sein, dass sie/er zunächst grob durchblättert, dabei kaum Text liest, aber die Abbildungen anschaut. Aus dieser Erfahrung entstammt die „Regel“, dass man die wichtigsten Punkte der Arbeit auf diese Weise verstehen können sollte.

Abbildungen stehen nie alleine, sondern werden durch die Unterschrift (*caption*) beschrieben. Dabei sollte alles enthalten sein, was notwendig ist, um die Abbildung zu verstehen. Nur in Ausnahmefällen muss man in der Unterschrift auf den Text verweisen. Umgekehrt muss auf jede Abbildung mindestens ein Mal im Text verwiesen werden, dazu siehe auch Abschnitt 6.12.

In den folgenden beiden Abschnitten wird zwischen Bildern (in Abschnitt 6.6.1) und Vektorgrafiken (in Abschnitt 6.6.2) unterschieden, da es sich um ganz unterschiedliche Techniken handelt, die jeweils passend genutzt werden sollten.

Denken Sie daran, dass nicht alle Menschen alle Farben gleich gut sehen können. Etwa 10 % der Männer in Deutschland sind beispielsweise von einer Rot-Grün-Schwäche betroffen. Vielleicht wird Ihre Arbeit auch auf einem Schwarz-Weiß-Drucker gedruckt. Daher sollten Sie Abbildungen im besten Fall so gestalten, dass sie auch ohne Farben verständlich sind.

6.6.1 Bilder

Bilder können Sie mit „`\includegraphics`“ einbinden. Es reicht (und wird sogar empfohlen!), den Dateinamen ohne Endung und ohne Pfad anzugeben. Beim Kompilieren werden alle Verzeichnisse durchsucht, die im „`\graphicspath`“ angegeben sind. In aller Regel soll ein Bild nicht alleine im Dokument erscheinen, sondern in einer

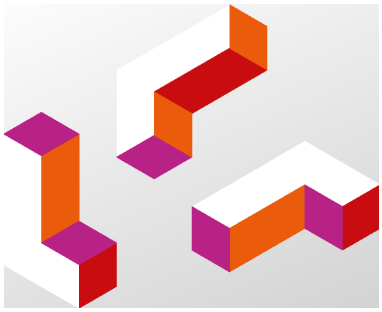

Abbildung 6.1: Vielleicht handelt es sich hierbei um Kunst?

Umgebung, die die automatische Nummerierung sicherstellt, eine Bildunterschrift hinzufügt und schließlich ermöglicht, dass die Abbildung an einer optimalen Stelle platziert wird (daher auch die Bezeichnung „Gleitumgebung“. In diesem Fall ist das die „`\figure`“-Umgebung.

Für die Umgebung stellen wir ein, wo sie auftauchen darf (dazu siehe auch Abschnitt 6.1). Dabei steht `t` für ganz oben auf der Seite, `b` für ganz unten und `h` für „hier“, was also die Positionierung innerhalb des Texts meint. Falls Sie mal Platz sparen müssen, sind `t` und `b` zu bevorzugen.

Achtung: Viele Inhalte wie Formeln, Code, Diagramme, Visualisierung von Daten, usw. sollten *nicht* als Bild eingefügt werden, sondern in einer passenden Form. Dazu siehe den folgenden Abschnitt über Vektorgrafiken.

6.6.2 Vektorgrafiken

Einfache Abbildungen (z. B. Koordinatensysteme, Ablaufdiagramme, usw.) müssen Sie nicht als Bild einfügen. Stattdessen können diese im Quellcode direkt erzeugen können. Dafür bietet sich das mächtige „`tikz`“-Paket an.

Ein Vorteil ist, dass Ihr Dokument so kleiner bleibt. Aber auch, dass die Abbildungen i. d. R. hübscher aussehen. Das gilt insbesondere beim Betrachten am Bildschirm, da sich Vektorgrafiken beliebig skalieren lassen. Das erlaubt es Ihnen sogar, Ihre Daten,

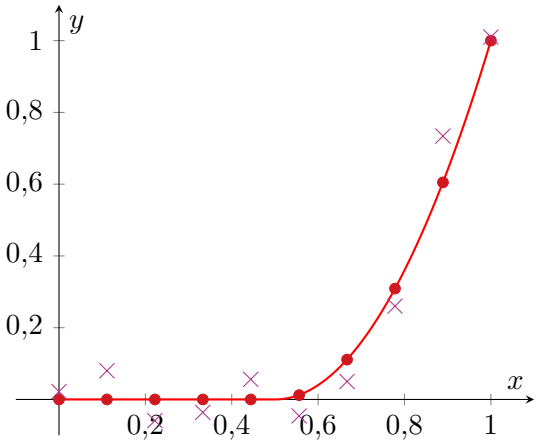

Abbildung 6.2: Eine schöne Grafik, die im Quellcode erzeugt wird!

z. B. aus Experimenten, separat zu halten und entsprechende Abbildungen dynamisch daraus zu generieren. Siehe dazu das Beispiel in Abbildung 6.2.

Sie finden ganz viel Beispiele zu TikZ natürlich im Internet. Außerdem gibt es ein aktuelles Buch [**kottwitz:tikz**].

6.7 Tabellen

Grundsätzlich werden Tabellen in LaTeX mit der „`\tabular`“-Umgebung gebaut. Das ist dann nur die Tabelle selbst, ohne Nummerierung und ohne Bildunterschrift. Das Prinzip ist also das gleiche wie bei Abbildungen (s. o.): Erst die Umgebung (hier „`\table`“), darin die Tabelle selbst. Vielleicht sind die Befehle `rowcolor` oder

Überschrift links	Überschrift rechts
1	2222
10	222
100	22

Tabelle 6.1: Eine einfache Tabelle

`multicolumn` irgendwann für Sie nützlich. Es gibt noch viele weitere Pakete, die helfen, noch hübschere Tabellen zu gestalten, beispielhaft seien hier nur `array`, `booktabs` und `tabularx` genannt.

6.8 Abbildungs- und Tabellenverzeichnis

Mit LaTeX lassen sich Abbildungs- und Tabellenverzeichnis automatisch erstellen. Dabei tauchen alle Einträge entsprechend auf, für die Sie *Gleitumgebungen* korrekt angelegt haben (siehe Abschnitt 6.6.1 und 6.7).

Für diese Verzeichnisse wird standardmäßig der Text aus der `caption` übernommen. Dabei kommt es immer wieder vor, dass diese Beschreibung zu lang ist. Dafür kann mit in der `caption` in eckigen Klammern optional eine kürzere Version angeben. Dazu siehe auch **??**.

6.9 Formeln

Eine der größten Stärken von LaTeX ist, dass man viele Möglichkeiten hat, Formeln einfach und schön aufzuschreiben. Das „`amsmath`“-Paket ist in diesem Zusammenhang

besonders beliebt, weil es ganz viele Möglichkeiten bietet. Hier nur ein kleines Beispiel mit der „`align`“-Umgebung:

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + (n-1) + n \tag{6.1}$$

$$= (1 + n) + (2 + (n-1)) + \ldots \tag{6.2}$$

$$= \frac{1}{2} \cdot (n+1) \tag{6.3}$$

Aber auch einfache Formeln im Text wie $x \in \mathbb{N}$ sind natürlich möglich. Ein häufiger Fehler dabei ist, dass der „Mathe-Modus“ im Text vergessen wird: Wir sprechen über den x-Wert und *nicht* den x-Wert.

Ebenso häufig gibt es den Fehler auch andersrum, also dass Text im Mathe-Modus geschrieben wird.: $a_{falsch} = 42$, aber $a_{richtig} = 42$, vielleicht auch $a_{\text{richtig}} = 42$.

Funktionen wie sin werden automatisch gut dargestellt, wie in

$$\sin\alpha = \left(\frac{a}{c}\right) \tag{6.4}$$

Die Klammern wurden hier nur eingefügt, um den entsprechenden Mechanismus zu demonstrieren: automatisch wachsende Klammern!

Manchmal wollen Sie einen eigenen „Operator“ benutzen, der optisch gleich aussehen soll. Genau dafür ist der „`\DeclareMathOperator`“-Befehl da, damit kann man fehlende Funktionen wie etwa $\operatorname{sgn}(x)$ hinzufügen.

Es empfiehlt sich, allen mathematischen Symbole, die Sie in Ihrer Arbeit benutzen, im Quellcode sprechende Namen zu geben, das geht am einfachsten mit dem „`\newcommand`“-Befehl. Dann können Sie jederzeit anpassen, wie Sie den Gewichtsvektor $\mathbf{w}$ im gesamten Dokument darstellen wollen oder wie die imaginäre Einheit i mit $\mathrm{i}^2 = -1$ aussehen soll. Das setzt natürlich voraus, dass Sie die Bezeichnungen konsequent nutzen. Dadurch wird aber auch Ihr Quellcode besser lesbar!

6.10 Quellcode, Pseudocode

Soll in der Abschlussarbeit ein Ausschnitt vom Quelltext dargestellt werden, so ist die naheliegende Idee, einfach einen Screenshot davon aufzunehmen und via `\includegraphics` als Abbildung einzufügen. Allerdings entpuppt sich diese Idee als schlecht, sobald das fertige Dokument näher herangezoomt wird: Sofort verpixelt der dargestellte Quelltext. LaTeX bietet hierfür jedoch eine elegantere Alternative:

Das Paket `listings` zum Darstellen von Quelltext – direkt im Quelltext des LaTeX-Dokuments oder aber direkt aus einer externen Datei ausgelesen.

Mithilfe der Umgebung `lstlisting` lässt sich der Quelltext direkt im LaTeX-Dokument eingeben. Mit dem Befehl `\lstinputlisting{`⟨*Datei*⟩`}` lässt sich der Quelltext aus einer externen ⟨*Datei*⟩ auslesen und darstellen. Achtung: Der Pfad zu ⟨*Datei*⟩, relativ zum LaTeX-Dokument, muss hierbei angegeben werden.

Außerdem kann das Layout des im fertigen Dokument dargestellten Quelltexts beeinflusst werden. Hierfür existiert ein *key-value-Interface*, über welches mithilfe spezieller *keys* Einfluss auf Dinge wie beispielsweise die zu verwendende Schriftart oder die Hintergrundfarbe genommen wird. Dazu wird der Befehl `\lstdefinestyle{`⟨*Stil*⟩`}` verwendet. Dabei ist ⟨*Stil*⟩ eine Liste mehrerer Paare der Form ⟨*key*⟩=⟨*value*⟩, welche jeweils durch ein Komma voneinander getrennt werden. Ein Beispiel ist in der Präambel dieser Vorlage, in der Datei `definitions.tex` zu finden. Für nähere Informationen sei an dieser Stelle auf die Dokumentation des Paktes verwiesen. Ein Beispiel für solch ein mit der Umgebung `lstlisting` erzeugten Quelltext ist in Abbildung 6.3 gegeben.

```
\lstdefinestyle{myLaTeX}{
  language=TeX,
  basicstyle=\footnotesize\ttfamily,
  frame=single,
  backgroundcolor=\color{gray!10},
}
```

```
\begin{lstlisting}
\lstdefinestyle{myLaTeX}{
  language=TeX,
  basicstyle=\footnotesize\ttfamily,
  frame=single,
  backgroundcolor=\color{gray!10},
}
\end{lstlisting}
```

Abbildung 6.3: Beispiel für ein listing, welches mithilfe der Umgebung `lstlisting` erstellt worden ist. Links ist das fertige Listing zu sehen, rechts ist der entsprechende Quelltext dargestellt, der zu ebenjener Ausgabe führt. Zufälligerweise handelt es sich um einen Ausschnitt desjeniges Stils, der in dieser Vorlage verwendet wird.

6.11 Weitere Verzeichnisse

Mithilfe des Pakets `glossaries` lassen sich weitere Verzeichnisse erzeugen. Ein Glossar sowie ein Abkürzungs- oder Symbolverzeichnis lassen sich direkt erzeugen. Außerdem können auch weitere Verzeichnisse definiert werden. Wer das komplette Potential von

`glossaries` ausschöpfen möchte, benötigt Perl auf dem Rechner sowie das Pearl-Skript `makeglossaries`. Allerdings existiert auch eine „eingedampfte“ Variante mit etwas eingeschränkter Funktionalität, welche komplett ohne Pearl und externes Skript auskommt. Hierzu sei auf die Dokumentation des Pakets verwiesen.

Durch die Option `toc` beim Laden von `glossaries` erscheinen die zusätzlichen Verzeichnisse auch im Inhaltsverzeichnis. Wird weiterhin das Paket `hyperref` verwendet, so sind die im Text ausgegebenen Einträge dieser Verzeichnisse Links, die direkt in das entsprechende Verzeichnis führen. Die Verzeichnisse selbst können dann durch den Befehl `\printglossaries` an der gewünschten Stelle im Dokument ausgegeben werden. Auch hier wird für weiterführende Informationen wieder auf die Dokumentation des Pakets verwiesen.

6.11.1 Glossar erstellen

Ein Glossar kann ohne weitere Vorkehrungen direkt verwendet werden. Ein Eintrag im Glossar kann dann über den Befehl `\newglossaryentry{`⟨*Label*⟩`}{`⟨*Spezifikation*⟩`}` definiert werden. Dabei ist ⟨*Spezifikation*⟩ eine *key-value*-Liste. Die wichtigsten *keys* sind `name` und `description`, über welche der Name und die Beschreibung des zu definierenden Eintrags festgelegt werden. Über den Befehl `\gls{`⟨*Label*⟩`}` kann dann der zuvor definierte Begriff im Text ausgegeben werden. Beispiel gefällig? Der Hund ist der beste Freund des Menschen.

6.11.2 Abkürzungsverzeichnis erstellen

Um ein Abkürzungsverzeichnis verwenden zu können, muss `glossaries` mit der Option `acronym` geladen werden. Eine Abkürzung kann dann in der Präambel über den Befehl `\newacronym{`⟨*Label*⟩`}{`⟨*Abkürzung*⟩`}{`⟨*Ausgeschrieben*⟩`}` definiert werden. Über den Befehl `\gls{`⟨*Label*⟩`}` kann dann die zuvor definierte Abkürzung im Text ausgegeben werden. Dabei stellt LaTeX dann automatisch sicher, dass die Abkürzung bei der ersten Erwähnung im Text ausgeschrieben wird. Bei allen späteren Erwähnungen wird dann nur noch die Abkürzung ausgegeben. Beispiel gefällig? Das ist eine support vector machine (SVM). Und dort ist gleich noch eine SVM.

6.11.3 Symbolverzeichnis erstellen

Um ein Symbolverzeichnis verwenden zu können, muss `glossaries` mit der Option `symbols` geladen werden. Ein Symbol kann dann in der Präambel über den Befehl `\newglossaryentry{`⟨*Label*⟩`}{`⟨*Spezifikation*⟩`}` definiert werden. Der Befehl

ist also genau derselbe wie beim Glossar. Zusätzlich muss für ⟨*Spezifikation*⟩ noch `type=symbols` angegeben werden. Über den Befehl `\gls{`⟨*Label*⟩`}` kann dann das zuvor definierte Symbol im Text ausgegeben werden. Beispiel gefällig? Die Kraft $\vec{F}$ ist gemäß der folgenden Gleichung definiert:

$$\vec{F} = m \cdot \vec{a}$$

6.12 Verweise

Setzen Sie in Ihrem Quellcode Marken mit dem „`\label`“-Befehl. Aus der Platzierung geht hervor, auf welche Nummerierung sich die Marke bezieht, also etwa Gliederungsebene (siehe Abschnitt 6.2), Tabelle (siehe Abschnitt 6.7), Abbildung (siehe Abschnitt 6.6) oder Gleichung (siehe Abschnitt 6.9). Alle genannten werden nämlich separat nummeriert, das kann am Anfang etwas gewöhnungsbedürftig sein.

Auf die markierten Stellen können Sie dann mit dem „`\ref`“-Befehl verweisen, wobei der eben nur die passende Nummer liefert. Die passende Bezeichnung, z. B. „Abbildung“, müssten Sie dann selbst ergänzen. Daher haben wir hier das Paket `cleveref` eingebunden, das uns den zuletzt genannten Schritt automatisiert.

Auf jede Abbildung und jede Tabelle muss im Text verwiesen werden, es dürfen keine nummerierten Umgebungen einfach „in der Luft hängen“. Im Gegensatz dazu müssen Abschnitte und Gleichungen nicht alle explizit referenziert werden. Sie können aber Ihren Leser*innen helfen, wenn Sie über sinnvolle Verweise nachdenken.

6.13 Besondere Abstände und Zeichen

An Leerzeichen kann grundsätzlich ein Zeilenumbruch (oder sogar Seitenumbruch) erfolgen. In manchen Fällen möchte man das vermeiden, u. a., weil Zeilen nicht mit Zahlen beginnen sollten. Ein typisches Beispiel ist „Lange Straße 123“. Hier benötigt man hinter „Straße“ ein sog. geschütztes Leerzeichen, das mit einer Tilde erzeugt wird.

Genauso wie Gleitumgebungen optimal verteilt werden (vgl. Abschnitt 6.1), werden auch horizontale Abstände zwischen Wörtern und Sätzen in LaTeX in jeder Zeile dynamisch angepasst. Dabei werden alle Punkte als Satzende interpretiert. Bei Abkürzungen wie „z. B.“ sieht das nicht schön aus, der Leerzeichen-Abstand ist zu groß. Hier muss in der Mitte manuell ein halbes Leerzeichen erzeugt werden mit „`\,`“. Wenn Ihnen das Tippen solcher Konstrukte zu umständlich erscheint, können Sie

sich eigene Kommandos wie „`\zb`“ definieren. Zum Definieren eigener Befehle vgl. Abschnitt 6.9.

Auch bei waagerechten Strichen gibt es, wie bei Leerzeichen, unterschiedliche Längen. Für Gedankenstriche – solche hier – oder wenn „bis“ gemeint ist (wie in 14:00–16:00), reicht der einfache Bindestrich (Minuszeichen) nicht aus, das passende Teichen wird in LaTeX einfach durch ein Doppel-Minus erzeugt.

Problematisch im Quellcode sind alle Zeichen, die in LaTeX eine Funktion haben: Prozentzeichen %, kaufmännisches Und &, Unterstrich _, geschweifte Klammern { ... } sind typische Beispiele. Diese müssen im Quelltext mit einem *Backslash* eingegeben werden, sonst erhält man Fehlermeldungen.

6.14 Wahl der Grundschriftart

Standardmäßig verwendet LaTeX für den Fließtext eine serifenbehaftete Schriftart. Für gedruckte Arbeiten sind serifenbehaftete Schriften vorteilhaft, weil die Serifen die Grundlinie betonen und somit das Auge beim Rücksprung am Zeilenende zum Beginn der nächsten Zeile unterstützt. Außerdem führen die unterschiedlichen Strichstärken zu eindeutigeren Wortbildern und unterstützen somit den Leseprozess.

Wird solch ein Dokument jedoch an einem alten Monitor mit geringer Auflösung betrachtet, so kann es sein, dass die feinen Serifen nicht mehr vernünftig dargestellt werden. In solch einem Fall kann es vorteilhaft sein, eine serifenlose Schrift zu verwenden. Auch kann es sein, dass serifenlose Schriften aus Gründen der Barrierefreiheit bevorzugt werden.

In diesem Fall kann mit dem Befehl `\renewcommand{\familydefault}{\sfdefault}` eine serifenlose Schrift als Grundschriftart festgelegt werden. Wer `lualatex` zum Kompilieren sowie das Paket `fontspec` verwendet, kann außerdem auf alle verfügbaren Schriften zugreifen. Ist auf dem Rechner die Schriftart Arial vorhanden, so kann mit dem Befehl `\setsansfont{Arial}` die Schriftart Arial als serifenlose Schrift festgelegt werden. Am Ende der Datei `definitions.tex` sind die beiden besagten Zeilen zu finden und müssen bei Bedarf nur auskommentiert werden.

6.15 Metadaten für den pdf-Betrachter

Manche pdf-Betrachter können zusätzliche Metadaten, wie Name des Autors, Titel des Dokuments (auch abweichend vom Namen der Datei) oder Schlüsselbegriffe anzeigen. Mit dem Paket `hyperref` lassen sich diese Metadaten mit dem Befehl

`\hypersetup{`⟨*Einsellungen*⟩`}` konfigurieren. Dabei ist ⟨*Einstellungen*⟩ eine *key-value*-Liste. Die wesentlichsten *keys* sind `pdfauthor`, `pdftitle` und `pdfkeywords`. Die Bedeutung dieser *keys* ist selbsterklärend.

Außerdem werden Links im Dokument (bei Verwendung des Pakets `hyperref`) in manchen pdf-Betrachtern als farbige Kästchen hervorgehoben. Diese farbigen Kästchen erscheinen natürlich nicht im gedruckten Dokument. Sie dienen lediglich als Hilfe, dass man nicht „auf gut Glück“ mit dem Cursor über das Dokument fahren muss, bis man den Link gefunden hat. Wenn die farbigen Kästchen stören, so können diese in `\hypersetup` mit `hidelinks` deaktiviert werden.

6.16 Wechsel zwischen ein- und doppelseitigem Layout

Diese Vorlage ist für ein einseitiges Layout optimiert. Dabei sind die linken und rechten Ränder jeweils gleich groß auf allen Seiten, neue Kapitel beginnen unmittelbar auf der nächsten Seite. Wird das fertige pdf-Dokument am Computer betrachtet, sieht das genau richtig aus. Soll das Dokument hingegen doppelseitig ausgedruckt werden, so kann das Layout noch etwas angepasst werden: Typischerweise sind die inneren Ränder dann etwas schmaler als die äußeren Ränder. Und neue Kapitel beginnen jeweils auf einer neuen, rechten Seite – was zu einzelnen Vakatseiten zwischen den Kapiteln führen kann. Für doppelseitig ausgedruckte Dokumente sieht das dann besser aus.

Um das doppelseite Layout zu aktivieren, genügt es bereits, die Auskommentierung der Option `twoside` im optionalen Argument von `\documentclass` zu entfernen.

6.17 Kompilieren

Das Erstellen (Kompilieren) von großen Dokumenten mit LaTeX kann verhältnismäßig lange dauern. Da man i. d. R. nur an wenigen Stellen gleichzeitig arbeitet, kann es daher sinnvoll sein, übrige Teile auszukommentieren. Das geht besonders leicht, wenn man Text in getrennte Dateien auslagert und mit dem „`\input`“- und / oder „`\include`“-Befehl einbindet. So bleibt auch das Hauptdokument übersichtlich.

Grundsätzlich sollte das Ziel sein, dass Ihr Dokument ohne Warnungen kompiliert. Am besten kümmert man sich regelmäßig darum, entsprechende Probleme zu beheben.

Eine typische Warnung ist „*Reference ... undefined*“. Vielleicht verschwindet sie beim nochmaligen Erstellen, denn erst dann sind ggf. neue Positionen bekannt. Wenn

diese Warnung bleibt, muss das Problem unbedingt behoben werden, sonst haben Sie irgendwo Fragezeichen im Text stehen.

Warnungen, die sich auf zu volle Boxen beziehen, sind teilweise schwieriger zu verstehen und / oder zu beheben. Im `draft`-Modus (vgl. Abschnitt 6.18) werden die zugehörigen Stellen genau markiert, das kann eine große Hilfe sein. Gegen zu lange Zeilen hilft teilweise, Trennstellen zu markieren (vgl. Abschnitt 6.4). Sonst muss ggf. ein Satz minimal umformuliert werden.

Der `draft`-Modus hat drüber hinaus den Vorteil, dass das Kompilieren schneller geht (s. o.), dafür werden für Abbildungen nur Platzhalter eingefügt.

6.18 Diese Vorlage

In dieser Vorlage wird KOMA-Script verwendet, eine „Sammlung von Klassen und Paketen für LaTeX“[1], die insbesondere das Erstellen von deutschen Texten mit den entsprechenden üblichen typographischen Standards unterstützt.

Dokumente mit LaTeX zu erstellen ist ganz ähnlich wie Programmieren. Ein Beispiel: Überall dort, wo ein Absatz entstehen soll, haben wir in unserem „Quellcode“ den Befehl „`\par`“ benutzt. Was dieser Befehl genau tut, wird durch dessen Implementierung festgelegt. Und diese ergibt sich hier sozusagen aus dem Parameter `parskip` der Dokumentklasse.

Andere Einstellungen, die direkt in der Dokumentklasse erfolgen können, betreffen z. B. die Schriftgröße, die Bindungskorrektur (`BCOR`) und die Größe von Überschriften (`headings`). Im Prinzip kann man auch die Größe der Ränder mit dem `DIV`-Parameter beeinflussen, davon wird aber abgeraten. Die Ränder werden automatisch so eingestellt, dass Zeilen eine Länge haben, die gut zu lesen ist.

Hier haben wir außerdem die Option `twoside` gewählt, für beidseitigen Druck. Daher sind die Ränder außen auf geraden und ungeraden Seiten unterschiedlich. Falls Sie Ihr Dokument am Ende einseitig drucken wollen, stellen Sie das bitte um.

Nach der Festlegung der Dokumentklasse haben wir in der sog. Präambel einige Pakete eingebunden. Zum Beispiel das `scrlayer-scrpage`-Paket, mit dem wir das Aussehen der Fuß- und Kopfzeile definieren können. Diese Vorlage wurde so eingerichtet, dass in den Kopfzeilen einer Doppelseite oben links immer die aktuelle Kapitel-Überschrift und oben rechts die aktuelle Abschnitt-Überschrift angezeigt wird (siehe `definitions.tex`).

[1] `https://komascript.de`

In der vorliegenden Vorlage werden einige Pakete eingebunden. Im Folgenden wird die Funktion der wichtigsten davon kurz erläutert.

`fontspec` Erlaubt die freie Wahl der Schriftart (funktioniert aber nur bei Kompilation mit `lualatex`)

`babel` Erlaubt das Umstellen der Standard-Sprache auf Deutsch

`selnolig` Sorgt für automatisch korrekt gesetzte Ligaturen (funktioniert aber nur bei Verwendung von `fontspec` und somit auch `lualatex`, übernimmt automatisch die Spracheinstellung von `babel`)

`microtype` Optimiert das Aussehen des Textes (Satzspiegel)

`csquotes` Sorgt für automatisch korrekt gesetzte Anführungszeichen (übernimmt automatisch die Spracheinstellung von `babel`)

`tikz` und `pgfplots` Damit können Abbildungen direkt im Quellcode erzeugt werden, vgl. Abschnitt 6.6.2

`hyperref` Anklickbare Links im PDF

`biblatex` Verbesserte Quellenangaben und -verzeichnis

`amsmath` und `amssymb` Große Erweiterung der Möglichkeiten, mathematische Inhalte darzustellen

`listings` Zur Darstellung von Quellcode, vgl. Abschnitt 6.10

`cleveref` Vereinfacht das Einfügen von Verweisen (siehe Abschnitt 6.12)

`glossaries` Komfortables Erstellen eines Abkürzungsverzeichnis

Literatur

[1] Mehmet Bayram, formilo, *Popularität und Statistiken der PDF*. Adresse: https://www.formilo.com/pdf-formulare/einfuehrung/popularitaet-statistiken/ (besucht am 19.12.2023).

[2] Wikipedia, *Portable Document Format*, 2023. Adresse: https://de.wikipedia.org/wiki/Portable_Document_Format (besucht am 19.12.2023).

[3] Oliver Helfrich, KOFAX, *30 Jahre PDF - Ein Geschenk, das uns immer wieder neu überrascht*, Blogeintrag, 2023. Adresse: https://www.kofax.de/learn/blog/30-years-of-pdf (besucht am 19.12.2023).

[4] Wikipedia, *Offener Standard*, 2023. Adresse: https://de.wikipedia.org/wiki/Offener_Standard (besucht am 20.12.2023).

[5] Wikipedia, *Seitenbeschreibungssprache*, 2021. Adresse: https://de.wikipedia.org/wiki/Seitenbeschreibungssprache (besucht am 20.12.2023).

[6] Adobe Systems Incorporated, *PostScript LANGUAGE REFERENCE third edition*, 1999. Adresse: https://web.archive.org/web/20090419181826/http://www.adobe.com/devnet/postscript/pdfs/PLRM.pdf (besucht am 19.12.2023).

[6] Wikipedia, *PostScript*, 2023. Adresse: https://de.wikipedia.org/wiki/PostScript (besucht am 19.12.2023).

[7] Oliver Helfrich, KOFAX, *30 Jahre PDF - Ein Geschenk, das uns immer wieder neu überrascht*, Blogeintrag, 2023. Adresse: https://www.kofax.de/learn/blog/30-years-of-pdf (besucht am 19.12.2023).

[8] Oliver Helfrich, KOFAX, *30 Jahre PDF - Ein Geschenk, das uns immer wieder neu überrascht*, Blogeintrag, 2023. Adresse: https://www.kofax.de/learn/blog/30-years-of-pdf (besucht am 19.12.2023).

[9] Oliver Helfrich, KOFAX, *30 Jahre PDF - Ein Geschenk, das uns immer wieder neu überrascht*, Blogeintrag, 2023. Adresse: https://www.kofax.de/learn/blog/30-years-of-pdf (besucht am 19.12.2023).

[10] GitHub, *Understanding GitHub Actions*. [Online]. Available: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions (accessed: June 11 2023).

[11] w3schools, *JavaScript Promises*. [Online]. Available: https://www.w3schools.com/js/js_promise.asp (accessed: June 8 2023).

[12] Imperva, *Cross site request forgery (CSRF) attack*. [Online]. Available: https://www.imperva.com/learn/application-security/csrf-cross-site-request-forgery/ (accessed: June 10 2023).

[13] PortSwigger, *HTTP Host header attacks*. [Online]. Available: https://portswigger.net/web-security/host-header (accessed: June 10 2023).

[14] Mozilla, *Cross-Origin Resource Sharing (CORS)*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS (accessed: June 10 2023).

[15] Rowan Merewood, *SameSite cookies explained*. [Online]. Available: https://web.dev/samesite-cookies-explained/ (accessed: June 10 2023).

[16] django, *HTTP Strict Transport Security*. [Online]. Available: https://docs.djangoproject.com/en/4.2/ref/middleware/#http-strict-transport-security (accessed: June 10 2023).

[17] mohanpedala, *set -e, -u, -o, -x pipefail explanation* . [Online]. Available: https://gist.github.com/mohanpedala/1e2ff5661761d3abd0385e8223e16425?permalink_comment_id=3935570 (accessed: June 10 2023).

[18] Django, *Migrations*. [Online]. Available: https://docs.djangoproject.com/en/4.2/topics/migrations/ (accessed: June 11 2023).

[19] Docker, *Docker run reference*. [Online]. Available: https://docs.docker.com/engine/reference/run/ (accessed: June 11 2023).

[20] Docker, *Overlay network driver*. [Online]. Available: https://docs.docker.com/network/drivers/overlay/ (accessed: June 11 2023).

[21] Docker, *Use a volume with Docker Compose*. [Online]. Available: https://docs.docker.com/storage/volumes/ (accessed: June 11 2023).

[22] Docker, *Performance tuning for volume mounts (shared filesystems)*. [Online]. Available: https://docs.docker.com.zh.xy2401.com/v17.12/docker-for-mac/osxfs-caching/ (accessed: June 11 2023).

[23] Traefik, *Configuration Introduction*. [Online]. Available: https://doc.traefik.io/traefik/v2.0/getting-started/configuration-overview/ (accessed: June 12 2023).

Anhang

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Anmerkung: In einigen Studiengängen findet sich die Erklärung unmittelbar hinter dem Deckblatt der Arbeit.

Köln, 04.03.2024 Unterschrift