

Einführung in die maschinennahe, imperative,
funktionale, relationale und objektorientierte
Programmierung

-

EMIFROP 0.28

Markus Alpers
B.Sc. Media Systems und Ausbilder f. Industriekaufleute

31. Juli 2016

Inhaltsverzeichnis

I	Einfache Einführung in die imperative Programmierung	16
1	Das ist Programmieren (wirklich)	17
1.1	Das ist an diesem Buch anders	18
1.2	Zentrale Begriffe und Konzepte beim Programmieren	19
1.2.1	Der Begriff des Programmierens	19
1.2.2	Paradigmen	20
1.2.3	Middleware, Framework, Bibliothek	26
1.2.4	IDE - Entwicklungsumgebung	26
1.2.5	Dokumentation	27
1.2.6	SCM / Versionskontrolle	28
1.2.7	Software Engineering / Softwareentwicklung	28
1.2.8	App-Entwicklung	29
1.3	Informatik versus Programmierung, Studium und Arbeit . .	33
1.3.1	Informatik und Programmierung	33
1.3.2	Informatik: Uni versus HAW (FH)	35
1.3.3	Informatik und Programmieren im Beruf	35
1.4	Zusammenfassung	44
2	Nachrichtentechnik und Programmierung	47
2.1	Nachrichtentechnik – So kommen Nullen und Einsen in den Rechner.	47
2.2	Codierung – Was steht wofür?	49
2.3	Informatik und Nachrichtentechnik - Die zankenden Geschwis- ter	51
2.4	Von der Nachrichtentechnik zu logischen Gattern	53
2.5	Weiter zur Maschinensprache	56
2.6	Maschinennahe Programmierung – Assembler	57
2.7	Zusammenfassung	59
3	Vorbereitung fürs Programmieren	60
3.1	Assembler und C – Vorbereitung für die maschinennahe und imperative Programmierung	62
3.2	Pascal - Eine weitere imperative Sprache	64

3.3	Scheme - Eine funktionale Programmiersprache	65
3.4	PROLOG - Logische Programmierung	65
3.5	Java - Vorbereitung für die Programmierung in Java	65
3.6	HTML, PHP und MySQL – Vorbereitung für die Entwicklung verteilter Anwendungen	69
3.7	Textverarbeitung mit LaTeX	71
3.8	Alle - Vorbereitung für die Teamarbeit	71
3.9	Alle – Nutzung des Netzlaufwerks zur Speicherung eigener Daten	73
3.10	Änderungen	73
4	Ausgewählte Programmiersprachen	74
4.1	Nach B kam C	74
4.2	C++ : C mit Objektorientierung	76
4.2.1	Objektorientierung nach Alan Kay	76
4.2.2	Objektorientierung nach Lieschen Müller	77
4.3	Java – C++ ohne maschinennähe	78
4.4	Verteilte Anwendungen	81
4.4.1	Entwicklung von Webanwendungen – MySQL und PHP versus Ruby on Rails	82
4.5	Konzepte bei der Programmierung	85
4.5.1	Dynamisch versus statisch – Ruby und Python versus C und Java	86
4.5.2	Typisierung von Daten	86
4.5.3	Funktionen – Dynamisch versus statisch	89
4.6	First-class Objects	90
4.7	Zusammenfassung	91
5	Grundlagen verteilter Anwendungen	93
5.1	Internet, WWW und HTML	95
5.1.1	HTTP und HTTPS	97
5.2	Funktionalitäten unserer Webanwendung	98
5.2.1	Erste Schritte zur Webanwendung	99
5.2.2	Projektdokumentation und Arbeitsumgebung	99
5.2.3	Erste Übung	100
5.2.4	Mehr zu den Leistungsnachweisen für alle	102
5.2.5	Mehr zu den Leistungsnachweisen für MS-Studis	102
5.2.6	Mehr zum Leistungsnachweis für MT-Studis	103
5.3	Technische Grundlagen verteilter Anwendungen	104
5.4	Erste Gehversuche mit Server und Client	104
5.4.1	Eine erste Webanwendung mit PHP	108
5.5	Programmierung von Webanwendungen	110
5.5.1	MVC – Das Model View Controller Pattern	112
5.6	Das semantische Web	114

5.6.1	Syntax und Semantik	114
5.6.2	Semantik und das WWW	116
5.6.3	DOM und Microdata	117
5.6.4	Zusammenfassung	118
5.7	Übertragung der Dateien auf einen Webserver im Netz . . .	119
6	Einstieg in HTML 5	122
6.0.1	Das ist HTML	123
6.0.2	Erster HTML-Quellcode	124
6.0.3	Struktur eines HTML-Dokuments	129
6.1	Barrierefreiheit, Internationalisierung und Lokalisierung . .	136
6.1.1	Barrierefreiheit	136
6.1.2	Internationalisierung (kurz i18n) und Lokalisierung (kurz l10n)	138
6.2	Der head-Container, Meta-Daten und Attribute	138
6.2.1	Entitys - Umlaute und andere Sonderzeichen	140
6.3	Mehr Grundlagen in HTML	141
6.3.1	Mehr Auslagerung von Code	141
6.3.2	Verwendung von Escape-Sequenzen	142
6.3.3	HTML5: Anführungszeichen sind weitestgehend op- tional	143
6.3.4	Zusammenfassung	144
6.4	Strukturen von HTML5-Dokumenten	144
6.4.1	header, footer, main und aside	144
6.4.2	article und section	145
6.4.3	h1 bis h5	146
6.4.4	p	146
6.4.5	Aufgabe	147
6.5	Polyfills	147
6.5.1	Einbindung von Polyfills - script-Container	148
6.6	Zusammenfassung	148
6.6.1	Hausaufgabe	151
6.7	Hyperlinks	151
6.7.1	Anker	152
6.7.2	Links	153
6.7.3	Verlinkungen als expliziter Download	154
6.7.4	Hausaufgabe:	154
6.7.5	URLs – absolute und relative Adressen	155
6.8	Formulare	156
6.8.1	Elemente eines Formulars	157
6.8.2	Das name-Attribut und das id-Attribut – Sonderfälle in Formularen	158
6.8.3	Container für Formulare	159

6.8.4	Container für die Gruppierung und Zuordnung von Eingabeelementen	168
6.8.5	Zusammenfassung	169
6.9	Multimediale Inhalte einfügen	169
6.9.1	Bilder	170
6.9.2	figure und figcaption	171
6.9.3	figcaption für Fortgeschrittene	171
6.10	Weitere multimediale Formate	172
6.10.1	Einbindung eigener und frei verfügbarer Videodateien	172
6.10.2	Anpassungsmöglichkeiten für den Video-Player . . .	173
6.10.3	Einbindung von geschützten Inhalten (Stichwort: DRM)	175
6.10.4	Der audio-Container	175
6.10.5	Close Captions, Untertitel, Einbindung von Webcams usw.	176
6.10.6	Hinweis bezüglich Flash und ähnlichen Formaten . .	176
6.10.7	Hausaufgabe	177
6.11	Weitere Formatierungen und Möglichkeiten in HTML	177
6.11.1	Spoiler und andere ausklappbare Texte	178
6.11.2	Zeitangaben	179
6.11.3	Hervorhebung von Texten	181
6.11.4	Unterdrückung von Übersetzungen für Textpassagen	181
6.11.5	Aufzählungen (Ordered und Unordered Lists)	183
6.11.6	Glossare (Description Lists)	184
6.11.7	Tabellen (table)	184
6.11.8	Microdata	185
6.11.9	Validator für Microdata	189
6.12	Zusammenfassung	189
7	ML, Teil 2 - Textverarbeitung mit LaTeX	191
7.1	Grundlagen	192
7.1.1	Struktur eines LaTeX-Dokuments	192
7.1.2	Einfache Präambel	193
7.2	Text, Sonderzeichen und Formeln	195
7.2.1	Sonderzeichen	196
7.2.2	Formeln	197
7.3	Umgebungen	198
7.4	Inhaltsverzeichnis, Kapitel und Abschnitte	198
7.4.1	Kapitel, Abschnitte usw.	198
7.5	Auslagern von Kapiteln	199
7.6	Titelblatt und Glossar	199
7.7	Glossar	200
7.7.1	Stichwörter und Bezüge festlegen	201
7.7.2	Weitere Verzeichnisse	201
7.8	Listen und Tabellen	201

7.8.1	Tabellen	202
7.9	figures	204
7.9.1	Bilddateien in LaTeX	205
7.10	Referenzen und Labels	206
7.11	Boxen	206
7.12	Abschluss	206
8	Gestaltung mit CSS	207
8.1	Einbindung von CSS in HTML	207
8.2	Programmierung einer CSS-Datei	208
8.3	Programmierung von Formatierungen	208
8.3.1	Mehr über Farben	209
8.3.2	Zwei weitere Properties in Bezug auf Farben	210
8.4	Ein wenig Form	211
8.5	Definition von CSS für einzelne Container eines Typs	212
8.6	Mehr zu CSS	213
9	Dynamische Webentwicklung	214
9.1	PHP 5	216
9.1.1	Vorbereitung des Computers, um in PHP programmieren zu können.	218
9.2	PHP-Programme in HTML-Dokumenten integrieren	218
9.3	PHP-Code innerhalb von HTML – Erstes Beispiel	220
9.3.1	Kommentare	221
9.4	Überarbeitung: Grundlagen der Programmierung in PHP	224
9.4.1	Variablen	226
9.4.2	Datentypen	228
9.4.3	Operationen	230
9.4.4	Anonyme Variablen	231
9.4.5	Operationen für Text-Variablen	232
9.4.6	Nachteile der dynamischen Typisierung	232
9.5	Überarbeitung: Kontrollstrukturen und boolesche Ausdrücke	233
9.5.1	Grundidee	234
9.5.2	Vergleichsoperationen	234
9.5.3	Conditionals 1 – if-then-else	235
9.5.4	Conditionals 2 – Switch-Cases	236
9.6	Überarbeitung: Funktionen	240
9.6.1	Was ist ein Parameter?	241
9.6.2	Eigene Funktionen programmieren	243
9.7	Überarbeitung: Datenstrukturen	244
9.7.1	Felder bzw. Arrays	245
9.8	Überarbeitung: Wiederholungen von Programmteilen (Schleifen und Rekursionen)	248
9.8.1	Rekursionen	250

9.8.2 Schleifen	254
9.9 Überarbeitung: Hinweis bezüglich objektorientierter Softwareentwicklung in PHP	256
9.10 Überarbeitung: Auswertung von Nutzereingaben in HTML durch PHP	256
9.10.1 Aufgabe	258
9.11 Überarbeitung: Abschluss	258
10 Langfristige Datenspeicherung mit MySQL 5.6	260
II Fortsetzung für Studierende der Informatik	261
11 Einführung in den Aufbau von ARMv6-Prozessoren am Beispiel eines Cortex-M0-Prozessors	262
11.1 Mikroprozessoren und eingebettete Systeme	264
11.2 SoC, Mikrocontroller und die Umgebung von Mikroprozessoren	265
11.2.1 Das Olimex LPC1114 Entwicklungsboard	267
11.3 Grundlagen des Cortex-M0	268
11.4 Die Register des Cortex-M0	269
11.4.1 R0 bis R12 – Die Allzweckregister	269
11.4.2 Einschub: LSB = 1 für Thumb State	270
11.4.3 SP – Der Stack Pointer (alternativ R13)	270
11.4.4 LR – Das Link Register (alternativ R14)	274
11.4.5 PC – Der Program Counter (alternativ R15)	274
11.4.6 xPSR – Das kombinierte Statusregister	275
11.4.7 PRIMASK – Das Interrupt-Maskierungs-Register	276
11.4.8 Weitere Register	276
11.4.9 Zusammenfassung	276
11.5 Die Aufteilung des Speichers	277
11.5.1 Die Aufteilung des Speichers des Cortex-M0	277
11.5.2 Weitere Bezeichnungen für Speichereinteilungen	278
11.5.3 Anbindung von externem Speicher über den SPI Controller	279
11.6 Exceptions und Interrupts	279
11.6.1 Exception Types	281
11.6.2 Initialisierung von IRQs	282
11.6.3 IRQs im Quellcode	283
11.6.4 Codebeispiel	283
11.7 NVIC – Der Nested Vectored Interrupt Controller	285
11.7.1 Memory Mapping	285
11.8 Das SPI – Serial Peripheral Interface	286
11.9 Grundlagen der eigentlichen Programmierung	286

11.9.1	Die Startsequenz des Cortex-M0	286
11.9.2	Prozesse – So planen Sie Ihr Programm	287
11.9.3	Ein- und Ausgaben	289
11.9.4	Eingebettete Systeme und Debugging im laufenden Betrieb	291
11.9.5	Mangelnde Portabilität und das CMSIS	292
11.9.6	Das ABI – Die Klassenbibliothek für die Cortex-Programmierung	293
11.10	Wie Sie die Software in den Mikrocontroller bekommen . . .	293
11.10.1	Die Verbindung zwischen Entwicklungsboard und Rechner herstellen	293
11.10.2	Vom Quellcode zum Programm	294
11.10.3	Program Image flashen	295
11.11	Zugriff auf die Peripherie bei einem Cortex-M0	296
11.12	Der CMSIS – Cortex Microcontroller Software Interface Standard	297
12	Grundlagen imperativer Programmierung	298
III	Einführung in die objektorientierte Programmierung und Softwareentwicklung	299
13	Einführung in die Objektorientierung mit Java	300
13.1	Java 8 - Grundlegende Struktur	301
13.1.1	Die Java API	302
13.1.2	JVM - Java Virtual Machine Technology	302
13.1.3	Base Libraries	304
13.1.4	Integration Libraries	304
13.1.5	User Interface Libraries	304
13.1.6	Deployment	304
13.1.7	Weitere Pakete im JDK	305
13.2	Base Libraries im Detail	305
13.2.1	lang	305
14	Java 8 Basics	308
14.1	Eingaben und Ausgaben	308
14.1.1	Erzeugen eines Objekts	309
14.1.2	Benutzen von Objekten innerhalb eines Java-Programms	310
14.1.3	Hello, world.	311
14.1.4	Hello, (Name des Benutzers)	314
	Stichwortverzeichnis	316

Hinweis bezüglich diskriminierender Formulierungen

In diesem Text wurde darauf geachtet Formulierungen zu vermeiden, die diskriminierend verstanden werden können. Im Sinne der Lesbarkeit wurden dabei Formulierungen wie „Informatiker und Informatikerinnen“ durch „InformatikerInnen“ (mit großem i) ersetzt. An anderen Stellen habe ich Formen wie eine/einer durch eineR zusammengefasst. Hier berufe ich mich auf den Artikel „Sprache und Ungleichheit“ der Bundeszentrale für politische Bildung, kurz BpB, vom 16. April 2014, insbesondere auf den Absatz „Zum Umgang mit diskriminierender Sprache“, online abrufbar unter:

<http://www.bpb.de/apuz/130411/sprache-und-ungleichheit?p=all>

Sollten Sie dennoch Formulierungen entdecken, die diesem Anspruch nicht entsprechen, möchte ich Sie bitten, mir eine entsprechende Nachricht zu senden, denn es ist mir wichtig, Ihnen mit diesem Buch eine wertvolle Unterstützung beim Start in die faszinierende Welt der Informatik zu bieten. Das sollte nicht durch verletzte Gefühle in Folge missverständlicher Formulierungen torpediert werden.

Sie erreichen mich unter markus.alpers@haw-hamburg.de.

Hinweis zur Lizenz

Dieses Buch wird in Teilen unter der Lizenz *CC BY-SA 3.0 DE* veröffentlicht. Das bedeutet, dass Sie die entsprechenden Teile z.B. kopieren dürfen, so lange der Name des Autors erhalten bleibt. Sie dürfen diese auch in eigenen Werken weiterverwenden, ohne dafür z.B. eine Lizenzgebühr zahlen zu müssen. Dennoch müssen Sie auch hier bestimmte Bedingungen einhalten. Eine davon besteht darin, dass eine solche Veröffentlichung ebenfalls unter dieser Lizenz erfolgen muss. Sinn und Zweck solcher Lizenzen besteht darin, dass geistiges Eigentum frei sein und bleiben soll, wenn derjenige, der es erschaffen hat das wünscht. Und es ist mein Wunsch, dass so viele Menschen wie möglich von den Erklärungen in diesem Text profitieren.

Der vollständige Wortlaut der Lizenz ist auf folgender Seite nachzulesen. Dort erfahren Sie dann auch, welche Bedingungen einzuhalten sind:

<https://creativecommons.org/licenses/by-sa/3.0/de/>

Alle Teile des Buches, die ich unter der Lizenz *CC BY-SA 3.0 DE* veröffentliche enthalten am Anfang diesen Abschnitt „Hinweise zur Li-

zenz““. Wenn Sie einen Teil finden, in dem diese „Hinweise zur Lizenz“ nicht zu finden ist, dann dürfen Sie für den persönlichen Gebrauch dennoch Kopien davon anfertigen und Sie dürfen diese Kopien außerhalb von kommerziellen Projekten frei verwenden.

Hinweis zur Verwendbarkeit in wissenschaftlichen Arbeiten

Bitte beachten Sie dabei aber, dass die Verwendung dieses Textes im Rahmen wissenschaftlicher Publikationen zurzeit aus anderen Gründen problematisch ist:

- Wie viele andere Quellen, die frei im Internet verfügbar sind, wurde auch dieser Text bislang nicht durch einen nachweislich entsprechend qualifizierten Lektor verifiziert. Damit genügen Zitate aus diesem Band streng genommen noch nicht den Ansprüchen wissenschaftlicher Arbeiten.
- Weiterhin fehlen in diesem Buch häufig die für eine wissenschaftliche Arbeit nötigen Quellenangaben. Wann immer Sie in einer wissenschaftlichen Arbeit eine Behauptung aufstellen, müssen Sie diese durch einen Beleg oder Beweis untermauern. Ein solcher Beleg/Beweis kann zum einen eine andere wissenschaftliche Arbeit sein, in der bewiesen wurde, dass die Behauptung den Tatsachen entspricht oder es muss ein eigenständiger Beweis für die Aussage sein.

Bitte beachten Sie außerdem, dass dieses Buch eine Konvention nutzt, die in wissenschaftlichen Arbeiten verpönt ist: Wenn in einer wissenschaftlichen Arbeit ein Begriff hervorgehoben wird, dann wird dazu kursive Schrift verwendet. In diesem Buch verwende ich dagegen Fettdruck, da es vielen Menschen schwer fällt, einen kursiv gedruckten Begriff schnell zu finden und ich mir wünsche, dass Sie es möglichst effizient auch als Nachschlagewerk nutzen können.

Dieses Buch ist allerdings auch **nicht als wissenschaftliche Arbeit**, also als Ergebnis einer Forschung über Programmiersprachen, **sondern als Lehrwerk für den Einstieg in die Programmierung gedacht**. Es dient somit nicht dazu, Ihnen die theoretischen Grundlagen der Programmierung zu vermitteln, sondern dazu, ihnen eine fundierte Unterstützung beim Einstieg in die Programmierung anzubieten.

Zielgruppe und Vorwort

Dieses Buch habe ich erstellt, um Studierenden der Studiengänge Media Systems (entspricht Medieninformatik an anderen Hochschulen) und Medientechnik an der HAW Hamburg den Einstieg ins Programmieren zu erleichtern. Deshalb finden sich hier teilweise Anmerkungen für die Studierenden der beiden Studiengänge, die aber in ähnlicher Form für Studierenden der Informatik und der Elektrotechnik gelten. Da es jedoch so formuliert ist, dass es für Studienanfänger ohne Programmiererfahrung geeignet ist, kann jede/r Studierende es gut nutzen, um sich in die Programmierung einzuarbeiten. Wenn die Version 1.0 abgeschlossen ist wird es eine **grundlegende Einführung** in die Konzepte (Paradigmen), der maschinennahen, der imperativen, der funktionalen, der relationalen und der objektorientierten Programmierung in den Ausprägungen prototypbasiert und klassenbasiert sein. Zusätzlich behandelt es den Einstieg in die Entwicklung verteilter Anwendungen.

Wie alle Lehrbücher für Studierende setzt es eines voraus: Wenn Sie es nutzen wollen, dann funktioniert das dann, und ausschließlich dann, wenn Sie zusätzlich zum Lesen zwei Dinge tun: Zum einen müssen Sie ständig kontrollieren, ob Sie jeden **neuen Begriff wirklich verstanden** haben und prüfen, wie er im Zusammenhang mit dem bisher Gelernten steht und zum anderen **müssen Sie tatsächlich programmieren**.

Was Sie hier nicht finden sind zum einen alle Varianten der Programmierung, die im Kern aus der Elektrotechnik entstanden sind oder für deren Verständnis Sie die Grundlagen kontinuierlicher Systeme beherrschen müssen. In der Informatik werden diese Bereiche als **Technische Informatik** bezeichnet. Das schließt beispielsweise die Programmierung von Steuer- und Regelsystemen, also insbesondere **SPSe** und **FPGAs** ein.

Damit sind wir auch schon bei einem ersten Missverständnis das zwischen InformatikerInnen einerseits und NaturwissenschaftlerInnen, IngenieurInnen und TechnikerInnen (kurz **INT-Akademiker**) existiert: Was in der Informatik als **Technische Informatik** bezeichnet wird ist alles, was die übrigen drei als Informatik kennen. Diese gehen deshalb in aller Regel von der irrigen Vorstellung aus, das InformatikerInnen Programmierung meinen, wenn sie von **Praktischer Informatik** reden. Tatsächlich haben beide (Praktische Informatik und Programmierung) kaum etwas miteinander zu tun. An dieser Stelle sei deshalb (vorrangig für Informatikstudierende) betont:

Dies ist eine **Einführung ins Programmieren, nicht in die Praktische Informatik**. Es wird zwar immer wieder Hinweise auf die Praktische und

Theoretische Informatik geben, aber vorrangig ist und bleibt dies eine Einführung ins Programmieren.

Die **systemnahe Programmierung** und die Programmierung von **Parallelprozessoren** sowie die Implementierung von **Protokollen** für die Datenübertragung über Netzwerke entfallen ebenfalls. Dennoch werden Sie in diesem Buch zumindest einen Einblick in die Grundlagen der systemnahen Programmierung erhalten, da diese Systeme die Grundlage für alle Programmieransätze darstellen, die Sie hier kennen lernen können. Hier gilt dasselbe, was schon im letzten Absatz galt: INT-Akademiker kennen in aller Regel nur die systemnahe Programmierung und alle Konzepte, die sich direkt daraus ableiten lassen und die von InformatikerInnen mit dem Oberbegriff **Technische Informatik** bezeichnet werden. Es gibt jedoch auch Programmierkonzepte, die damit nicht mehr verständlich sind und für die es nötig ist, sich wesentlich grundlegender und abstrakter mit der Programmierung zu beschäftigen. Dazu kommen wir im zweiten Teil dieses Buches.

Fragen des **Software Engineering** werden zwar angerissen und es gibt Hinweise auf typische Missverständnisse, eine grundlegende Einführung ins Software Engineering kann dieser Band jedoch nicht ersetzen. Wie der Titel dieses Buches klar ausdrückt, geht es hier ums Programmieren. An den entsprechenden Stellen werden Sie aber entsprechende Hinweise auf Bücher und Themengebiete finden, damit Sie ggf. wissen, wonach Sie für weiterführendes Wissen suchen müssen. Ob sie sich nun zunächst in die Programmierung oder ins Software Engineering stürzen wollen, bleibt Ihnen überlassen; beides hat seine Vor- und Nachteile. In der Informatik müssen Sie aber in jedem Fall beides durcharbeiten, um auch nur in Ansätzen gute Software entwickeln zu können.

Der Grund ist simpel: Bei der **Programmierung** geht es darum, Konzepte zur Lösung eines Programms in eine Sprache zu übersetzen, die ein Computer ausführen kann. Beim **Software Engineering** geht es dagegen darum, gute Konzepte zu entwickeln, die in Programmiersprachen übersetzt werden können. Wer nur eines von beidem beherrscht entwickelt häufig Programme, die niemandem nützen oder nützliche Konzepte, die niemand (in Form eines Computerprogramms) nutzen kann. Deshalb gibt es in jedem brauchbaren Kurs zur Programmierung Auszüge Teile, die eigentlich in den Bereich des Software Engineering gehören¹. Umgekehrt enthält jeder brauchbare Kurs zum Software Engineering Teile zur Programmierung².

¹Weshalb es nur wenige Kurse zur Programmierung gibt, die nach Ansicht dieses Autors brauchbar sind.

²Weshalb auch hierfür aus Sicht dieses Autors nur wenig Brauchbares auf dem Markt

Zusätzlich müssen Sie jedoch in jedem Fall noch die Grundlagen der Praktischen Informatik erlernen, um hochwertige Software zu erstellen. Diese können Sie im Bereich der Algorithmik (genauer **Algorithmen und Datenstrukturen**, **Algorithmendesign** sowie **Algorithmik**) erlernen.

Aufgrund der häufigen **Änderungen bei aktueller Software** kann dieses Buch nur beschränkt Unterstützung bei Installations- und Konfigurationsfragen bieten. Hier bleibt zu hoffen, dass die Entwickler der einzelnen Sprache bzw. zusätzlicher Software eine ausreichende Dokumentation auf Ihrer Webpage bereitstellen.

Nochmal in anderen Worten: Ein häufiges Missverständnis besteht darin, dass Programmierung und Informatik bzw. Programmierung und Praktische Informatik miteinander verwechselt werden. Denn **Programmierung** ist lediglich die Umsetzung einer Idee mit Hilfe einer Sprache, die einem Computer befiehlt, **was er tun soll**. Ob sie auch festlegt, **wie er das tun soll** ist eine ganz andere Frage. Einführungen in die Programmierung, die hier nicht deutlich werden sind der Grund für eine Vielzahl von Missverständnissen rund um die Programmierung.

Um überhaupt zu programmieren, müssen Sie also lediglich wissen, wie die Befehle und Befehlsstrukturen einer Sprache aussehen. Im Kern ist das also nichts anderes als das Erlernen einer gesprochenen Sprache. Doch so wie es selbst für das Erlernen nahe verwandter gesprochener Sprachen eben nicht ausreicht, nur die Übersetzung einzelner Wörter zu erlernen, genügt es für die kompetente Beherrschung von Programmiersprachen nicht, sich nur grundsätzlich damit beschäftigt zu haben: So wie Sie eine gesprochene Sprache tatsächlich in Gesprächen benutzen müssen, um sie zu erlernen, müssen Sie eine Programmiersprache benutzen, indem Sie eine Vielzahl an Programmen damit entwickeln.

Wichtig:

Fähige (Medien-)InformatikerInnen sind nicht automatisch guten ProgrammiererInnen. Wenn Sie verstanden haben, was der Unterschied zwischen Informatik und Programmieren ist, dann wird es sie wundern, dass es überhaupt Menschen gibt, die diese Aussage bezweifeln.

Die **Informatik** dagegen setzt sich mit der Frage auseinander, wie und ob eine bestimmte Idee besonders elegant und effizient umgesetzt werden kann. **Ob für die Umsetzung der Idee ein Computer nötig ist, ist zweitrangig**. Aber da Computer die Stärke haben, dass Sie langweilige Aufgaben

ist.

ben mit einer für uns unfassbaren Geschwindigkeit ausführen, sind Sie das Werkzeug Nummer 1 für die Informatik. Zumindest ist nach Ansicht dieses Autors die Durchführung von 4 Milliarden Additionen pro Sekunde unfassbar schnell. Zum Vergleich: Würden alle Menschen auf dieser Welt gleichzeitig eine Addition zweier Zahlen mit bis zu 20 Stellen durchführen und für die Berechnung sowie das Aufschreiben nur zwei Sekunden brauchen, dann wären sie alle gemeinsam genauso schnell wie ein einzelner Prozessor, der in einem handelsüblichen Computer steckt.

Hier ein Beispiel, mit dem sich Informatikstudierende gegen Ende des Bachelorstudiums auseinander setzen: Sie fahren in den Skiurlaub und wollen mal das Skifahren ausprobieren. Nun könnten Sie die Skier kaufen oder mieten. Da Sie ja nicht wissen, ob Ihnen Skifahren wirklich Spaß macht, wäre es unsinnig, gleich am ersten Tag das Geld für den Kauf auszugeben. Aber auch am zweiten Tag wäre es nicht unbedingt sinnvoll, denn wer weiß, ob Sie am dritten Tag noch Lust dazu haben. Die Frage lautet also: Wann macht es Sinn, die Skier zu kaufen? Das ist ein Beispiel für einen Bereich, der in der Informatik als **online-Algorithmen** bezeichnet wird. Der zugehörige Bereich der Praktischen Informatik heißt **Algorithmendesign**.

Online-Algorithmen dienen beispielsweise dazu, die Verwaltung des Speichers einer Festplatte zu organisieren oder um die Mitarbeiter für Kassen in einem Supermarkt einzuplanen. Bei online-Algorithmen geht es immer um die Frage: Wie bereite ich mich am besten auf eine Situation vor, von der ich noch nicht genau weiß, wie sie aussehen wird? Und wie Sie sehen hat das zunächst einmal nichts mit Programmieren zu tun.

Sie möchten ein Beispiel, bei dem es um Programmierung und online-Algorithmen geht? Dann haben Sie leider noch nicht verstanden, was der Unterschied zwischen Praktischer Informatik und Programmieren ist. Also weiter mit den online-Algorithmen: Denken Sie beispielsweise daran, was bei ebay kurz vor Ende einer Versteigerung passiert. Oder wie wäre es mit einem online-Händler wie amazon, kurz vor Weihnachten: Wann wie viele Menschen versuchen werden, auf eine einzelne Seite zuzugreifen, kann niemand wissen. Aber mit fähigen Informatikern kann jeder sich darauf so gut wie möglich vorbereiten. Hier sind wir übrigens auch schon ganz nahe an einem aktuellen Forschungsgebiet der (Medien-)Informatik: Bei **Big Data** handelt es sich um einen Bereich, in dem aus gigantischen Datenmengen versucht wird, Gruppen von Personen mit gemeinsamen Eigenschaften herauszufiltern und dann Prognosen über deren zukünftiges Verhalten zu schließen. Hier sind wir ganz nahe am Gebiet des **Datenschutzes** mit ganz konkreten Auswirkungen auf das alltägliche Leben. Denn Big Data führt zum Teil zu absurden Abläufen: Wenn Sie beim Ausfüllen eines Kreditantrages online mehrfach Einträge ändern, dann wird alleine deshalb

der Zinssatz erhöht oder der Antrag abgelehnt. Die Begründung stammt direkt aus dem Bereich angewandten Big Data und lautet so: Menschen mit niedrigem Bildungsniveau vertippen sich häufiger als Menschen mit hohem Bildungsniveau. Menschen mit hohem Bildungsniveau haben in aller Regel ein höheres Einkommen, längere Arbeitsverhältnisse und eine geringere Wahrscheinlichkeit, arbeitslos zu werden. Das lässt sich verkürzen zu: Menschen mit höherem Bildungsstand sind in aller Regel zuverlässiger bei der Rückzahlung von Krediten. Wenn wir jetzt noch die erste Aussage hinzunehmen, lautet die Schlussfolgerung nach den Prinzipien des Big Data: Wer sich öfter vertippt wird häufiger Probleme haben, einen Kredit zurückzuzahlen. Also wird der Zinssatz erhöht oder der Kredit gleich ganz verweigert. Und nein, das ist kein Scherz, sondern findet so Anwendung bei online-Finanzdienstleistern.

Wieder zurück zu den online-Algorithmen: In der BWL wird dieser Teil der Informatik als **Logistik** bezeichnet, wobei der Bereich der online-Algorithmen deutlich mehr umfasst als nur Logistik. BWLern ist dabei in aller Regel leider nicht bewusst, dass es sich hier um einen Bereich handelt, der eine Kernkompetenz der Informatik ist. Das führt dazu, dass in der Forschung zur BWL teilweise Forschungen betrieben werden, um Probleme zu lösen, für die die Informatik längst eine Lösung bereit hält. Denken Sie bitte dennoch nicht in Kategorien wie Schuld: Es ist ein grundsätzliches Problem, dass zu viele Akademiker nur in den Kategorien der eigenen Disziplin denken und kaum den Austausch mit anderen Disziplinen suchen. Das ändert sich zwar in einigen wenigen Fällen, doch häufig kommt es dabei nicht zu einem vollwertigen Austausch, sondern es wird versucht, die jeweils andere Disziplin der eigenen anzupassen. Ein erster Schritt, um das nicht zu tun besteht darin, sich darüber auszutauschen, was Begriffe der jeweiligen Disziplin bedeuten. Ein „gutes“ Beispiel haben bereits kennen gelernt: Was INT-Akademiker als Informatik bezeichnen ist für Informatiker in aller Regel nur der Teilbereich der **Technischen Informatik**. So lange solche Missverständnisse nicht ausgeräumt sind und AkademikerInnen unterschiedlicher Disziplinen die Kompetenz des/der jeweils anderen nicht anerkennen, ist ein echter Austausch nicht möglich. Und dann sind auch umfassende Projekte mit Beteiligung unterschiedlicher wissenschaftlicher Disziplinen nicht möglich.

Sie studieren **Medientechnik** und fragen sich, was das mit Ihrem Studium zu tun hat? Ganz einfach: Auch in Ihrem Bereich werden Computer eingesetzt und Sie werden nicht immer eine fertige Lösung in Form eines Computerprogramms vorfinden. Also müssen Sie im Stande sein, einfache Probleme mit einem Computer selbst zu lösen. Und wenn die Probleme zu komplex werden, dann müssen Sie im Stande sein, InformatikerInnen zu erklären, worin das Problem besteht, denn sonst können die keine Lösung

für Sie entwickeln. Wie wäre es beispielsweise mit einem Programm, mit dem Sie Ihre Schaltskizzen für das E-Technik-Labor erstellen können, die Sie außerdem online speichern und abgeben können, ohne sie als Mailanhang verschicken zu müssen? Wenn Sie die Veranstaltung „Programmieren 1“ (entspricht Teil I dieses Buches) erfolgreich absolvieren, dann werden Sie im Stande sein, solche Programme selbst zu entwickeln.

Sie studieren **Media Systems** und fragen sich, warum Sie einen Kurs mit dem Titel „Einführung ins Programmieren“ (kurz PRG) belegen sollen, wenn Sie doch schon die Veranstaltung „Programmieren 1“ (kurz P1) belegen? In der Veranstaltung P1 lernen Sie die Entwicklung von Programmen mit einer imperativen und klassenbasierten objektorientierten Programmiersprache. In PRG konzentrieren wir uns dagegen auf einen anderen Bereich: Die Entwicklung verteilter Anwendungen. Wann immer Sie eine App nutzen, nutzen Sie eine verteilte Anwendung. Wann immer Sie ein Programm nutzen, das das Internet oder ein anderes Netzwerk voraussetzt, nutzen Sie eine verteilte Anwendung. Normalerweise ist das Stoff für ein Masterstudium, aber ich habe diese Veranstaltung so konzipiert, dass sie für Einsteiger ohne Vorkenntnisse geeignet ist. Denn so bekommen Sie einen Eindruck, wie all die verschiedenen Veranstaltungen Ihres Studiums ein sinnvolles Ganzes ergeben.

Hier nochmals meine Bitte: Wenn Sie in diesem Skript Fehler finden (was bei einem Dokument dieser Länge unausweichlich der Fall ist), Sie weitergehende Fragen haben oder Ergänzungsvorschläge, dann senden Sie diese bitte an mich: markus.alpers@haw-hamburg.de

Work in Progress

Der Begriff Work in Progress bedeutet, dass eine Arbeit noch nicht abgeschlossen ist und somit in Teilen unvollständig und in anderen Teilen unnötig detailliert ist. Das gilt zurzeit für dieses Buch: Zum einen habe ich verschiedene Passagen noch nicht abgeschlossen, zum anderen habe ich verschiedene Texte, die ich ursprünglich für unterschiedliche Kurse erstellt hatte hier zusammengefasst. Da diese Zusammenführung noch nicht abgeschlossen ist, wird es Passagen geben, in denen Sie nahezu identische Aussagen und Erklärungen erneut finden werden. Auch wenn Ihnen solche Passagen auffallen, schicken Sie mir bitte eine E-Mail.

Teil I

Einfache Einführung in die imperative Programmierung

Kapitel 1

Typische Irrtümer darüber, was Programmieren ist.

Studierende im Studiengang **Media Systems** besuchen unter anderem die Veranstaltungen Programmieren 1 und 2 sowie Informatik 3. Ziel dieser Veranstaltungen ist, dass Sie die Grundlagen zweier Arten der Programmierung erlernen. Diese werden als imperative bzw. prozedurale und klassenbasierter objektorientierte Programmierung bezeichnet.

Studierende der **Medientechnik** besuchen ebenfalls zwei Veranstaltungen mit dem Namen Programmieren 1 und 2. Die Inhalte entsprechen einer einfachen Zusammenfassung dessen, was Studierende in Media Systems in den Veranstaltungen „Einführung ins Programmieren“, „Software Engineering“ und „Relationale Datenbanken“ erlernen. Sie bekommen so einen kurzen Einblick in die Bereiche, mit denen sie immer wieder zu tun haben werden, die aber eigentlich Kernbereiche der Informatik sind. Der Grund dafür ist recht simpel: Sobald elektrotechnische Systeme (also der Kernbereich der Medientechnik) zu komplex werden, um sie ohne zusätzliche Strukturierung zu nutzen, kommen wir in einen von zwei Bereichen: Nachrichtentechnik und Informatik. Beide können ohne Verständnis der Elektrotechnik nur zum Teil verstanden werden, aber das gleiche gilt auch umgekehrt.

Aber bevor wir uns ansehen, was diese beiden Arten der Programmierung ausmacht, wo Schnittpunkte und wo Unterschiede vorliegen, sollten wir eine Frage klären: Was verstehen wir eigentlich unter dem Begriff „Programmieren“? Gerade diejenigen, die schon programmiert haben, sollten diesen Abschnitt lesen, denn Sie werden denken, dass Ihnen dieser Begriff klar ist. Einzig diejenigen, die bereits imperativ und (!) deklarativ programmiert haben, werden wissen, worin der Unterschied liegt und können ihn überspringen. (Verwechseln Sie aber bitte nicht die Deklaration einer Va-

riablen mit der deklarativen Programmierung. Beide haben soviel miteinander gemein wie Schweinezucht mit Flugzeugbau.)

1.1 Das ist an diesem Buch anders

Es gibt eine Vielzahl an Einführungen ins Programmieren. Die meisten davon gehören in eine von zwei Kategorien:

- **Variante a** richtet sich an Studierende an Universitäten und ignoriert weitgehend die konkrete Programmierung in einer Sprache. Der Fokus liegt hier vorrangig auf Aspekten der **Algorithmik**. Diese sind zwar außerordentlich wichtig, um fähigeR InformatikerIn zu werden, aber ohne eine Einführung in die konkrete Programmierung in einzelnen Sprachen ist sie kaum verständlich. Daran scheitern dann auch viele Studienanfänger. Und von denen, die nicht daran scheitern versteht nur ein Bruchteil, was Algorithmik ist. Am Ende gibts dann haufenweise Informatikabsolventen von Universitäten, die zwar ganz passabel programmieren können, deren Programme aber letztlich sehr schlecht strukturiert sind.
- **Variante b** behandelt dagegen nur die konkrete Programmierung in einer Sprache und in einer bestimmten Version, ohne dabei auf die allgemeinen Grundlagen einzugehen. Wer eine solche Einführung nutzt hat in aller Regel ein derart mangelhaftes Verständnis der grundlegenden Prinzipien, auf deren Basis die jeweilige Sprache entwickelt wurde, dass er/sie selbst mit großem Aufwand nicht im Stande ist, eine weitere Programmiersprache so zu erlernen, dass er/sie diese wirklich nutzen könnte. Ständig heißt es dann „warum macht der das denn nicht,“ und es wird über die vermeintlich schlechte andere Sprache geflucht. Dabei ist das Problem nicht die „andere“ Sprache, sondern die Tatsache, dass jemand mit dem Verständnis einer Programmiersprache versucht, eine andere Programmiersprache zu erlernen. Doch wenn diese andere Sprache alles genauso machen würde, wie die erste, dann wäre es komplett unsinnig, sie zu erlernen. (Medien-)informatikerInnen lernen deshalb vorrangig die Konzepte kennen, die in verschiedenen Programmiersprachen jeweils unterschiedlich eingesetzt werden.

Beide Ansätze ignorieren darüber hinaus, dass für viele Menschen sich mit der Programmierung beschäftigen wollen, das Innenleben von Rechnern unbekanntes Gebiet sind. Diese Einführung holt Leser dagegen an dem

Punkt ab, an dem keine Vorkenntnisse nötig sind und führt sie kontinuierlich in das Themengebiet ein. Der erste Teil ist dabei so aufgebaut, dass ein Überblick über einige Möglichkeiten der Programmierung vermittelt werden. Erst wenn das geschafft ist, wenn also Leser ein Grundverständnis von verschiedenen Arten der Programmierung haben, beginnt mit dem zweiten Teil die eigentliche Einführung in die Grundlagen der Programmierung. Aber auch wenn Sie schon programmieren können (egal ob in HTML, Java, C oder welcher Sprache auch immer), sollten Sie Teil I des Buches durcharbeiten, weil hier bereits einige Konzepte eingeführt werden und anhand von Beispielen in einer oder mehreren Sprachen verdeutlicht werden.

1.2 Zentrale Begriffe und Konzepte beim Programmieren

Häufig werden die Begriffe Programmieren und Informatik in einen Topf geworfen, dabei haben Sie nicht wirklich viel gemeinsam. Damit Sie also wissen, was Ihnen dieses Buch im Rahmen eines Informatikstudiums bietet und was nicht, schauen wir uns einmal an, was Programmieren eigentlich ist und was Sie von Anfang an beachten sollten.

1.2.1 Der Begriff des Programmierens

Wenn Sie beispielsweise einen HDD-Rekorder programmieren, dann reden Sie zwar vom Programmieren, gehen aber sicher nicht davon aus, dass Sie sich in einem Informatikstudium mit der Frage auseinander setzen, wie Sie einen solchen Rekorder programmieren können.

Interessanterweise können Sie diese Frage aber nach dem Besuch der Veranstaltung „Informatik 3“ beantworten: Dort geht es um die Programmierung von Mikroprozessoren, also just der kleinen schwarzen Boxen, die seit Mitte der 80er Jahre praktisch jedes elektrische Gerät steuern. Na gut, die meisten Toaster noch nicht... Spätestens mit dem **IoT**, dem **Internet of Things** wird das aber kommen.

Aber was machen wir dann in Media Systems in Programmieren 1 und 2? Außerdem fehlt immer noch die Antwort auf die Frage, was Programmieren denn eigentlich ist. Von der Antwort auf die Frage, was das dann wiederum mit Informatik oder gar Medieninformatik zu tun hat, mal ganz zu schweigen.

Wenn wir (wie üblich) zunächst per deutscher Wikipedia suchen, dann erhielten wir am 27. April 2015 die Auskunft, dass es um das Erstellen von

Computerprogrammen geht, was dabei wichtig ist und wer schon etwas darüber geschrieben hat. Aber die eigentliche Antwort auf die Frage, was wir tun, wenn wir programmieren, steht nicht dort.

Dabei ist das recht simpel: Wenn wir programmieren, dann teilen wir einem Computer schlicht mit, **dass er eine Reihe von Aufgaben erfüllen soll**. Und ja, damit ist auch das Drücken der Ruftaste auf einem Telefon eine Programmierung. Nochmal: Es geht darum, dass wir dem Rechner mitteilen, dass er etwas tun soll. Die Frage in welcher Form wir das tun ist davon vollkommen unabhängig und wird unter dem Oberbegriff des **Paradigmas** geklärt.

Kontrolle

Sie sollten jetzt verstanden haben, dass wir den Begriff des Programmierens deutlich allgemeiner verwenden, als das üblicherweise von Programmierern getan wird. Wenn Sie denken, dass Programmieren beinhaltet, wie der Rechner Aufgaben ausführen soll, dann haben Sie eine zu beschränkte Vorstellung des Begriffs Programmieren.

1.2.2 Programmierparadigmen – Wie Programme entwickelt werden können

Sie wissen es jetzt bereits: Einen Computer zu programmieren bedeutet nicht, dass Sie ihm Schritt für Schritt erklärt, wie er eine Aufgabe lösen soll. Denn wenn wir über diese spezielle Art der Programmierung reden, dann nennen wir das **imperative Programmierung**: Hier erstellen Sie wie bei einem Kochrezept Zeile für Zeile eine Liste von Anweisungen, die beschreiben, wie der Rechner eine Aufgabe in Form einzelner Schritte lösen soll. Da das Programm aber nur aus den einzelnen Schritten besteht, ist später nicht mehr erkennbar, welche Aufgabe das Programm lösen soll.

Probleme tauchen hier immer dann auf, wenn ProgrammiererInnen ein Programm erstellt haben, das zu einem Ergebnis kommt, dieses Ergebnis aber nicht die gewünschte Aufgabe löst. Das liegt zum Teil an so subtilen und doch nicht trivialen Aspekten wie der Division einer Zahl durch eine andere Zahl mittels eines Computers.

Im Gegensatz zu dem, was Sie aus dem Deutschunterricht in der Schule als „den Imperativ“ kennen bedeutet imperative Programmierung also nicht nur, dem Computer Befehle zu erteilen, sondern auch ihm zu befehlen, **wie** er einen Befehl auszuführen hat.

Kommen wir damit zur obersten und unumstößlichen Regel bei der Programmierung: **Nicht der Computer oder die Nutzer sind schuld, wenn et-**

was schief läuft, sondern ausschließlich die Entwickler. Wenn Entwickler beispielsweise den Eindruck bei Käufern erzeugen, dass die Nutzung ganz simpel ist, dann ist das nicht die Schuld von Menschen, die sich auf diese Aussage verlassen. Na gut: Wenn Kunden mit Kommentaren kommen wie „Das will ich nicht wissen,“ dann sind sie selbst schuld, aber auch nur dann... also leider fast immer...

Wenn Sie schon einmal programmiert haben, dann werden Sie jetzt wahrscheinlich einwenden, dass man doch nur imperativ programmieren kann. Und damit liegen Sie so falsch wie jemand, der denkt, dass **Programmieren** und **Informatik** praktisch dasselbe wären, oder dass Programmieren und **Praktische Informatik** dasselbe wären. Wie im Vorwort geschrieben haben beide kaum etwas gemeinsam, sondern das eine (Informatik) kann unter anderem dazu genutzt werden, um das andere (Programmieren) gut zu machen.

Mit der Informatik und dem Programmieren ist beispielsweise so, wie mit dem Energiesparen und dem Bau eines Hauses: Es ist möglich, ein energiesparendes Haus zu bauen, aber der Hausbau an sich hat mit Energieersparnis nichts zu tun. Und umgekehrt können Sie in wesentlich mehr Bereichen Energie sparen als nur beim Hausbau: Das eine (Energiesparen) hat etwas mit der Herangehensweise an eine Vielzahl von Bereichen zu tun, das andere (Hausbau) ist eine im Vergleich dazu nur in wenigen Bereichen einsetzbare Tätigkeit, für die Sie das erste aber sehr sinnvoll einsetzen können.

Wenn wir von imperativer Programmierung sprechen, dann fassen wir damit eine Reihe an Programmierparadigmen zusammen. Wenn Sie imperativ programmieren, dann wird das in bestimmten Fällen als **prozedurale Programmierung** oder auch als **strukturierte Programmierung** bezeichnet. Die prozedurale Programmierung ist eine Methode, die dazu gedacht ist, um schlecht lesbaren Programmcode zu vermeiden. Meist ist mit imperativer Programmierung der Spezialfall der prozeduralen und der strukturierten Programmierung gemeint.

Bei der prozeduralen Programmierung zerlegen wir eine Aufgabe so lange in immer genauer definierte Anweisungen, bis wir eine Abfolge von Zeilen haben, die direkt einer Programmzeile einer Programmiersprache entspricht.

Bei der strukturierten Programmierung müssen wir außerdem bestimmte Vorgaben beachten, die verhindern, dass unser Programm unübersichtlich wird. So sind hier Sprünge innerhalb des Programms nur dann erlaubt, wenn wir dadurch einen anderen Programmteil überspringen. Ein Rücksprung an eine beliebige frühere Stelle ist verboten. Es gibt zwar noch

die sogenannten Schleifen und Rekursionen, doch die erlauben keinen beliebigen Sprung zurück zu irgen einem früheren Teil des Programms, sondern stellen nur die Möglichkeit zur Verfügung, Teile des Programms zu wiederholen, bevor das Programm weite Zeile für Zeile abgearbeitet wird. Dabei können wir allerdings durchaus Programmteile entwickeln, bei denen unter bestimmten Bedingungen andere Teile des Programms übersprungen werden.

Leider wird häufig von der **objektorientierten Programmierung** gesprochen (auch diesem Autor passiert das immer wieder), dabei gibt es streng genommen keine objektorientierte Programmierung. Objektorientierung ist strenggenommen ein Konzept des Software Engineering, das zwar in einigen Programmiersprachen direkt angewendet werden kann, aber in aller Regel handelt es sich bei dem, was „objektorientierte“ Sprachen anbieten nur um ein Konzept, das eher eine aktualisierte Form der strukturierten Programmierung ist: Umfangreiche Programmteile werden hier in sogenannte Module (bei Java beispielsweise als Klassen und Package bezeichnet) „verpackt“. Dadurch werden umfangreiche Programme übersichtlicher. ■

Objektorientierung an sich geht einerseits weit darüber hinaus und hat andererseits mit der Lösung einer Aufgabe durch einen Computer eigentlich kaum etwas zu tun. Sie ist im Grunde ein Gegenentwurf zur Grundlage der imperativen Programmierung: Da bei dieser binäre Prozessoren mit Datenübertragungsleitungen und Speicher der konzeptionelle Ausgangspunkt sind, hat sie streckenweise starke Restriktionen, was die Umsetzung von Problemlösungen anbelangt. Die Objektorientierung ist also ein Entwurf, der die Softwareentwicklung von den strikten Restriktionen der imperativen Programmierung unabhängig macht. Sprachen, die tatsächlich die Objektorientierung integrieren sind deshalb für Entwickler mit Erfahrung in imperativer Programmierung kaum zu verstehen. Wenn wir uns der prototypbasierten Programmierung in JavaScript zuwenden, dann werden Sie verstehen, warum das so ist.

Bevor wir zu einer anderen Art der Programmierung kommen, sollten noch einige Begriffe eingeführt werden: Ein **Algorithmus** ist eine Beschreibung dafür, **wie** ein Problem gelöst werden soll. Es ist allerdings noch kein **Computerprogramm**. Wenn Sie basierend auf Algorithmen programmieren, dann haben wir es immer (!) mit **imperativer Programmierung** zu tun. Wann immer also eine Einführung in die Programmierung mit dem Begriff des Algorithmus beginnt, handelt es sich nicht um eine allgemeine Einführung, sondern um eine Einführung in die imperative Programmierung. Ein Computerprogramm ist in diesen Fällen grundsätzlich die Umsetzung eines oder mehrerer Algorithmen in eine bestimmte Programmiersprache.

Manchmal wird in diesem Zusammenhang auch von **Pseudocode** gesprochen. Das ist ein Algorithmus, der so aufgeschrieben wurde, dass er einem (imperativen) Computerprogramm ähnelt. Deshalb ist die Lösung eines Problems in Pseudo-Code sehr praktisch: Angenommen, Sie suchen im Netz nach einer effizienten Lösung für ein Problem, über das Sie bei Ihrer aktuellen Programmieraufgabe stolpern. Wenn Sie die Aufgabe in einer bestimmten Programmiersprache lösen sollen und im Netz finden Sie Code für eine andere Programmiersprache, dann müssen Sie beide Sprachen beherrschen, um die Lösung in Ihre Sprache zu übersetzen. Ist die Lösung dagegen in Pseudocode gegeben, dann können Sie sie umsetzen, so lange Sie die eigene Programmiersprache grundlegend beherrschen.

Dann wäre da noch der Unterschied zwischen Hardware und Software:

- **Hardware** ist die Gesamtheit aller Computerprogramme und sonstiger Bestandteile von Computern, die als „anfassbare“ Komponenten vorliegen,
- **Software** ist die Gesamtheit aller Computerprogramme, die ausschließlich in Form von Daten in Rechnersystemen unterwegs sind.
- Richtig gelesen: **Hardware ist genauso sehr ein Programm bzw. Bestandteil von Programmen, wie die sogenannte Software.** Häufig werden diese Begriffe dagegen so erklärt, als wenn Hardware kein Programmteil wäre. Und das ist falsch; bis auf Dinge wie das Gehäuse eines Rechners dient praktisch die Gesamtheit der Komponenten aus denen ein Rechner zusammen gesetzt ist dazu, Daten zu speichern und zu verarbeiten. Die Speicherung und Verarbeitung eines Programms ist aber bereits ein Programmablauf. Und damit stellt auch die Hardware eine Sammlung von Programmen dar. Später werden wir uns über Server und Client oder auch über Backend und Frontend unterhalten. Genau wie die Unterteilung in Hardware und Software sind auch diese Unterteilungen für uns als (Medien-) InformatikerInnen vollkommen irrelevant.

Hier sind wir auch direkt beim Zusammenhang zwischen Elektrotechnik, Nachrichtentechnik und Informatik: Alle drei beschäftigen sich zum überwiegenden Teil mit der Nutzung von Stromflüssen, um sinnvolle Aufgaben zu erfüllen. Allerdings übernehmen biologische Moleküle und Reaktionen zwischen diesen einen immer größeren Raum in allen drei Bereichen ein oder schaffen sogar vollständig neue Anwendungs- und Forschungsgebiete. In diesem Buch werden wir uns allerdings fast ausschließlich auf die Bereiche konzentrieren, in denen es um Anwendungen auf Basis fließender

Ströme geht:

- Die **Elektrotechnik** setzt einfache Schaltelemente ein, verbindet diese zu zum Teil hochkomplexen Schaltungen und findet vorrangig in der Peripherie von IT-Systemen Anwendungen.
- Die **Nachrichtentechnik** beschäftigt sich mit der Frage, wie beliebige Daten über verschiedene Medien und Distanzen oder Zeiträume transportiert werden können.
- Die **Informatik** beschäftigt sich dagegen mit der Frage, wie Daten zur Erzeugung von Daten genutzt werden können. Sie setzt die Nachrichtentechnik also zur Datenübertragung von Ort zu Ort und zur Speicherung von Daten ein. Die Elektrotechnik kommt hier insbesondere bei der Interaktion von Informatik-Systemen mit der Umgebung ein.
- Wenn wir Informatik mit Nachrichtentechnik oder Elektrotechnik verbinden, landen wir direkt bei der **Technischen Informatik**.

Aber zurück zum eigentlichen Thema dieses Abschnitts und damit zu einer anderen Art der Programmierung:

Es gibt auch Programmiersprachen, in denen man die **Prämissen** der Aufgabe beschreibt und den Rechner dann auffordert, eine mögliche Lösung zu nennen. Eine Prämisse ist eine Voraussetzung für etwas bzw. bei der logischen Programmierung so etwas wie eine Bedingung, die in irgend einer Form Auswirkung darauf hat, welche möglichen Lösungen für unser Problem existieren. Dieser Ansatz der Programmierung wird als **logische Programmierung** bezeichnet und gehört in den Bereich der **deklarativen Programmierung**. Zur Erklärung:

Und vermutlich ploppen genau jetzt vor Ihrem inneren Auge die Fragezeichen auf: Wie soll das denn gehen?! Da wir uns zunächst mit verschiedenen Formen der imperativen Programmierung beschäftigen werden, sei hier nur ein Beispiel angeführt:

Jemand fragt Sie, ob Sie ihm den Weg zu einem Restaurant beschreiben können. Hier gibt es natürlich mehrere Möglichkeiten zu antworten. Wie sinnvoll eine Antwort ist, das hängt von den Prämissen ab, die für den Fragenden gelten, z.B.: Welches Verkehrsmittel will er nutzen? Welche Teile des Stadtplan kennen Sie? Usw. usf.

Eine Form deklarativer Programmierung besteht nun darin, dass Sie all diese bekannten Fakten (Straßennetz, Bedingungen des Fragenden, usw.)

einprogrammieren. In der logischen Programmiersprache **PROLOG** geschieht das in Form sogenannter **Klauseln**. Diese Klauseln ähneln sehr den Relationen, die Sie in mathematischen Vorlesungen kennen lernen. Abschließend geben Sie eine Klausel ein, die z.B. die Frage repräsentiert, ob es einen Weg zum Ziel (hier dem Restaurant) gibt, ob es einen Weg einer bestimmten Länge dorthin gibt usw. Im Gegensatz zur imperativen Programmierung müssen Sie dem Computer dagegen nicht einprogrammieren, wie er nach diesem Weg suchen soll. Das erfolgt nach Regeln der Aussagenlogik und ist bereits als Teil der Programmiersprache festgelegt. Das Programm gibt dann eine mögliche Lösung an oder es gibt an, dass es keine solche Lösung gibt. Wenn Sie sich also bislang mit der Planung des Einsatzes von Mitarbeitern herumgeschlagen haben, dann erlernen Sie doch stattdessen die Programmierung in PROLOG, dann können Sie dieses Problem durch einen Computer lösen lassen.

Solche unterschiedlichen Ansätze der Programmierung werden auch als Paradigmen bzw. Programmierparadigmen bezeichnet. Langfristig werden Sie eine Vielzahl weiterer Paradigmen kennen lernen. Wenn Sie später einen Master in Informatik machen wollen, müssen Sie sich damit bereits während des Bachelorstudiums beschäftigen.

Kontrolle

Was Sie sich an dieser Stelle merken sollten, sind zunächst zwei Punkte:

- dass es verschiedene Paradigmen gibt,
- dass Sie Programmiersprachen danach auswählen sollten, ob sie für das Paradigma passend sind, mit dem Sie gerade zu tun haben.

Wenn Sie das verstanden haben, dann haben Sie auch verstanden, warum Diskussionen sinnlos sind, in denen es darum geht, dass gewisse Sprachen nichts taugen. Für Betriebssysteme gilt ähnliches. Es ist allerdings durchaus möglich, dass frühere Versionen von Sprachen (genau wie Betriebssystemen) schlichtweg überholt sind und damit tatsächlich nicht mehr sinnvoll einsetzbar sind. Vor allem zeichnet es (Medien-)InformatikerInnen aus, dass sie im Stande sind, ein Paradigma auszuwählen, mit dem ein bestimmtes Problem effizient gelöst werden kann.

Das zweite, was Sie jetzt verstanden haben sollten ist, dass das was die meisten Menschen allgemein unter Programmierung verstehen die sogenannte prozedurale Programmierung ist, die zur Obergruppe der imperativen Programmierung gehört.

1.2.3 Man muss doch nicht immer das Rad neu erfinden – Middleware, Framework und Bibliothek

Nahe verwandt mit Programmiersprachen sind die Begriffe **Middlewa-re**, **Framework** und **Bibliothek**. In der Urzeit der Programmierung entwickelte jeder Programmierer alles selbst. Dann wurden Softwarepakete entwickelt, die wie der Teil einer Programmiersprache verwendet werden können, aber im Grunde vollständige oder fast vollständige Programme sind. Je nachdem, wie umfangreich diese Pakete sind und was sie an funktionalem Umfang bieten, unterscheidet man zwischen den drei genannten Arten.

Der Begriff Middleware hat eine besondere Bedeutung, die Sie dann kennen lernen werden, wenn Sie eine Veranstaltung zur Nachrichten- oder Kommunikationstechnik besuchen. Dort steht er in aller Regel weniger für etwas, das direkt mit Programmierung zu tun hätte, sondern vielmehr für bestimmte Teile von Strukturen, Aufgaben und Hierarchien innerhalb eines Netzwerkes.

Im Rahmen dieses Buches werden wir zwischen den dreien nicht unterscheiden, die Begriffe werden hier nur eingeführt, damit Sie wissen, dass es da um Programmteile geht, die Sie in Ihrer Software nutzen können, ohne sie selbst entwickelt zu haben.

Wichtig

Java beinhaltet zwar auch einen Teil, mit dem Sie imperativ programmieren können, aber **zum Großteil ist Java eine Middleware**.

Kontrolle

Es sollte Ihnen bewusst sein, dass Sie sich viel Zeit sparen können, indem Sie auf Middlewares, Frameworks und Bibliotheken zurückgreifen. Wie genau das geht, ist Teil aller Veranstaltungen, in denen Sie programmieren. Aber wie schon eingangs erläutert ist das Programmierung und nicht Praktische Informatik.

1.2.4 Damit Sie sich aufs Entwickeln konzentrieren können – IDEs / Entwicklungsumgebungen

Auch IDEs (Integrated Development Environments bzw. Entwicklungsumgebungen) sind ein Mittel, mit dem Sie sich viel Arbeit sparen können. Eine IDE ist eine Zusammenstellung von Programmen, die Sie beim Programmieren unterstützen. Anfangs werden wir ohne eine IDE arbeiten, da Einsteiger häufig von den vielen Komfortfunktionen eher verwirrt als unterstützt werden.

Kontrolle

Sie sollten den Begriff IDE in Zukunft so selbstverständlich benutzen, wie andere Menschen den Begriff Brot.

1.2.5 Dokumentation

Nachdem Sie jetzt also einen ersten Eindruck davon haben, was Programmieren ist und wie Sie es sich erleichtern können, kommen wir zu einem entscheidenden Unterschied zwischen dem Alltag von professionellen ProgrammiererInnen und von HobbyprogrammiererInnen: Die Arbeit im Team.

Sie wissen wahrscheinlich, dass professionelle Software üblicherweise nicht von einzelnen Entwicklern in der Garage, sondern zum Teil von Hunderten von Mitarbeitern entwickelt wird. Und die müssen irgendwie miteinander arbeiten. Bevor wir hier auf die Details eingehen, folgt die zweite wichtige Regel fürs Programmieren:

Ein einsamer Wolf kam eine gute Software initiieren, aber dauerhaft können nur Teams daraus eine gute Software entwickeln.

Der erste Schritt, um im Team erfolgreich Software zu entwickeln, ist die Dokumentation. Dokumentationen sind gleichzeitig der aufwändigste und vermeintlich wertloseste Teil der Arbeit im Team. Doch langfristig ist eine Softwareentwicklung ohne Dokumentation zum Scheitern verurteilt.

Stellen Sie sich einfach folgende Situation vor: Ein Kollege hat (irgendwann) einen Programmteil entwickelt, die in einem Spezialfall fehlerhaft funktioniert, der bislang nie auftrat. Zum Glück ist Ihnen das aufgefallen. Aber leider kann sich niemand mehr daran erinnern, wo genau sie in den 5000 Zeilen steckt und wer das damals eigentlich programmiert hat. Hätten Sie eine gute Dokumentation, dann würden Sie es jetzt nachschlagen. So können Sie nur beten, dass der Fehler niemals auffällt... Was natürlich bei der Steuerung eines AKWs keine gute Lösung ist.

Kontrolle

Wenn Sie in ein bestehendes Team einsteigen wollen, fragen Sie nach der Dokumentation. Gibt es keine oder ist sie sehr dünn, dann wird das Projekt scheitern, weil am Ende niemand mehr versteht, welche Funktion wo erfüllt wird. Allerdings werden Dokumentationen in aller Regel nicht ausgedruckt.

1.2.6 SCM / Versionskontrolle

Damit Sie nun mit anderen gemeinsam an einer Software arbeiten können, gibt es Programme, die als Versionskontrollsysteme bezeichnet werden. Daneben ist auch die Bezeichnung **Software Control Management (SCM)** üblich. Bei diesen wird ihr Programm in einem sogenannten **Repository** gespeichert, das auf einem Server platziert wird, den Sie über ein Netzwerk erreichen können.

Im Gegensatz zu dem, was sie wahrscheinlich bislang kennen gelernt haben, werden im Repository auch alle Änderungen (**Deltas**) gespeichert, so dass Sie mittels einzelner Befehle unterschiedliche Versionen der Software einsehen und bearbeiten können.

Sie werden aktuell vorrangig auf zwei SCMs treffen: **Git** und **SVN** (kurz für **Subversion**). Ohne auf die Details einzugehen: Bei Git haben Sie den Vorteil, dass Sie auch dann problemlos weiterarbeiten können, wenn Sie keinen Zugriff auf das Repository haben. Das ist bei SVN nur sehr beschränkt möglich.

Übrigens ist Git ein SCM, das von Linus Torvalds, dem Initiator und höchsten Entwickler von **Linux** angestoßen wurde. Mehr dazu auf linux.com und git-scm.com

Kontrolle

Ja, Sie können im stillen Kämmerlein vor sich hin arbeiten, aber das Thema Versionskontrolle müssen Sie im Hinterkopf haben. (Und das Thema Dokumentation natürlich erst recht.)

1.2.7 Software Engineering / Softwareentwicklung

Und wieder folgt ein Begriff, der Ihnen in Fleisch und Blut übergehen muss: **Software Engineering**, was als **Softwareentwicklung** übersetzt wird. Die Idee hier besteht im Gegensatz zu den Paradigmen weniger darin, dass Sie bestimmte Konzepte nutzen, um eine Aufgabenstellung zu lösen, sondern es geht um die Arbeitsteilung bei der Entwicklung eines großen Projekts. Die ursprüngliche Bedeutung des Begriffs lässt sich mit „Sicherstellung hoher Qualität von Softwarezusammenfassungen, doch wenn wir uns damit beschäftigen, sind wir wieder einmal im Bereich der **Praktischen Informatik**.

Wo es bei den Paradigmen um die Frage geht, wie wir eine möglichst optimale Lösung für unser Problem erreichen, stehen beim Software Engineering entsprechende Fragen im Mittelpunkt: Was will der Kunde eigentlich?

Wie lange brauchen wir dafür? Können wir das überhaupt anbieten? Was wollen wir dafür berechnen? Wie oft müssen wir mit dem Kunden Besprechungstermine vereinbaren? Usw. usf.

Streng genommen handelt es sich hier also um eine Kernkompetenz der Betriebswirtschaftslehre (BWL), die als Projektmanagement bezeichnet wird. Aber im Gegensatz zum **Projektmanagement** der BWL haben wir Werkzeuge zur Verfügung, mit denen wir räumlich getrennt gemeinsam an einer Software arbeiten können.

Einer der neuesten Vertreter dieser Spezies wird als **agile** Softwareentwicklung bezeichnet. Ältere Vertreter laufen unter Namen wie **Wasserfallmodell** und **V-Modell**. Aber keine Sorge, die Feinheiten des Software Engineering lernen Sie erst später im Studium kennen.

Kontrolle

Sie sollten wissen, dass es beim Software Engineering um Methoden geht, um Teamarbeit bei Softwareprojekten zu koordinieren. Dadurch wird eine hohe Qualität von Software in großen Projekten sichergestellt. Um hohe Qualität von Software geht es zwar auch bei der Praktischen Informatik, doch was Sie dort lernen können nützt Ihnen individuell bei der Entwicklung hochwertiger Software. Als letztes werfen wir nochmal kurz einen Blick auf die Programmierung: Um qualitativ hochwertige Software zu entwickeln müssen Sie einige Grundlagen verschiedener Programmierparadigmen lernen. Wenn Sie sehr gut in der Praktischen Informatik sind, ist es weitgehend belanglos, wie gut Sie eine bestimmte Programmiersprache beherrschen: Sie werden gute Software entwickeln können. Wenn Sie dagegen kaum etwas von Praktischer Informatik verstehen, dann werden Sie selbst in Sprachen, die Sie sehr gut beherrschen bestenfalls mittelmäßige Software entwickeln.

1.2.8 Für diejenigen, die die Programmierung von Apps erlernen wollen

Der Begriff **App** ist im Grunde absurd. Eine App ist nichts anderes als ein Kunstwort, das geschaffen wurde, um Menschen, die nichts von Informatik oder Programmierung verstehen den Eindruck vorzugaukeln, dass es sich hier um eine besonders moderne oder hochwertige Anwendung handelt. Tatsächlich ist das Gegenteil der Fall: Der Begriff App ist vom Begriff Application abgeleitet, was nichts anderes als **Anwendung** heißt. Eine Anwendung ist ein Programm, das für die Nutzung durch Menschen gedacht ist. Zwar wird in Bezug auf Anwendungen für mobile Endgeräte regelmäßig von Apps gesprochen, aber es gibt nichts und zwar überhaupt rein gar nichts, was an der Entwicklung solcher Anwendungen anders wäre als bei

der Entwicklung von Anwendungen für andere Endgeräte.

Im Gegensatz zu anderen Anwendungen sind Apps jedoch immer systemabhängig; Sie können also keine App entwickeln, die auf iPhone und Android unverändert lauffähig ist: Für jedes System sind umfangreiche Anpassungen nötig, die einzig deshalb nötig sind, weil die Entwickler der Geräte die Softwareentwickler davon abhalten wollen, für mehrere Systeme zu entwickeln. Gerade wenn Sie sich die Entstehungsgeschichte von Java ansehen werden Sie feststellen, dass die Systemabhängigkeit bei der App-Entwicklung mit Java (beispielsweise bei Android) geradezu absurd ist.

Wer die Entwicklung von Apps lernen will zeigt damit also im Regelfall, dass er/sie so viel von Softwareentwicklung versteht wie jemand, der glaubt, dass das Rauchen von Zigaretten, der Konsum von Alkohol bzw. Drogen oder der Kauf von Waren auf Ratenzahlung etwas mit Freiheit zu tun hätte: Er/Sie ist auf einen Werbeslogan hereingefallen und bedankt sich noch dafür, dass hohe Kosten für etwas zu zahlen sind, das das Versprechen nicht einhält und meist noch andere Kosten nach sich zieht.

Leider müssen wir diesen Begriff aber im Regelfall nutzen, weil Konsumenten und nicht-Informatiker auf genau dieses Marketingkonstrukt hereinfließen und gar nicht begreifen, wie unsinnig der Begriff App ist. Da diese Menschen uns für unsere Arbeit bezahlen müssen wir uns hier quasi dumm stellen und die „asiatische Lösung“ wählen: Immer schön nicken und lächeln. Womit wir bei einem der Gründe wären, warum viele IT'ler und NT'ler im Servicebereich regelmäßig schlechte Laune haben, wenn sie mit nicht-IT'lern über IT reden (müssen): Sie werden von Nutzern damit konfrontiert, dass die ach so tollen Apps eben längst nicht das leisten, was die NutzerInnen sich aufgrund der Marketingbotschaften einbilden.

Schauen Sie sich dazu die Geschichte von Apple in den letzten zwanzig Jahren an, denn die ist direkt mit der Einführung des Begriffs der App verbunden: Gegen Ende der 90er stand Apple kurz vor dem Konkurs. Also entwickelten Jobs & Co. mobile Endgeräte, für die im Ein- bis Zweijahresrhythmus Nachfolger mit geringfügigen Verbesserungen bezüglich der Funktionalität erschienen. Vielmehr wurden hier jeweils signifikante ästhetische Änderungen durchgeführt, die Käufern den Eindruck vermittelten, die Nutzung dieser Geräte sei gleichbedeutend damit, zur technologisch-gesellschaftlichen Elite zu gehören. Gleichzeitig wurde wie in der Mode der Eindruck vermittelt, dass nur die jeweils aktuelle Baureihe genügen würde, um diesen vermeintlichen Status beizubehalten. Als dann nach einigen Baureihen absehbar war, dass dieses Vorgehen beim iPod nicht mehr die nötige Kundenbindung erzeugen würde, wurde mit

dem iPhone bzw. dem iPad eine Kombination von iPod, Mac und Handy entwickelt, bei der die gleiche Strategie erfolgreich verfolgt wurde.

Wenn Sie über den Begriff **mobiles Endgerät** irritiert sind: Damit werden vorrangig in der Nachrichtentechnik all die Geräte bezeichnet, die an ein Netzwerk angeschlossen sind, die aber relativ frei bewegt werden können. Es geht hier also um Handys, Smartphones, Laptops mit WLAN, usw. Denn auch wenn die Nutzung eines Smartphones mit „Internet“-zugang für Sie genauso selbstverständlich sein dürfte wie die Nutzung eines Rechners, der per Kabel ans Internet angeschlossen ist, ist die Technik und Technologie, durch die insbesondere kabellose Anschlüsse realisiert werden alles andere als simpel. LTE ist dabei der erste erfolgreiche Versuch, verschiedene Arten von Vernetzung zu kombinieren.

Mittlerweile ist aber auch im Bereich von Smartphones die maximale Ausreizung des Marktes erreicht. Deshalb wurden inzwischen die **Wearables** entwickelt: Kleine Geräte, die nur in Verbindung mit der aktuellsten Baureihe eines bestimmten Smartphones nutzbar sind. Während bislang die Ästhetik der Geräte selbst im Vordergrund stand, um eine vermeintlich elitäre gesellschaftliche Position und modisches Bewusstsein der BesitzerInnen zu vermitteln, steht bei den Wearables das Konzept im Vordergrund, KäuferInnen würden durch das sichtbare Tragen dieser Geräte zusätzlich sportliche Fitness, Flexibilität und Belastbarkeit beweisen. Mittlerweile steht Apple hier allerdings in direkter Konkurrenz zu Samsung. Die Koreaner haben dabei den großen Vorteil, dass Sie aus dem Land kommen, das weltweit seit Jahren die modernste Internetinfrastruktur besitzt. Deutschland ist in dieser Hinsicht selbst im EU-weiten Vergleich bestenfalls Mittelmaß, was ein essentieller Grund dafür sein dürfte, dass Siemens schon vor Jahren aus der Entwicklung von mobilen Endgeräten ausgestiegen ist.

Es sei an dieser Stelle allerdings noch angemerkt, dass Geräte von Apple häufig tatsächlich einen funktionalen Mehrwert gegenüber denen der Konkurrenz besitzen. Nur wird dieser Mehrwert eben von vielen Kunden gar nicht genutzt: Beispielsweise sind die Schnittstellen derart hochwertig, dass die Verbindung mehrerer Geräte im Regelfall problemlos ohne Zutun von Nutzern funktioniert. Microsoft versucht seit einigen Jahren ebenfalls in diesen Marktbereich einzudringen, um zur Konkurrenz im Bereich mobiler Endgeräte aufzuschließen. Das neueste Surface und Nokias neuste Modelle der Lumia-Reihe stellen in Verbindung mit dem Marketingkonzept rund um Windows 10 und dem augmented reality System HoloLens den ersten erfolgreichen Versuch von Microsoft dar, die Grenzen zwischen verschiedenen Systemen aufzubrechen und neue Systemtypen ins Angebot zu integrieren. Wenn Sie sich dann allerdings die Situation des Herstellers von Blackberry Smartphones ansehen oder die Gründe für Apples wirt-

schaftlichen Einbruch in den 90er Jahren, werden Sie feststellen, dass eine Konzentration auf hochwertige Geräte für den Einsatz im professionellen Bereich wahrscheinlich keine Basis für dauerhaften wirtschaftlichen Erfolg darstellt.

Nun fragen Sie sich vielleicht, was all das mit einem Hochschulkurs zur Programmierung zu tun hat. Die Antwort ist simpel: Nach Ihrem Studium (egal ob Sie nach dem Bachelor noch einen Master und ggf. eine Doktorarbeit anfertigen oder direkt ins Berufsleben bzw. die Forschung starten) werden Sie entweder für einen Arbeitgeber tätig werden oder selbständig sein. Und wo auch immer das sein wird, gilt eine grundsätzliche Tatsache: Wenn Sie Software (oder auch Hardware) entwickeln, dann hängt Ihre Zukunft davon ab, ob diese Software tatsächlich deutlich mehr Geld einbringt, als Ihre Entwicklung gekostet hat. Schließlich müssen kontinuierlich neue Hardware angeschafft, Lizenzen erworben, Miete gezahlt werden usw. Das soll kein Appell gegen Indie Games oder Open Source sein, sondern es geht hier um Faktoren, die Ihre Zukunft bestimmen werden. Denn egal wohin Sie im Anschluss gehen, müssen Sie sich bewusst sein, dass Geld nicht einfach so entsteht, wie beispielsweise die Planer von Projekten wie der HafenCity, der Elbphilharmonie, von Stuttgart 21, Flughafen BER usw. usf. denken, sondern dass Sie nur dann dauerhaft ein zumindest ausreichendes Einkommen erreichen, wenn Sie Software entwickeln, die sowohl bestimmte Qualitätsstandards erreicht als auch von einer gewissen Anzahl von Kunden gekauft wird. Sollten Sie also in einem Unternehmen tätig werden, in dem entweder die Qualität der Software oder die Änderungen im Umfeld ignoriert werden, dann sollten Sie sich schnell nach einer neuen Aufgabe umsehen. Diese ignorante Strategie des Managements kann einige Jahre funktionieren, aber früher oder später wird sie das nicht mehr tun. Und ja, das gilt auch für den öffentlichen Dienst wie beispielsweise in öffentlichen Hochschulen, auch wenn wir hier nicht über wenige Jahre sprechen, sondern über Zeiträume, die eher in Richtung von zehn bis zwanzig Jahren gehen.

Zwei konkrete Beispiele möchte ich an dieser Stelle nennen: Zynga galt in den ersten Jahren, in denen Facebook international erfolgreich wurde als das erfolgreichste Unternehmen im Bereich der Browsergames. Werfen Sie dazu einen Blick in die Making Games-Ausgaben von 2012. Beispielsweise stammt das Spiel Farmville von diesem Entwickler. Zynga existiert zwar noch, aber mittlerweile hat das Unternehmen vier Fünftel seines Wertes verloren und viele Mitarbeiter wurden entlassen. Noch schlimmer sieht es beim ursprünglich größten Konkurrenten von Blizzard-Activision im Bereich MMORPGs aus: Sony Online Entertainment (kurz SOE) war u.a. mit Everquest II einer der erfolgreichste Entwickler von MMORPGs bis World of Warcraft erschien. Bei nachfolgenden Titeln versuchte SOE dann stets be-

sonders hohe Qualitätsmaßstäbe zu setzen, was regelmäßig misslang. Im Februar 2015 wurde das Unternehmen an einen Finanzdienstleister verkauft und existiert heute unter dem Namen Daybreak Game Company. In beiden Fällen haben Sie es mit Unternehmen zu tun, die eine Zeitlang sehr erfolgreich waren, die es aber nicht geschafft haben, sich den Änderungen des Marktes anzupassen. Und die Auswirkung bestand darin, dass viele Mitarbeiter arbeitslos wurden.

Umgekehrt gibt es aber auch Unternehmen, die eine ausgewählte Kundengruppe bedienen, damit kontinuierlichen Erfolg haben aber nur gelegentlich in Spieletestzeitschriften auftauchen. Der isländische Entwickler CCP-Games beispielsweise betreibt seit mehr als 10 Jahren das MMORPG Eve online, das seit Jahren kontinuierlich von mehr als 300.000 Kunden monatlich bezahlt wird. Zwar gab es hier durchaus Einbrüche bei den Nutzerzahlen, doch das Unternehmen reagierte auf Kritik und konnte so weiterhin bestehen.

1.3 Informatik versus Programmierung, Studium und Arbeit

Nachdem Sie jetzt einen ersten Einblick in einige Bereiche bekommen haben, die bei der Programmierung eine Rolle spielen, schauen wir uns jetzt an, wo Informatik und Programmierung zusammenhängen.

1.3.1 Informatik und Programmierung

InformatikerInnen oder der Informatik nahe stehende Akademiker sind im Stande, ein Problem unabhängig von einer bestimmten Programmiersprache auszuformulieren und sich für ein Paradigma entscheiden, mit dem sie dann das Problem lösen können.

Der Bereich der **Praktischen Informatik** widmet sich u.a. der Frage, wie Sie das Problem effizient lösen können und hat mit der **Programmierung**, also der Umsetzung dieser Lösung in einer Programmiersprache nichts zu tun. Wenn Sie zusätzlich die Grundlagen der **Theoretischen Informatik** gemeistert haben, können sie außerdem erkennen, ob ein Problem überhaupt lösbar ist. Quereinsteiger versuchen dagegen häufig Programme zu entwickeln, die gar nicht lösbar sind. Und erst wenn sie wissen, dass und mit welchen Mitteln ein Problem idealerweise lösbar ist, entwickeln sie eine Lösung und setzen diese dann in einer gut geeigneten Sprache um. Außerdem sind sie im Stande, mit Dutzenden oder Hunderten anderer InformatikerInnen und ProgrammiererInnen gemeinsam Software zu entwickeln.

Auch hier nochmal der Hinweis: NaturwissenschaftlerInnen, TechnikerInnen und IngenieurInnen kennen diese Unterteilung in aller Regel nicht. Sie erlernen das Programmieren in einer oder mehreren Sprachen, die für Ihren Bereich voll und ganz ausreichen. Dazu lernen Sie meist noch die Grundlagen dessen, was InformatikerInnen als **Technische Informatik** zusammenfassen, gehen aber davon aus, dass es sich hier um Informatik als ganzes handelt. Der Austausch mit diesen INT-AkademikerInnen ist deshalb meist schwierig: Sie begreifen oftmals nicht, dass sie die beiden Bereiche der Praktischen und Theoretischen Informatik mit Ihren Kenntnissen nicht erfassen können und gehen zusätzlich davon aus, dass Praktische Informatik dasselbe wie Programmieren ist. Und das ist nicht einmal ansatzweise richtig.

ProgrammiererInnen können all das nicht oder nur zu einem geringen Teil. Sie beherrschen eine oder mehrere Programmiersprachen und quetschen eine Lösung in diese Sprachen, egal ob das Ergebnis überhaupt brauchbar ist oder nicht. Manches, was Media Systems Studierende schon im Studium kennen lernen erkennen reine Programmierer erst nach Jahren oder Jahrzehnten. Aber unterschätzen Sie sie nicht: Dafür kennen ProgrammiererInnen häufig Kniffe, die man eben erst dann kennen lernt, wenn man eine Programmiersprache in- und auswendig gelernt hat. Und das ist immer wieder sehr wertvoll.

Leider werden InformatikerInnen und ProgrammiererInnen auch häufig deshalb in einen Topf geworfen, weil es den Ausbildungsberuf zum/zur „FachinformatikerIn“ gibt. Dort liegt der Fokus im Gegensatz zum Informatikstudium klar auf der Programmierung. Im Anglo-Amerikanischen Raum gibt es noch den Studiengang Computer Science, der einem FH-Studium der Informatik hierzulande ähnelt.

Sie können sich den Unterschied zwischen Informatik und Programmierung auch dadurch veranschaulichen, dass Sie an Bauarbeiter und Bauingenieure denken: Ein Bauingenieur weiß, wie ein von ihm geplantes Gebäude errichtet werden muss, damit es z.B. nicht unter dem eigenen Gewicht zusammen bricht. Aber er wird im Regelfall viele Arbeitsschritte nicht selbst durchführen können. Der Bauarbeiter dagegen wird sehr genau wissen, wie die Arbeitsschritte richtig auszuführen sind. Dafür kennt er sich beispielsweise nicht mit den Kräften aus, die bei einem modernen Hochhaus auftreten. Sie beide haben unterschiedliche Fähigkeiten, aber nur gemeinsam können sie großes erreichen.

1.3.2 Informatik: Uni versus HAW (FH)

Im Gegensatz zu Ihnen beschäftigen sich Universitätsstudierende in der Informatik praktisch durchgehend mit der Frage, wie eine bestimmte Aussage zu beweisen ist. Diese Fragestellung ist spannend, aber es geht hier um rein abstrakte Konzepte, die bei höchst komplexen Projekten massive Effizienzsteigerung bedeutet.

Deshalb sollten Sie sich bewusst machen: Gute UniversitätsabsolventInnen sind Ihnen im Regelfall deutlich überlegen, wenn Sie es bei einer Aufgabenstellung mit sehr großen Datenmengen zu tun haben.

1.3.3 Informatik und Programmieren im Beruf

Schon beim Studienbeginn fragen viele Studierende, wie denn Ihre Aussicht auf dem Arbeitsmarkt ist. Hier gibt es zwei grundsätzliche Varianten: Zum einen können Sie nach dem Bachelor-Abschluss ein Masterstudium anstreben. Das ist gerade in der Elektrotechnik und der Informatik kein allzu großes Problem, weil hier viele Studienabsolventen bereits mit einem Bachelor-Abschluss spannende Stellen bekommen. Die Bewerbungssituation ist also entspannter, als beispielsweise bei den Sozialwissenschaften. Wenn Sie diesen Weg verfolgen, können Sie langfristig (ein Dokortitel ist da leider immer noch eine übliche Voraussetzung, wenn es Ihnen nicht genügt, als Assistent zu arbeiten) eine akademische Forschungskarriere anstreben.

Wenn Sie dagegen das Studium aufgenommen haben, um anschließend direkt in den Arbeitsmarkt zu starten, dann sollten Sie die Zeit neben dem Studium nutzen, um eigene Projekte zu realisieren. Denn das ist bei vielen Arbeitgebern eine gute Möglichkeit, um die eigenen Fähigkeiten nachzuweisen.

Doch das beantwortet natürlich nicht die Frage, welche Kenntnisse Arbeitgeber eigentlich erwarten, bzw. was Sie (neben dem Studium) an Kenntnissen erwerben müssen, um für eine bestimmte Tätigkeit gut vorbereitet zu sein. Dieser Abschnitt soll Ihnen da eine kleine Unterstützung bieten. Natürlich treffen die folgenden Aussagen nicht auf alle Arbeitgeber zu und wie die Situation in drei Jahren aussieht (also zu dem Zeitpunkt, zu dem Sie das Ende Ihres Studiums anstreben), ist noch eine andere Frage. Als Grundlage für die folgenden Aussagen dienten zwei Quellen: Zum einen aktuelle Stellenausschreibungen (August 2015), zum anderen Gespräche mit HR'lern im Hamburger Raum.

Die einfachste Antwort darauf, was von Ihnen häufig erwartet wird ist die

am wenigsten hilfreichste: Es wird von Ihnen erwartet, dass Sie **backend** developer oder **frontend** developer sind oder sonst eine Stelle ausfüllen können, die in Unternehmen zu besetzen ist. Das hat aber mit einem Studium (z.B. der Informatik) nichts zu tun, denn was Sie hier lernen (können) sind grundlegende Techniken und Methoden, die in beiden Bereichen angewendet werden. Leider gibt es immer wieder HR'ler (ja selbst in IT-Unternehmen), denen das nicht klar ist und die Sie dann mit Fragen konfrontieren, die Sie kaum beantworten können. Machen Sie sich in diesen Fällen nichts draus, wenn's mit der Stelle nichts wird; da gibt es bessere Angebote.

Die nächste Antwort hilft auch nicht weiter: Selbstverständlich sollen Sie... naja, so ziemlich alles können. Ums kurz zu machen, auch von solchen Unternehmen sollten Sie Abstand halten, denn dort wird von Ihnen im Grunde erwartet, dass Sie die Kenntnisse mitbringen, die Ihrem Vorgesetzten fehlen. Und glauben Sie mir, das wollen Sie nicht.

Kommen wir jetzt also zu hilfreichen Antworten.

Zunächst sollten Sie, nachdem Sie alle (!) Leistungsnachweise der ersten drei Semester erworben haben (ja, damit sind insbesondere Mathe 1 und 2 gemeint), sich ein wenig Zeit nehmen und prüfen, welche Bereiche Ihnen besonders liegen, bzw. mit welchen Konzepten der verschiedenen Veranstaltungen Sie am meisten anfangen konnten.

Hier ein Einwurf: Viele Studierende, die in Mathe 1 (oder einem ähnlich anspruchsvollen Fach) durchfallen, versuchen im nächsten Semester dennoch alle Veranstaltungen zu bestehen. Manche „schiebenäuch das, was noch nachzuholen ist in ein späteres Semester, weil Sie denken, dass das der passende Ansatz wäre. Beides hat wenig Aussicht auf Erfolg: Erledigen Sie zuerst das, was zuerst im Studienplan vorgesehen ist. Und wenn Sie dann zwei Veranstaltungen haben, zwischen denen Sie sich entscheiden müssen, dann entscheiden Sie sich für diejenige, die Sie für schwerer halten. Wenn Sie dann merken, dass Sie nirgends richtig vorankommen, dann streichen Sie die jeweils leichteste Veranstaltung, damit Sie für die übrigen Veranstaltungen mehr Zeit haben. Denn wenn Sie so agieren, haben Sie gute Aussicht darauf, mit nur einem zusätzlichen Semester einen guten Abschluss zu erlangen.

Jetzt also wieder zurück zur Situation, wenn Sie alles aus den ersten drei Semestern bestanden haben. Sie haben dann ein grundlegendes Verständnis für zwei Bereiche der Softwareentwicklung: FPGA-basiert und imperativ bzw. objektorientiert.

Wenn Sie sich jetzt im Bereich der maschinennahen Softwareentwicklung bzw. bei der Entwicklung von **FPGAs** wohler fühlen, dann sprechen Sie die Kollegen Edeler und Behrens an; diese können Ihnen die spannenden Möglichkeiten dieser Bereiche aufzeigen.

Wenn Sie sich dagegen eher bei **Desktoprechnern** und **Smartphones** zu Hause fühlen, dann sind Sie im Bereich der Web- bzw. der Anwendungsentwicklung richtig. Sie können nun Spezialkenntnisse erwerben, um z.B. eine guter Entwickler für Android- oder iPhone-Apps zu werden. In dem Fall wäre Herr Pläß der Ansprechpartner Ihrer Wahl.

Um Software für mobile und/oder vernetzte Systeme entwickeln zu können, nutzen Sie die Kenntnisse aus PRG, P1 und P2 sowie Software Engineering, RDB, NWI und Kryptologie für Media Systems aus und setzen Sie diese Kenntnisse in Projekten um.

Sie wollen es genauer wissen? Gut: Aktuell (Herbst 2015) gibt es vorrangig Stellengesuche nach Leuten mit Kenntnissen in den folgenden Bereichen:

- **Programmierung in C bzw. C++**
Diese Stellen sind meist eher etwas für reine Informatiker, da hier umfangreiche Kenntnisse im Bereich der Algorithmik nötig sind, die in Ihrem Studienplan leider vollständig fehlen, obwohl sie ein der Grundlagen jeder Informatik und damit auch der Medieninformatik sind.
- **Programmierung in .NET und angrenzenden Bereichen**
Das sind Spezialisten, die sich in die Softwareentwicklung mit Sprachen vertieft haben, die von Microsoft entwickelt wurden und ausschließlich auf Windows-Rechnern und –Smartphones genutzt werden können.
- **Programmierung in Objective-C oder C#**
Hier wird's schon interessanter, weil Sie zwar diese Sprachen in Ihrem Studium nicht kennen lernen, diese Sprachen aber eine sehr große Ähnlichkeit mit Java aufweisen.
- **Programmierung in Java für Server**
Einerseits lernen Sie im Studium zwar Java, andererseits belegen Sie keine Kurse zu Betriebssystemen bzw. zur Serverwartung. Und genau das bräuchten Sie dafür.
- **Programmierung in HTML, CSS und PHP oder JavaScript**
Hier sind Sie genau richtig, wenn Sie im Department Medientechnik der HAW Hamburg studieren, denn die Grundlagen erlernen Sie

im ersten Semester im Kurs "Einführung ins Programmieren," bzw. „Programmieren 1“. Deshalb können Sie sie leicht ausbauen.

Bei den aktuellen Stellenanzeigen wird dann noch nach weiteren Kenntnissen gefragt. Damit Sie nachvollziehen können, was darunter verstanden wird und Sie in den noch folgenden Semestern die Möglichkeit haben, sich jeweils einzuarbeiten, folgen ein paar Erklärungen:

- **HTML5 und CSS3**

Zunächst sind das die beiden Sprachen, in denen Sie Webpages programmieren (HTML) und das Design festlegen (CSS). Allerdings ist noch hinzuzufügen, dass die Änderungen in HTML5 (gegenüber der Version 4) so umfangreich sind, dass es eigentlich eine komplett neue Programmiersprache ist. So ist vieles, was früher mit Hilfe von PHP oder JavaScript programmiert werden musste Teil von HTML5. Aber das ist nur ein kleiner Teil von HTML5, der im Grunde auch als HTML 4.1 hätte veröffentlicht werden können. Dagegen beinhaltet die Version 5 mit den sogenannten **Microdata** eine umfangreiche und standardisierte Semantik, womit es eine der ersten vollwertigen semantischen Programmiersprachen sein dürfte. Das ist gleichbedeutend mit derart fundamentalen Änderungen bei der Entwicklung von Anwendungen, dass die Auswirkungen ähnlich wie beim WWW erst in zehn bis zwanzig Jahren spürbar werden dürften. Aber sie dürften ähnlich weite Kreise ziehen wie die Entwicklung des WWW selbst.

Im Gegensatz dazu ist CSS nichts anderes als eine Sammlung von Befehlen, mit denen sich die Anzeige von Elementen innerhalb einer Webanwendung anpassen lässt. Es gibt dort sicher einiges, was Sie sich ansehen könnten, aber für MedieninformatikerInnen ist es ungefähr so wichtig, die Feinheiten der CSS-Programmierung zu beherrschen wie die genaue Bestimmung der Farbe der eigenen Tastatur. Sie halten das für vollkommen überflüssig und fragen sich, warum CSS dann überhaupt erwähnt wird? Na dann willkommen im Club: Der einzige Grund, sich als MedieninformatikerIn mit CSS zu beschäftigen ist der, dass es MediendesignerInnen gibt, die nicht einmal im Stande sind, eine Definition für einen Farbwert z.B. RGB-Werte in einen Computer einzugeben. Und dann wird diese Aufgabe eben an MedieninformatikerInnen delegiert...

Wichtig

Wenn ein Arbeitgeber explizit nach HTML und CSS fragt, aber sonst kaum nach Kenntnissen, die in dieser Aufstellung auftauchen, dann geht es in aller Regel um eine Stelle als Webdesigner und nicht um eine Stelle als Webdeveloper. (Letzteres wäre Ihr Gebiet, für ersteres sind Sie im falschen Studiengang.)

- **CSS Präprozessoren**

Dieses Schlüsselwort weist eindeutig auf eine Stelle für Webdesigner hin.

Beispiele: **Twig, Gulp, SASS, LESS.**

- **XHTML**

ist eine Kombination aus HTML und XML. Diese Kombination ist im Grunde durch HTML5 und Microdata oder andere Formen des semantic web in den meisten (aber nicht allen Bereichen) überflüssig geworden. Über das semantic web reden wir gleich in der ersten Veranstaltung des Kurses „Einführung in die Programmierung“.

- **JavaScript oder PHP**

Für die Einarbeitung in PHP sollten Sie nicht allzu lange brauchen, wenn Sie eine andere imperative Sprache (wie Java) beherrschen, weil PHP nur wenige Konzepte der imperativen Programmierung umsetzt.

Bei JavaScript sieht das anders aus: Der Einstieg ist zwar nicht schwer, aber um gute Software zu entwickeln (was in PHP eher nicht möglich ist), werden Sie schon ein Jahr Übungszeit brauchen. Das liegt auch daran, dass JavaScript u.a. das funktionale Programmierparadigma umsetzt. Entwickler, die nur die klassenbasierte Objektorientierung (z.B. aus Java) kennen sind damit in aller Regel überfordert, ohne es zu merken. Aufgrund der Möglichkeiten, die JavaScript bietet und sein Status als Standard-Programmiersprache für dynamische HTML5-Seiten, dürfte PHP in zehn Jahren kaum mehr von Belang sein, wenn die Entwickler der Sprache hier keine fundamentalen Änderungen einführen. Leider sieht es zurzeit nicht so aus, als wenn das bei PHP7 der Fall wäre. <https://github.com/php/php-src/blob/php-7.0.0RC1/UPGRADING> Es scheint so, als wenn hier lediglich Neuerungen eingeführt werden würden, die in anderen Sprachen wie Ruby längst üblich sind oder die einfach selbst programmiert werden können. (Stichworte: UTF-8-Unterstützung oder die Einführung eines expliziten Spaceship-Operators)

- **ECMAScript**

ist eine standardisierte Version u.a. von JavaScript. Wenn Sie JavaScript beherrschen, sollten Sie sich hier relativ schnell zurecht finden. Umgekehrt gilt dasselbe.

- **JavaScript ... frameworks**

Beispiele: **AngularJS, Backbone, EmberJS, Grunt, jQuery, requireJS, AJAX.**

- **PHP ... frameworks**

Im Falle des **Zend** framework handelt es sich bei Zend nicht um eine kleine Erweiterung von PHP, sondern um ein sehr mächtiges Werkzeug, das eine längere Einarbeitungszeit benötigt. Dieses setzt u.a. voraus, dass Sie PHP objektorientiert programmieren können.

- **Flash und ActionScript**

sind praktisch dasselbe. ActionScript ist eine Konkurrenzsprachen zu JavaScript, deren Rechte bei Adobe liegen. Da JavaScript als Standardsprache für dynamische Webpages in HTML5 festgelegt wurde und es im Gegensatz zu ActionScript kein Unternehmen gibt, das hier Rechte beanspruchen würde, dürfte ActionScript ähnlich wie PHP in zehn Jahren nur mehr eine Nischensprache sein.

Die Abkürzung AS wird teilweise für ActionScript verwendet, aber nicht nur dafür: Im Bereich der Programmierung von **SPSen** gibt es eine Programmiersprache namens Ablaufsprache, die ebenfalls mit **AS** abgekürzt wird. Im Englischen wird diese Sprache als SFC / Sequential Function Chart bezeichnet. ActionScript und Ablaufsprache haben so viel miteinander gemein wie Michael Schuhmacher mit einem Chinesen, der Tai Chi macht.

- **Frontend und BackendBackend**

stehen für zwei Bereiche, die bei PHP noch strikt getrennt waren. (Auf die Details kommen wir bei der Einführung in PHP zu sprechen.) Meist ist mit Frontend die Programmierung des View gemeint (auch dazu kommen wir bald) und damit geht es dann in aller Regel um eine Aufgabe für Webdesigner. Das muss aber nicht so sein; sehen Sie sich ggf. die Stellenausschreibung genau an und fragen Sie beim Arbeitgeber nach. Beim Backenddevelopment haben Sie dagegen nie etwas mit Gestaltung zu tun. Als Studierende der Medieninformatik wäre das also der Bereich, in dem Sie vorrangig tätig werden. Für die Entwicklung des Frontends brauchen Sie immer zumindest eineN MediendesignerIn mit Grundkenntnissen der Typographie, da nur diese ein wenigstens ausreichendes Verständnis für die Darstellung medialer Inhalte haben. Hier nochmal der entsprechende Hinweis: Medieninformatik und Mediendesign arbeiten zwar beide im weitesten Sinne mit Medien, aber ersteres ist eine Spezialisierung der Informatik, letzteres eine Spezialisierung des Designs. Und wer aus dem einen Bereich kommt, kann bestenfalls verstehen, worüber die SpezialistInnen des anderen Bereichs reden. Mehr nicht!

Grundsätzlich ist die Trennung in Front- und Backend aber recht

willkürlich, weil damit suggeriert wird, es gäbe eine klare Trennung, wo eigentlich keine ist. Denn an welcher Stelle (auf dem Rechner eines Nutzers oder auf einem Server im Netz) Sie bei einem Softwareprojekt welche Funktionalitäten und Komponenten realisieren, ist Ihre Entscheidung.

- **Versionsverwaltung**
(siehe Abschnitt „SCM / Versionskontrolle“)
- **Continuous Integration**
ist ein relativ neues aber sehr mächtiges Konzept, wird teilweise mit CI abgekürzt, was aber leider auch für andere Begriffe stehen kann. Im Grunde ist es eine massive Erweiterung der Versionsverwaltung. Denn hier werden auch Tests und anderes in die Entwicklung eingeführt, das Sie später in der Veranstaltung Software Engineering kennen lernen.

Beispiel: **Jenkins**

- **Continuous Deployment**
ist dann gewissermaßen die aktuelle Luxusklasse fürs Software Engineering. Denn hier wird eine Software auch nach der Auslieferung an Kunden kontinuierlich weiter entwickelt. An bestimmten Punkten werden dann die Neuerungen an den Kunden ausgeliefert, um beispielsweise Fehler zu bereinigen (**Patchen**) oder neue Funktionalitäten in die Software einzuführen.
- **Responsive Design**
hat streng genommen nichts mit Design zu tun. Hier geht es darum, Webanwendungen zu entwickeln, die auf den unterschiedlichsten Displays nutzbar sind. Dieses Thema werden wir in in „Programmieren 1“ (für Medientechnik) bzw. „Einführung in die Programmierung“ (für Media Systems) kurz behandeln. Es gibt eine Vielzahl an JavaScript Frameworks, die Ihnen hier viel Arbeit abnehmen, die aber leider fast ausschließlich für HTML 4.01 gedacht sind.
- **Strukturiertes Arbeiten**
Ein Arbeitgeber, der danach fragt ist für Sie die passende Wahl... außer wenn Sie eigentlich im falschen Studium gelandet sind.
- **UX / UI (User Experience, User Interface)**
sind Begriffe, die in den Bereich Webdesign und **Usability** fallen.
- **Photoshop**
gehört ins Webdesign.

- **Design Patterns**

Seitdem die objektorientierte Programmierung in vielen professionellen Unternehmen genutzt wird, haben sich einige de-facto Standards ausgebildet, die die Arbeit im Team deutlich erleichtern. Damit werden Sie sich in der Veranstaltung Software Engineering auseinander setzen. Eines davon werden Sie aber schon in dieser Veranstaltung kennen lernen.

Beispiel: **MVC**

- **Agile Softwareentwicklung**

setzt objektorientierte Softwareentwicklung voraus und ist ein aktuelles Buzz Word: Viele Menschen benutzen es (in der Softwareentwicklung) aber leider gibt es hier wie bei der Objektorientierung eine Reihe von Missverständnissen. So gibt es ein Unternehmen, in dem ernsthaft versucht wird, agile Softwareentwicklung zu nutzen ohne dabei zentrale Aspekte der Objektorientierung zu verwenden. Und das ist unmöglich. Auch dieses Konzept lernen Sie in der Veranstaltung Software Engineering kennen.

Beispiele: **TDD** (Test-Driven-Development), **Iterationen** im Projektmanagement, **SCRUM**

- **Extreme Programming**

diese Art der Softwareentwicklung geht in eine andere Richtung als die bislang besprochenen Varianten. Hier wird im Regelfall auf eine strukturierte Vorgehensweise verzichtet, wenn dadurch Kundenwünsche so schnell wie möglich integriert werden können.

Schlagwörter: **YAGNI**

- **Skalierbarkeit**

wenn dieser Begriff auftaucht, geht es um die Entwicklung von Anwendungen oder Systemen, die möglichst gut damit umgehen sollen, wenn die Anzahl Nutzer oder Daten, die jeweils auf die Anwendung oder das System zugreifen oder von diesem genutzt werden zum Teil innerhalb kurzer Zeit stark ansteigen oder abfallen können bzw. stark schwanken. Dieser Bereich kommt für Sie leider nicht in Frage, weil Sie dafür mindestens die Veranstaltungen „**Algorithmen und Datenstrukturen**“ sowie „**Algorithmen design**“ (Praktische Informatik 1 und 2) erfolgreich abgeschlossen haben müssen, die in unserem Department nicht auf dem Lernplan stehen.

Hier noch ein paar Begriffe, die jeder Arbeitgeber in diesem Bereich erwartet, weshalb sie eigentlich überflüssig sind:

- Motivation
- Belastbarkeit
- Teamfähigkeit
- Kenntnis von Webstandards

Und nun noch ein paar Begriffe, bei denen Sie von einer Bewerbung absehen sollten, wenn Sie als Softwareentwickler ins Webdevelopment gehen wollen: So wie es wenig Sinn macht, Webdevelopment und Webdesign von einer Person durchführen zu lassen, macht es auch keinen Sinn, Webdevelopment und Serveradministration von einer Person durchführen zu lassen: Entweder ist diese Person sehr fähig (und damit sehr teuer) oder beherrscht nur einen der beiden Bereiche und das wäre sehr problematisch:

- **Linux-/UNIX-Administration**
Nicht zu verwechseln mit Linux-/UNIX-Kenntnissen; die müssen Sie bis zum Ende des Studiums zumindest grundlegend erworben haben, wenn Sie Media Systems studieren. Medientechnikstudierende, die in irgend einer Form mit Rechnern arbeiten wollen, sollten hier aber zumindest lernen, wie sie Befehle über die sogenannte Konsole eingeben können.
- **Konfiguration von ...-Servern (z.B. Linux, Apache)** wird gefordert.
Die Administration von Servern hat nichts mit Softwareentwicklung zu tun, wie Sie sie im Rahmen von Veranstaltungen wie Programmieren 1 und 2 kennen lernen. Hier geht es vielmehr um Aspekte der Rechteverwaltung, Abwehr von netzbasierten Angriffen durch konkurrierende Unternehmen oder das organisierte Verbrechen und ähnliches. Wenn ein Unternehmen also jemanden sucht, der sowohl als Webdeveloper als auch als Serveradministrator fähig ist, dann herrschen dort offenkundig sehr große Wissensdefizite vor oder die Bezahlung ist entsprechend der geforderten Kenntnisse. Das würde dann aber bedeuten, dass Sie erst mit mehrjähriger Praxis in diesem Bereich die nötigen Kenntnisse erworben haben werden.
- Formulierungen wie **„Gängige Webtechnologien wie HTML, CSS und ... sind Ihnen nicht fremd“** weisen eher darauf hin, dass dem zuständigen Mitarbeiter des Unternehmens diese Programmiersprachen und die ihnen zugrunde liegenden Konzepte sehr fremd sind.
- Ebenfalls abzuraten ist von Anzeigen, in denen Ihre Aufgaben wie folgt beschrieben werden: **„Development of the next generation of ...“** Denn wenn Sie das wirklich könnten, dann sollten Sie sich lieber einen Investor suchen, der Sie bei der Umsetzung Ihrer Ideen

unterstützt. In 99% aller Fälle geht es hier um eine Unternehmen, dessen „Fachkräfte“ derart wenig über die Entwicklungen im Bereich der Softwareentwicklung wissen, dass sie glauben, eine abgerundete Ecke sei bereits eine historisch bedeutsame Produktentwicklung. (Nichts gegen abgerundete Ecken... Die sehen schick aus... findet jedenfalls der Autor dieses Buches.)

- Problematisch ist es, wenn bei Programmiersprachen keine Versionsnummern angegeben werden. Vielleicht kennen Sie genau die geforderte Version nicht, aber darüber lässt sich im Bewerbungsgespräch immer reden. Dagegen macht es beispielsweise einen großen Unterschied, ob Sie tatsächlich HTML5 oder eigentlich nur HTML4.01 beherrschen. Und umgekehrt ist es auch für Ihre berufliche Zukunft relevant, ob Sie für Arbeitgeber tätig werden, dessen SoftwareentwicklerInnen diesen Unterschied kennen oder eben nicht.
- Ebenfalls kritisch ist es, wenn einerseits nach einem „Junior ... Developer“ gesucht wird, andererseits aber eine sehr umfangreiche Palette an Kenntnissen gefordert wird. Denn einerseits wird damit angegeben, dass ein Einsteiger gesucht wird, der also gerade erst seinen Abschluss gemacht hat, andererseits werden derart viele Kenntnisse gefordert, dass im Grunde drei Jahre Berufserfahrung dafür nötig sind.

Kommen wir nun zu einem Bereich, den Sie schlicht deshalb ignorieren sollten, weil das eine Spezialdisziplin für Wirtschaftsinformatiker bzw. Informatiker mit mehrjähriger Praxiserfahrung in Unternehmen einer Branche ist:

- **ERP (kurz für Enterprise Resource Planning)**
umfasst die verschiedensten Programme, die in Unternehmen zum Einsatz kommen. Hier ist neben der Kenntnis der jeweiligen Software ein detailliertes Verständnis für Unternehmensstrukturen und die Feinheiten der Branche nötig, in dem das Unternehmen tätig ist.

Beispiele: **SAP, Microsoft Dynamics**

1.4 Zusammenfassung

Nach diesem Kapitel wissen Sie, dass es eine Vielzahl von Möglichkeiten gibt, den Begriff des Programmierens zu verstehen, und dass InformatikerInnen den Begriff des Paradigmas nutzen, um zwischen diesen Möglichkeiten zu unterscheiden. Einige davon sind Teil dieses Buches. (Deshalb ja

auch der Titel.) Viele andere werden Sie in den nächsten Jahren kennen lernen, als Teil Ihres Studiums oder auch bei anderer Gelegenheit.

Sie wissen, dass es neben reinen Programmiersprachen Dinge wie IDEs, Bibliotheken, Frameworks und Middlewares gibt, die Ihnen einen Teil Ihrer Arbeit abnehmen.

Dann haben Sie etwas über Teamarbeit gehört, über die drei Teilbereiche der Informatik: Praktische, Theoretische und Technische Informatik. Und Sie haben insbesondere etwas darüber gehört, warum keines dieser drei Teilgebiete das gleiche ist wie Programmierung. Anschließend ging es um die Unterschiede zwischen den Inhalten der Informatik an Uni und Fachhochschule. Abgeschlossen wurde das ganze mit einem kurzen Überblick über den Arbeitsmarkt und die Begriffe, die Ihnen dort begegnen werden.

Was Sie jetzt aber noch nicht wissen ist, wie Sie mit all diesen Dinge umgehen sollen, bzw. wie Sie sie nutzen können. Keine Sorge, genau darum geht es ja in dieser Veranstaltung. Also sein Sie nicht frustriert, wenn Ihnen dieses Kapitel zu oberflächlich erscheint; es ist lediglich ein erster Einblick.

An dieser Stelle ein Anmerkung, die in der Wissenschaft (also auch in der Informatik bzw. in Ihrem Studium) immer und überall gilt: Alles wird kontinuierlich weiter entwickelt und wer glaubt, dass er ein fähiger Wissenschaftler ist, ohne sich in seinem Bereich ständig auf dem aktuellen Stand zu halten, der macht etwas falsch. Dementsprechend sollten Sie sich stets vor Augen halten: Was Sie hier lernen, ist Wissen, dass in wenigen Jahren so nicht mehr aktuell sein wird. Und Sie werden kontinuierlich prüfen müssen, welche Aspekte sich geändert haben. Denn sonst werden Sie in spätestens zehn Jahren nicht mehr verstehen, worüber in der Informatik geredet wird. Und das ist auch ein zentrales Element wissenschaftliches Arbeit: Hinterfragen und selbst prüfen, was man gehört hat und kontinuierlich die bestehenden Systeme verbessern.

Das ist übrigens eine der Besonderheiten der (Medien-)informatik: Einerseits ist sie so abstrakt, dass sie auf viele so abschreckend wirkt wie ein Gemälde von Pablo Picasso, andererseits in einer derart schnellen Veränderung, dass Kenntnisse zum Teil innerhalb von Monaten veraltet sind. Während also Ihre Kommilitonen z.B. in Elektrotechnik 1 und 2 Kenntnisse erlangen, die so noch in fünfzig Jahren nahezu gleich sein werden, müssen Sie als angehende (Medien-)informatikerInnen sich ständig mit den Änderungen auseinander setzen, weil Sie sonst nichts mehr von dem verstehen, was in Ihrem wissenschaftlichen Bereich passiert.

Sollten Sie allerdings zu denjenigen gehören, die eigentlich **Mediendesign**

studieren wollten und die keine Lust haben, sich mit Fragen der Informatik und der Programmierung zu beschäftigen, dann möchte ich Sie auf Ihre Prüfungsordnung bzw. das Modulhandbuch des Studiengangs Media Systems hinweisen: 110 der 180 Credit Points Ihres Studiums liegen im Bereich der Informatik. Es wäre zwar schön, wenn ich durch dieses Buch oder die von mir angebotenen Veranstaltungen Ihr Interesse am Informatikteil der Medieninformatik bzw. von Media Systems wecken kann, aber Media Systems, Medieninformatik und Medientechnik sind Bachelors of Science und da sind die gestalterischen Anteile naturgemäß deutlich dünner gesät, als bei einem Bachelor of Arts. Wo Ihre Design-Kommilitonen kreativ übers Papier streichen, um Gefühlen Ausdruck verleihen, konzipieren Sie mit mathematischen und computerisierten Abstrakta, um umfangreiche Problemstellungen zu lösen.

Nach diesen freundlich gemeinten Ermahnungen wünsche ich Ihnen viel Erfolg bei dieser Veranstaltung und in Ihrem Studium. Lassen Sie sich von Rückschlägen nicht entmutigen und tun Sie das, wofür der Begriff Studium steht: Sich intensiv mit etwas beschäftigen.

Kapitel 2

Von der Nachrichtentechnik zur Programmierung

Computer sind eine Kombination aus Bauteilen und Datenübertragungsleitungen. In vielen Programmiersprachen brauchen ProgrammiererInnen sich nicht direkt um die Datenübertragung zu kümmern. Allerdings folgen aus der Datenübertragung einige Fakten, die wir bei der Programmierung in jeder imperativen Programmiersprache beachten müssen. Tun wir das nicht, dann sind Fehler die Folge, die wir ohne ein allgemeines Verständnis der Grundlagen von Datenübertragungen nicht verstehen und damit lösen können.

2.1 Nachrichtentechnik – So kommen Nullen und Einsen in den Rechner.

Sie haben schon öfter gehört, dass ein Computer im Kern mit Nullen und Einsen arbeitet. Was den ein oder anderen vielleicht zu Spekulationen über die Qualität dieser Geräte gebracht haben könnte... Wer arbeitet schon freiwillig einen Großteil seiner Zeit mit Nullen zusammen?

Was Sie aber in aller Regel in einem Informatikstudium nicht hören, ist dass diese Folgen von Einsen und Nullen nur ein Hilfsmittel sind, eine **Repräsentation** dessen, was tatsächlich vorhanden ist. Doch wenn Sie etwas darüber hören, dann wird Ihnen in aller Regel erzählt, dass damit der Unterschied zwischen fließendem oder nicht fließendem Strom dargestellt wird. Das ist so aber in aller Regel falsch: Es gibt die unterschiedlichsten Formen von Datenübertragungen, deren Signale am Ende als Folgen von Nullen und Einsen **interpretiert** werden. Und nur die am einfachsten zu verstehende Form ist die, bei der Signale erzeugt werden, indem zwischen fließendem und nicht-fließendem Strom unterschieden wird. Dieje-

nigen von Ihnen, die Medientechnik studieren werden sich damit wenigstens fünf Semester lang beschäftigen, auch wenn Ihnen das anfangs also in Veranstaltungen wie Elektrotechnik 1 und 2 gar nicht bewusst sein wird.

Es folgen zwei Beispiele dafür, wie Einsen und Nullen auch anders dargestellt werden können:

- Am Karlsruher Institut für Technologie (KIT¹) wird an der Möglichkeit geforscht, wie man Daten in Molekülen aus Lachs DNA speichern kann. Hier wurden Nullen mit 0,4 V und Einsen mit 0,9 V repräsentiert.
- Im Bereich der Speicherung mit DNA gibt es aber auch andere Ansätze. Wie Sie aus dem Biologieunterricht wissen, besteht DNA aus vier Molekülen, die als sogenannte Basensequenzen endlose Ketten bilden können. Und richtig, auch diese Basensequenzen kann man nutzen, um damit Einsen und Nullen darzustellen. Hier steht also gar kein Stromfluss für Nullen und für Einsen, weil sie chemisch und nicht elektrisch repräsentiert werden. DNA bietet dabei eine derart hohe Dichte pro Bit an, dass Sie den Inhalt von 50 Millionen BlueRay Disks in einer Kaffeetasse unterbringen können². Na gut, vielleicht war es auch ein Übersetzungsfehler und im Originalartikel war die Rede von einem Kaffeebecher, aber spielt das eine Rolle? Denn das bedeutet, dass Sie die gesamte Videothek der Welt in einem Kaffeebecher herumtragen könnten. Und überlegen Sie jetzt, wie viel Raum eine solche Videosammlung in Form von BlueRays oder Festplatten einnehmen würde.

Die wissenschaftliche Disziplin, die sich unter anderem mit der Frage beschäftigt, wie man Nullen und Einsen übertragen kann, nennt sich **Nachrichten- und Kommunikationstechnik** und ist ein Teilbereich der **Elektrotechnik**. Während es bei der Datenübertragung innerhalb eines Computers noch recht problemlos möglich ist, mittels Ein- und Ausschalten von Stromflüssen Signale zu erzeugen und zu interpretieren, ist das bei Übertragungen über längere Distanzen eine eher ineffiziente Methode. Deshalb werden hier Schwingungen manipuliert, um die Datenübertragung zu realisieren. Das wichtigste mathematische Werkzeug ist dabei die **Fourier-Transformation** (kurz FT). Diese transformiert Nullen und Einsen in eine Schwingung, indem Winkelfunktionen auf eine bestimmte Weise beliebig häufig angewendet werden. (Je häufiger, desto präziser.)

In der Nachrichtentechnik spielen dann Begriffe und Ansätze eine Rolle, die kaum etwas mit dem gemein haben, was die Tätigkeit von InformatikerInnen bestimmt. Während die **Media Systems** Studierenen sich also vor-

¹<http://www.kit.edu>

²<http://www.spektrum.de/news/auf-petabyte-programm/1182773>

rangig damit beschäftigen Gesamtaufgaben in Teilaufgaben zu unterteilen und mittels abstrakter Konzepte die Teilaufgaben möglichst übersichtlich zu gestalten und sie dabei möglichst effizient zu lösen, nutzen **Medien- und NachrichtentechnikerInnen** Formeln und Funktionen, um Daten so effizient wie möglich über einen bestimmten Leitungstyp zu übertragen.

Kontrolle

Die Nachrichtentechnik entwickelt die Systeme, die es uns überhaupt erst ermöglichen, Informatik zu betreiben. Einsen und Nullen sind zwar die Basis der IT-Systeme, die wir programmieren, was aber tatsächlich bei der Datenübertragung im Computer oder im Internet passiert, um Einsen und Nullen zu übertragen oder besser gesagt um eine jeweils passende Repräsentation für die Übertragung von Einsen und Nullen zu finden, hat damit größtenteils nichts gemein.

2.2 Codierung – Was steht wofür?

Eben haben Sie gelesen, dass Nullen und Einsen nur die Interpretation von z.B. Stromflüssen sind. Und damit sind wir bei einem zentralen Begriff, mit dem die meisten InformatikerInnen sich eher ungern beschäftigen, weil es dabei ausschließlich um die Frage geht, wie etwas interpretiert wird. Der Bereich, von dem hier die Rede ist, wird als **Codierung** bezeichnet.

Vielleicht haben Sie schon etwas vom **ASCII**-Code gehört. Der ASCII-Code ist nichts anderes als eine Tabelle, bei der jedem Buchstaben und verschiedenen anderen Zeichen eine Zahl zugeordnet wird. Wenn Sie nur mit iOS- oder Windows-Rechnern arbeiten, haben Sie mit der Codierung im Regelfall nichts zu tun, aber sobald Sie Daten zwischen Computern austauschen, müssen sie darauf achten. Denn im Hintergrund werden alle Daten, die Sie eingeben in Form von Codes gespeichert. Wenn Sie Daten auf einen anderen Rechner übertragen, dann werden die Codes ausgetauscht. Und nur wenn beide Rechner die gleiche Codierung verwenden empfängt der zweite Computer die Daten, die Sie eingegeben haben.

An dieser Stelle werden wir uns einen Begriff ansehen, der Ihnen immer wieder mit jeweils unterschiedlicher Bedeutung begegnen wird: **Interfaces** bzw. **Schnittstellen**. Ein Interface ist etwas, das die Verbindung zwischen zwei „Dingen“ herstellt. Wenn wir über den Datenaustausch zwischen zwei Rechnern reden, die beide unterschiedliche Codierungen verwenden, dann ist eine mögliche Bedeutung des Begriffs Interface der eines Übersetzers. Das Interface kennt dann die beiden Codierungen und wandelt die übertragenen Daten von der einen Codierung in die andere um, damit der Empfänger Daten in einer Codierung erhält, die er versteht.

Hier schon einmal ein Beispiel für eine ganz andere Bedeutung des Begriffs Interface: Bei der Programmierung in objektorientierten Sprachen kann ein Interface eine abstrakte Definition für einen Programmteil sein, der noch programmiert werden muss. Diese Interpretation des Begriffs Interface ist dann sinnvoll, wenn ein Programm im Team entwickelt werden soll. Das Interface enthält dann die Festlegung von Namen für „Befehle“, die zwar noch nicht funktionieren, aber bei denen schon festgelegt ist, was sie später tun sollen. So können dann alle Mitglieder des Teams mit Ihren Aufgaben beginnen: Sie können zwar Ihren Programmcode noch nicht ausprobieren, aber da Sie bereits wissen, wie die Befehle lauten, die später eine bestimmte Funktion erfüllen werden, können Sie zumindest schon mit der Entwicklung neuer Programmteile beginnen.

Ähnlich wie für den Begriff des Programmierens gilt auch für den Begriff der Codierung, dass er eine Vielzahl unterschiedlicher Methoden zusammenfasst, die jeweils für einen bestimmten Anwendungsfall sinnvolle Lösungen anbieten. Mehr dazu können Sie in dem großartigen Buch „**Information und Codierung**“ von **R.W. Hamming**³ nachlesen. Codierung ist eines der zentralen Themen der Nachrichtentechnik und wird einführend in Veranstaltungen zur **Technischen Informatik** behandelt.

Im Bereich der **maschinennahen Programmierung** kommt dann noch dazu, dass es zwei unterschiedliche Reihenfolgen gibt, in der Daten im Speicher abgelegt werden können. Die unterscheiden sich jedoch nicht (!) dadurch, dass Sie einfach nur spiegelverkehrt wären. Auch hierüber müssen Sie Bescheid wissen, wenn Sie zwischen zwei Computer Daten austauschen wollen. Denn selbst wenn Sie sichergestellt haben, dass die Codierung des einen Computers richtig in die Codierung des anderen übersetzt wird, kann Ihnen die unterschiedliche Reihenfolge im Speicher noch alles zerschießen. Die meisten von Ihnen werden dieses Problem aber nicht haben, da die Prozessoren heutige Rechner von Apple und Microsoft die gleiche Reihenfolge bei der Datenspeicherung nutzen. Vor einigen Jahren, als Apple noch Prozessoren von Motorola verwendete, war es dagegen eines der zentralen Probleme beim Datenaustausch zwischen Rechnern mit dem Betriebssystem beider Rechner.

Mit Codierungen bekommen es bereits Programmiereneinsteiger sehr schnell zu tun, auch wenn wir dann von **Datentypen** sprechen. Denn letztere basieren im Grunde auf der Codierung: Sobald Sie eine Zahl oder eine andere Zeichenfolge in einem Programm verwenden, kann es dazu kommen, dass Ihr Programm vermeintlich falsche Ergebnisse liefert. Aber wie schon aber

³Hamming ist gewissermaßen der Godfather der Nachrichtentechnik.

geschrieben: Das ist nicht der Fehler der Rechner, sondern der Fehler des Entwicklers, also ggf. von Ihnen, da hier die Arbeitsweise des Rechners ignoriert wurde.

Kontrolle Texte und andere Inhalte werden bei der Speicherung codiert. Das bedeutet in der Informatik, dass Sie als Folgen von Binärziffern im Speicher liegen. Je nachdem, wie oder was Sie programmieren, werden Sie detaillierte Informationen über verschiedene Codierungsverfahren benötigen.

2.3 Informatik und Nachrichtentechnik - Die zanken- den Geschwister

Wie beschrieben beschäftigen sich **NachrichtentechnikerInnen** vorrangig mit der Frage, wie Daten mit möglichst geringem Energieaufwand so übertragen werden können, dass sie am Ende der Leitung empfangen und verstanden werden können. **InformatikerInnen** dagegen beschäftigen sich vorrangig mit der Frage, wie Daten möglichst schnell verarbeitet werden können.

Aus diesen unterschiedlichen Perspektiven resultiert eine vollständig andere Herangehensweise an Aufgaben. Und leider führt das häufig zu einer Trennung zwischen NachrichtentechnikerInnen und InformatikerInnen bzw. zwischen Ihnen und Ihren Kommilitonen der Medientechnik bzw. in der Medieninformatik. Dabei ließen sich ungemein spannende Projekte realisieren, wenn Sie es nur schafften, diese Kluft zu überwinden. Somit verfolgt dieses Kapitel auch das Ziel, Ihnen ein wenig Verständnis für dieses ungemein spannende aber auch höchst anspruchsvolle Gebiet zu vermitteln.

Das was wir heute als Computer bezeichnen, ist in aller Regel eine Art Black Box, in der mehrere Hundert Komponenten unabhängig voneinander arbeiten und über verschiedene Arten von Datenübertragungswegen miteinander kommunizieren. Vermutlich sind Sie jetzt skeptisch, da Sie wissen, dass in einem Computer Komponenten wie das Mainboard, Festplatten, Grafikarten und ähnliches vorhanden sind. Der Begriff der Black Box gilt insbesondere bei Smartphones, wo sie im Gegensatz zu den meisten anderen Rechnern nicht einmal mehr eine Grafikkarte oder einzelne Laufwerke erkennen können. Dennoch ist es so, denn wenn Sie beispielsweise ein Mainboard als eine einzige Komponente betrachten, dann ignorieren Sie, dass es sich bereits hier um eine Ansammlung von Dutzenden Komponenten handelt. Und dann gibt es zusätzlich noch Technologien wie die **VLSI**, die very large scale integration, bei der es darum geht, eine sehr

große Anzahl von eigenständigen Einheiten in einem Bauteil zu integrieren.

Wie Sie in den Veranstaltungen der **Technischen Informatik** lernen, besteht ein Prozessor (fachlich korrekt ein **Mikroprozessor**) mindestens aus den drei Komponenten Rechenwerk, Steuerwerk und Speicher sowie den Verbindungen dazwischen, die als Bus bezeichnet werden. Das ist eine Abstraktion, die einem Prozessor aus den 50er Jahren entspricht, die aber genau dem entspricht, was wir bei der **imperativen Programmierung** beachten müssen. Wenn Sie mehr über den Aufbau von Computern mit Mikroprozessoren wissen wollen, werfen sie beispielsweise einen Blick in den Band „**Rechnerarchitektur**“ von **Paul Herrmann**.

Wenn wir uns nun die Arbeitsweise eines Computers unvoreingenommen⁴ ansehen, dann ist die wichtigste Komponente aber nicht „der“ Prozessor, sondern die Vielzahl der Kommunikationswege zwischen all den Komponenten, aus denen er besteht. Die bekannteste Art dieser Kommunikationswege wird als **Bus** bezeichnet. Womit wir bei der Antwort auf die Frage wären, warum anstelle der Bezeichnung Nachrichtentechnik häufig den Begriff der **Kommunikationstechnik** benutzt wird und warum die auch für **InformatikerInnen** so wichtig ist: Wenn die Komponenten unseres Rechners keine Daten austauschen könnten, könnten wir mit Ihnen exakt gar nichts anfangen. Ohne die Arbeit der Kommunikations- und Nachrichtentechnik wäre unsere Arbeit also gar nicht möglich.

Für Einsteiger in die imperative Programmierung scheint diese Aufteilung in Prozessor, Bus und Speicher irrelevant zu sein, doch das ist schlicht falsch: Die meisten Fehler und Missverständnisse von Einsteigern kommen schlicht dadurch zustande, dass Ihnen häufig nicht erklärt wird, wie eine **Variable** im Speicher abgebildet wird und was der **Datentyp** damit zu tun hat. Damit Sie professionell Software entwickeln können müssen Sie aber die aus der genannten Dreiteilung erwachsenden Probleme und Lösungen kennen: Wenn Sie später scheinbar simple Dinge wie den Zugriff auf eine Datei (Speichern bzw. Laden) selbst programmieren müssen, dann müssen Sie zuerst verstanden haben, dass es eine Datenübertragung zwischen dem Laufwerk und dem Prozessor gibt. Als nächsten müssen Sie verstanden haben, dass Sie niemals auf das Laufwerk, sondern nur auf den Datenstrom zugreifen können, der zwischen Prozessor und Laufwerk existiert. Sie müssen weiterhin beachten, dass die Datenübertragung über den Datenstrom eine gewisse Zeit dauert. Dann müssen Sie verstanden ha-

⁴Mit unvoreingenommen ist hier gemeint, dass Sie sich die Arbeitsweise eines Computers ansehen und nicht die Darstellung auf einem Display bzw. die Eingabemöglichkeiten in Form von Maus, Tastatur usw.

ben, dass bei dieser Datenübertragung einiges schief laufen kann, was das ist und was Sie ggf. tun müssen, damit die Datenübertragung trotzdem klappt. Weiterhin müssen Sie sich unter Umständen damit auseinander setzen, welche Codierung bei der Speicherung der Daten verwendet werden muss. In dem Fall müssen Sie die Codierung und Decodierung programmieren. Und es gibt noch eine Vielzahl weiterer Probleme, die bei der Nutzung eines Buses auftreten können. Für viele davon hat die **Nachrichtentechnik** Lösungen entwickelt, aber einige müssen Sie selbst im Rahmen der **Programmierung** lösen. Und das können Sie dann und nur dann, wenn Sie intensiv mit all den Problemen beschäftigen, die z.B. bei der Programmierung eines Taschenrechners irrelevant sind. Aber keine Sorge, wir fangen ganz einfach an. So lange Sie die Inhalte dieses Buches konsequent durcharbeiten und in eigenen Programmen umsetzen, werden Sie all das und noch viel mehr im Laufe der Zeit beherrschen.

Umgekehrt machen jedoch auch die Kommunikations- und NachrichtentechnikerInnen häufig den Fehler, die Konzepte und Modellierungen der Informatik als größtenteils untauglich oder überflüssig zu betrachten. Dabei sind diese Ergebnisse der Informatik Lösungen zu genau den Problemen, die bei der Nutzung nachrichtentechnischer Systeme entstanden sind bzw. entstehen. Hier sei wieder einmal auf das große Gebiet der **Praktischen Informatik** verwiesen, dass TechnikerInnen in aller Regel nicht kennen.

Bitte beachten Sie, dass wir hier über Kommunikationstechnik reden; in den **Kommunikationswissenschaften** (Teilbereich der Sozial- und Geisteswissenschaften) geht es um die Kommunikation zwischen Menschen.

Kontrolle Informatik und Nachrichtentechnik sind im Grunde wie zwei Seiten einer Medaille: Die eine kann ohne die andere nicht existieren. Im Regelfall sind beide für die Leistungen der jeweils anderen jedoch blind.

2.4 Von der Nachrichtentechnik zu logischen Gattern

Ein zweiter Bereich, mit dem sich vorrangig die Elektro- bzw. **Nachrichtentechnik** beschäftigt, ist der Aufbau von Komponenten, die logische Operationen ausführen. Diese bestehen seit mehr als fünfzig Jahren aus Kondensatoren. Wenn Sie sich nicht mehr an den Physikunterricht erinnern: Ein Kondensator ist ein Bauteil, über das mittels einer Eingangsleitung gesteuert werden kann, ob Strom weitergeleitet wird oder nicht. Hier sind wir dann auch in einem Bereich, in dem eine 1 gleichbedeutend ist mit fließendem Strom und eine 0 mit nicht fließendem Strom.

Solche logischen Gatter können Sie bei den **FPGAs**, den Field-Programmable Gate Arrays direkt programmieren⁵. Wenn Sie das tun, dann befinden Sie sich in einem der wenigen Gebiete, in dem Elektrotechniker und Informatiker dieselbe Sprache sprechen. Diese Technologie wurde erst Mitte der 80er Jahre entwickelt. Und obwohl wir hier von einer Programmiertechnik reden, werden Sie in Büchern über Programmierparadigmen kein Kapitel finden, dass das entsprechende Paradigma enthält.

Der Grund dafür ist ganz einfach: Diese Art der Programmierung ist eine direkte Umsetzung dessen, was in der Mathematik **boolesche Algebra** genannt wird. Und an dieser Stelle möchte ich eine Lanze für die **Mathematik** brechen: Immer wieder fallen Formulierungen wie „Mathe ist doch nur eine Hilfswissenschaft.“ Das allerdings ist ein Ausdruck, der eher auf die Ignoranz der Person verweist, die ihn gebraucht. Denn tatsächlich gab es die bei Computern verwendete Logik lange vor dem ersten Computer.

Und Mathematik hat zunächst auch nichts mit Rechnen zu tun: Im alten Griechenland gab es eine Disziplin namens Philosophie. Die Philosophen versuchten die Welt und das was in ihr geschieht zu erklären. Daraus entstanden dann weitere Disziplinen wie die Naturwissenschaften, die sich auf bestimmte Teilgebiete der Welt und des Universums konzentrieren. Wenn Sie also bislang über Philosophie als eine Wissenschaft betrachtet haben, in der es darum geht, wie Menschen sich verhalten sollten, dann haben Sie hier eine zu beschränkte Vorstellung.

Es gab aber auch Forscher, die sich fragten, wie ein Universum aussehen würde, wenn sie den umgekehrten Weg wählen würden. Sie untersuchten deshalb, wie ein Universum aussehen würde, für das Eigenschaften willkürlich festgelegt werden. Um das nochmal zu betonen: Im Gegensatz zu allen anderen Wissenschaften setzt sich die Mathematik nicht mit der Beschaffenheit eines einzigen (nämlich unseres) Universums auseinander: Sie geht wesentlich weiter! Sie stellt sich die Frage wie jedes nur denkbare Universum aussehen würde. Und wer so etwas als Hilfswissenschaft abtut, der... Nun ja, wie soll ich es höflich ausdrücken?

Wenn Sie jetzt genau gelesen haben, dann verstehen Sie, warum die Aussage wie „das wurde mathematisch bewiesen“ in den Naturwissenschaften häufig einen falschen Eindruck vermittelt: Sie können mathematisch alles beweisen, wenn Sie nur kreativ genug bei der Entwicklung von Annahmen sind. Ein mathematischer Beweis alleine genügt also nicht, um zu beweisen

⁵Diese Art der Programmierung ist Teil der Veranstaltung Informatik 2 für Media Systems. Das bedeutet leider auch, dass Sie am Studienende keine Kenntnisse der Praktischen Informatik haben werden. Dafür werden Sie im Bereich der Technischen Informatik deutlich mehr Kenntnisse besitzen als Ihre Kommilitonen von anderen HAWs bzw. FHs.

sen, dass etwas wahr ist. Sie müssen außerdem beweisen, dass die Voraussetzungen bzw. Annahmen wahr sind, die Sie für Ihre Beweisführung vorausgesetzt haben.

Ein aktuelles Beispiel sind die „Beweise“ rund um den Urknall. Die Physik ist zurzeit im Stande, zu beweisen, in welchem Zustand sich das Universum wenige Sekunden nach dem sogenannten Urknall befand. Was also direkt zum „Zeitpunkt“ des Urknalls passiert oder was möglicherweise davor „war“ ist zumindest momentan nicht beweisbar. Es gibt jedoch PhysikerInnen, die darauf beharren, das zu können. Sie legen dafür mathematische Beweise vor, die in sich schlüssig sind. Der Fehler liegt hier aber nicht in der mathematischen Beweisführung, sondern darin, dass sie Annahmen treffen, die zumindest noch nicht bewiesen sind. Wie Sie in „**Mathematik 1**“ lernen ist es aber möglich, aus einer falschen Annahme wahre und falsche Schlussfolgerungen zu ziehen. Wenn also PhysikerInnen Annahmen für eine mathematische Beweisführung verwenden, die noch nicht bewiesen sind, dann kommen sie damit zu Aussagen, bei denen unklar ist, ob sie nun wahr oder falsch sind. Also haben die eingangs genannten Forscher nicht etwa den Urknall oder Abläufe rund um den Urknall bewiesen, sondern lediglich gezeigt, welche Abläufe dort stattgefunden haben, wenn bestimmte Annahmen wahr sind.

Jedenfalls sollte Ihnen damit klar sein, dass MathematikerInnen häufig schon Lösungen parat haben, wenn die zugehörigen Probleme noch gar nicht in der Praxis auftreten. Und nein, wir reden hier nicht von Monaten; einige Lösungsansätze, die in den letzten Jahrzehnten zum Einsatz kamen wurden bereits vor mehreren hundert Jahren von Mathematikern konzipiert. Wenn Sie also in einem mathematischen Lehrwerk Formulierungen lesen wie „Es sei ein Universum mit den Eigenschaften...“, dann wissen Sie jetzt, dass das wortwörtlich gemeint ist: Es geht wirklich darum, ein Universum oder einen Teilbereich eines Universums zu beschreiben, das über bestimmte Eigenschaften definiert wird. Dieses Universum hat aber in aller Regel nicht das geringste mit dem zu tun, was Sie sich als ein Universum vorstellen.

Hier auch noch ein Exkurs: Häufig wird von radikalen (Mono-)theisten behauptet, Naturwissenschaftler würde behaupten, dass es Gott nicht gäbe. Das ist falsch: Naturwissenschaftler untersuchen, wie das Universum aussieht. Sie untersuchen dabei, welche Gesetzmäßigkeiten im Universum nachweisbar sind. Das hat aber nichts mit der Frage zu tun, ob es einen Gott gibt, der all das erschaffen hat.

Aber zurück zu den FPGAs: Wenn Sie die boolesche Algebra nicht beherrschen, können Sie auch kein FPGA programmieren. Unabhängig davon

kommt die boolesche Algebra auch immer dann zum Einsatz, wenn Sie den Rechner etwas prüfen lassen wollen. Für Fortgeschrittene gibt es dann zwar noch Mittel wie die Fuzzy Logic, aber das überlassen wir mal lieber den MathematikernInnen und universitären InformatikerInnen.

Kontrolle

Eine erste Art der Programmierung besteht in der Umsetzung der booleschen Algebra, einer mathematischen Methode, um aus einfachen Ja-/Nein-Fragen komplexe Modelle zu entwickeln. Prozessoren basieren auf nichts anderem.

2.5 Weiter zur Maschinensprache

Wenn wir nun einen Prozessor nicht entwickeln, bzw. seine Arbeitsweise programmieren wollen, sondern ihn so nutzen wollen wie er ist, um Programme zu entwickeln, dann ist die erste Methode die **Maschinensprache**. Danach folgt die **maschinennahe Programmierung**.

Wie Sie wissen reden wir von Prozessoren mit einer gewissen **Bittigkeit**. Heute sind die 32-Bit- und die 64-Bit-Prozessoren am bekanntesten. Diese Bittigkeit bedeutet nichts anderes, als dass der Prozessor bei jedem Rechenschritt eine Zahl mit genau dieser Anzahl an Bits addieren kann. Ein 32-Bit-Prozessor kann also bei jedem Rechenschritt eine Zahl zwischen 0 und $2^{32} - 1$ zu einer anderen Zahl in diesem Bereich addieren⁶. Wie Sie in der **Technischen Informatik** erfahren liegt darin auch der Grund, dass 32-Bit-Prozessoren nur einen Speicher mit rund 4 GB verwalten können: Für mehr Speicherbereiche haben Sie schlicht keine „Hausnummern“.

Aufgabe:

Berechnen Sie doch mal eben die maximale Größe für Speicher, die ein 64-Bit-Prozessor nutzen kann. Und suchen Sie nach einem anschaulichen Beispiel, dass diese Zahl verdeutlicht. (Anmerkung, die nötig ist, weil leider einige Studienanfänger das Potenzrechnen nicht beherrschen: 2^{64} ist nicht (!) $2 \cdot 2^{32}$. Die Antwort lautet also nicht „etwas weniger als 8,6 Millionen“.)

Die Maschinensprache besteht dementsprechend aus nichts anderem als Zahlen, die in dem Bereich liegen, der von der Bittigkeit des jeweiligen Prozessors abhängt. Ob eine Zahl nun ein Befehl darstellt, ob Sie als Zahl zu verwenden ist oder ob es eine Position im Speicher sein soll, kann nur wissen wer die Maschinensprache eines Prozessors beherrscht; es gibt keine Möglichkeit, einer Zeile eines Programms in Maschinensprache anzuse-

⁶ $2^{32}-1$ entspricht etwas weniger als 4,3 Millionen

hen, wofür sie steht.

Kontrolle

Maschinensprache ist so schlecht lesbar, dass Sie sie am besten gleich ignorieren. Verwechseln Sie aber bitte nicht die Maschinensprache (siehe dieser Abschnitt) mit der maschinennahen Programmierung (Assembler, siehe nächster Abschnitt).

2.6 Maschinennahe Programmierung – Assembler

Nachdem wir jetzt geklärt hätten, wie man einen Prozessor besser nicht programmiert, kommen wir zur ersten Methode, um einen Prozessor sinnvoll zu programmieren: Die maschinennahe Programmierung, die auch als **Assembler** oder Assembler Programmierung bezeichnet wird. Es ist die erste Variante der **imperativen Programmierung** und leider ist sie alles andere als anschaulich. Aber wenn Sie sich hiermit beschäftigen, dann zu **C** und später zu **C++** oder **Java** weitergehen, werden sie verstehen, was die Vorteile dieser Sprachen sind und werden diese zu schätzen wissen.

Im Regelfall lernen Sie zu Beginn einer Veranstaltung, in der Sie die maschinennahe Programmierung kennen lernen, etwas über den Aufbau des Prozessors, den Sie maschinennah programmieren werden. Da tauchen dann Begriffe wie Pipeline, RISC und CISC und andere auf. Das kann Ihnen helfen, bestimmte Abläufe besser zu verstehen. Für den Einstieg genügt es aber, wenn Sie die Dinge kennen lernen, die Sie tatsächlich programmieren können.

Aber bevor wir auf die eingehen, sollten Sie wieder (wie zu Beginn dieses Buches) die Antwort auf die Frage erfahren, was **maschinennahe Programmierung** eigentlich ist. Und die Antwort hierauf ist genau die, die allgemein mit dem Begriff Programmieren verbunden wird: Sie tippen Zeile für Zeile ein, was der Computer (besser gesagt der Prozessor) tun soll und der führt es dann in genau dieser Reihenfolge aus. Einzige Ausnahme: Sie können Sprünge ins Programm einbauen, die dafür sorgen, dass der Prozessor einen anderen Teil Ihres Programms ausführt, aber am zeilenweisen Ablauf ändert sich ansonsten nichts. Diese anderen Teile werden als **Sub-routinen** bezeichnet. Es sind Programmteile, die an verschiedenen Stellen im Programm ausgeführt werden sollen. Und sie werden einzig und allein deshalb ausgelagert, weil sie dadurch nur einmal programmiert, aber mehrfach ausgeführt werden können. Das reduziert die Fehleranfälligkeit und erleichtert die spätere Verbesserung des Programms.

In der maschinennahen Programmierung besteht nun jede Zeile aus ei-

nem sogenannten **Mnemon** und zwischen keiner und mehreren Zahlen. Ein Mnemon ist einem Hilfswort, das einem Befehl des Prozessors entspricht. Im Gegensatz zur tatsächlichen Maschinensprache können Sie hier also Buchstabenkombinationen für Befehle benutzen. Aber ob eine Zahl eine Zahl oder eine Speicheradresse ist, das müssen Sie zusammen mit der Definition jedes Mnemons lernen.

Dennoch gibt es Gründe, wegen denen bestimmte Entwickler immer noch in Assembler programmieren und aus denen es tatsächlich Sinn macht, sich auf diese Art der Programmierung zu konzentrieren: Es gibt keine effizientere Möglichkeit, einen Computer zu programmieren und viele Prozessoren lassen sich nicht vollständig in anderen Sprachen programmieren. Der Bedarf an Effizienz muss allerdings sehr hoch sein, denn mit C als höherer Sprache können Sie immer noch recht maschinennah und effizient programmieren.

Das Standardwerk für die maschinennahe Programmierung stammt von **Donald E. Knuth** und hat den Titel „**The Art of Computer Programming**“. Es besteht aus sieben Bänden, von denen die ersten drei veröffentlicht wurden. Band vier wurde in zwei Teile unterteilt, von denen der erste ebenfalls veröffentlicht wurde. Die übrigen Bände sind noch in Arbeit. Das mag so klingen, als wenn eigentlich ein anderes Buch als Standard gelten sollte, doch im Gegensatz zu allen mir bekannten Autoren untersucht Knuth in seinem Buch jedes grundlegende Problem, das bei imperativer Programmierung vorkommen kann. Insbesondere wird in seinem Band praktisch ab der ersten Seite klar, warum InformatikerInnen welche mathematischen Grundlagen beherrschen müssen.

Dennoch lassen sich die Bücher auch ohne diese mathematischen Grundlagen durcharbeiten: Knuth hat viele Passagen so verfasst, dass klar erkennbar ist, ob sie auch ohne mathematische Grundlagen verständlich sind. Das gleiche gilt für die Vielzahl an Aufgaben, die den Text begleiten. Die Musterlösungen machen beispielsweise rund ein Drittel des ersten Bandes aus. Damit kann jedeR Interessierte sich unabhängig von den persönlichen Kenntnissen weitgehend in das Themengebiet einarbeiten.

Kontrolle

Auch die maschinennahe Programmierung ist eher abstrakt. Die Zeilen eines solchen Programms bestehen aus Buchstabenkürzeln und Zahlen. Was das Programm bei der Ausführung an welcher Stelle tut ist sehr schwer zu erkennen.

2.7 Zusammenfassung

In diesem Kapitel haben Sie erfahren, dass die Nachrichtentechnik die Techniken und Technologien liefert, mit der InformatikerInnen Ihre Ergebnisse in die Praxis umsetzen können. Ohne Nachrichtentechnik gibt es keine IT-Systeme und letztlich würden Sie nicht nur dieses Buch nicht lesen, Sie würden wahrscheinlich etwas ganz anderes studieren, weil Computer und das Internet nicht existieren würden. Unter Umständen hätten Sie nicht einmal die Hochschulreife erreicht.

Sie haben außerdem einen ersten Einblick in das erhalten, was hinter den Nullen und Einsen steht, und dass das etwas ganz anderes ist, als das, was die meisten sich darunter vorstellen. Sie haben ebenfalls erfahren, dass wir auch die Einsen und Nullen in aller Regel nicht wahrnehmen, weil sie sich hinter den Ergebnissen der Codierung verbergen.

Abschließend haben Sie einen ersten Blick darauf erhascht, wie all das zu Programmiersprachen führt. Im nächsten Kapitel geht es dann um Formen der Programmierung, mit denen Sie in den nächsten Jahren voraussichtlich am häufigsten zu tun haben werden.

Kapitel 3

Vorbereitung fürs Programmieren

Im Grunde brauchen Sie zum Programmieren lediglich einen Texteditor sowie ein SDK bzw. einen Compiler oder einen Interpreter für die jeweilige Sprache. Ein Texteditor wird bei jedem beliebigen Betriebssystem mitgeliefert, während Sie sich SDK/Compiler/Interpreter häufig von der Seite des Entwicklers herunterladen können.

Sie werden im Folgenden des Öfteren die Bezeichnung **Werkzeug** (engl. **tool**) lesen. Beim Programmieren ist damit im Regelfall ein Programm gemeint, das Sie in irgendeiner Form bei der Softwareentwicklung unterstützt.

Da Sie bei der Suche nach Werkzeugen fürs Programmieren über eine Vielzahl an Begriffen stolpern werden, folgt hier eine auszugsweise Aufstellung häufiger Bezeichnungen, sortiert nach Bedeutung:

- Mit **Quellcode** bezeichnet man den Programmtext, den Sie eingegeben haben.
- Ein **Interpreter** ist eine Software, der Ihren Quelltext Zeile für Zeile übersetzt und kontinuierlich jede übersetzte Zeile sofort ausführt.
- Ein **Compiler** ist eine Software, der das von Ihnen verfasste Programm vollständig übersetzt und dabei eine beschränkte Anzahl an Fehlertypen erkennen kann. Personen, die ein sehr beschränktes Verständnis von Programmierung haben sind deshalb häufig überzeugt, dass kompilierte Sprachen sicher, interpretierte Sprachen dagegen unsicher wären. Tatsächlich handelt es sich schlicht um zwei Paradigmen, die unterschiedliche Vor- und Nachteile haben. Und wenn wir von **IT-Sicherheit** sprechen, dann hat das nichts aber auch rein gar nichts mit kompilierten Sprachen zu tun.

- Ein Debugger ist eine Software, die Ihren Quelltext auf Fehler untersucht.
- Ein **SDK** (kurz für Software Development Kit bzw. Softwareentwicklungskit) ist eine Softwaresammlung, die neben einem Interpreter und/oder Compiler eine Reihe weiterer Programme beinhaltet, die Sie bei der Entwicklung von Programmen unterstützt. SDKs werden jeweils für eine bestimmte Sprache entwickelt. Gelegentlich wird auch der Begriff **Toolchain** verwendet. Im hier verwendeten Sinne entspricht eine Toolchain einem SDK.
- Eine **IDE** (kurz für Integrated Development Environment bzw. Integrierte Entwicklungsumgebung) ist eine Sammlung von Programmen, die meist unabhängig von einer bestimmten Sprache zum Programmieren nutzbar sind. Im Gegensatz zu einer SDK beinhalten IDEs in aller Regel einen eigenen Editor, der u.a. Syntaxerkennung bietet. Mehr noch: Viele IDEs sind so entwickelt, dass es möglich ist, eine Vielzahl von SDKs in ihnen zu nutzen und damit eine Vielzahl von Programmiersprachen in Ihnen zu nutzen.

Die nachfolgenden Werkzeuge gehören in den fortgeschrittenen Bereich, mit dem Sie im Rahmen dieses Buches nichts zu tun haben werden. Sie sollten Sie allerdings kennen, damit Sie wissen, wonach Sie später suchen müssen, um Probleme zu vermeiden, die sehr viel Zeit kosten können. Ihre Beherrschung unterscheidet professionelle Software Entwickler von Quereinsteigern.

- **Deployment Support/Tools** sind Werkzeuge, die Sie dabei unterstützen, ein Programm zu verteilen. Stellen Sie sich dazu vor, dass Sie ein Programm entwickeln, das von allen 15.000 Mitarbeitern Ihres Unternehmens genutzt wird. Ohne Deployment Support/Tool müssten Sie nun an jeden Arbeitsplatz gehen, die alte Version deinstallieren, u.U. den Rechner neustarten und dann die neue Version einspielen. Und wenn Sie fertig sind, gehen Sie in Rente. Mit Deployment Support/Tools dagegen lassen Sie lediglich die Änderungen (auch als **Deltas** bezeichnet) automatisiert von einem zentralen Server aus an die Rechner verteilen und dort installieren.
- **Refactoring** bezeichnet eine Überarbeitung einer Software. Hier geht es aber nicht darum, einzelnen Fehler zu beheben, sondern darum, strukturelle Änderungen in eine Software einzuarbeiten. (Hier wären wir bei einem weiteren Aspekt des Software Engineering angelangt.)

Die folgenden Abschnitte enthalten jeweils die Informationen, die sie zur Vorbereitung der Programmierung auf einem Windows-Rechner benötigen, ■

da auf den Rechnern unseres Departments Windows zum Einsatz kommt. Für die Linux- und MacOS-User unter Ihnen gibt es hier leider keine bzw. nur wenige Hinweise. Nutzen Sie bitte über die üblichen Quellen im Netz.

3.1 Assembler und C – Vorbereitung für die maschinennahe und imperative Programmierung

In den Veranstaltungen zur maschinennahen und imperativen Programmierung unseres Departments kommen zurzeit **ARM Cortex-M0 Prozessoren** zum Einsatz. Bei diesen Prozessoren können Sie von der Webpage von IAR ein kostenloses SDK herunterladen. Im Rahmen der Veranstaltung Informatik 3 (für Media Systems) erhalten Sie außerdem Informationen darüber, wie Sie Ihr System einrichten können und wie Sie die dort verwendete IDE beziehen können. Diese stammt von der Fa. Keil, heißt μ Vision und wird auch als **MDK** bezeichnet. Weiterhin benötigen Sie für die Entwicklung von Software für einen ARM Prozessor ein Entwicklerboard, auf dem der Prozessor bereits montiert ist, sowie eine debugging Unit. In beiden Fällen handelt es sich um Hardware, die Sie im Labor für Ihre Versuche erhalten.

Wenn Sie so lange nicht warten wollen und wesentlich weniger Geld ausgeben wollen, um sich in die Programmierung von ARM-Prozessoren einzuarbeiten, dann möchte ich Ihnen einen Computer mit ARM-Prozessor empfehlen, der von der Universität in Cambridge extra für Studienanfänger entwickelt wurde. Es handelt sich um den **Raspberry Pi** (kurz RasPi oder Pi), der in der Version 2 für rund 45,- € zu haben ist. (Maus und Tastatur sowie eine MicroSD-Card als Laufwerk und ein passendes Ladekabel fehlen hier noch, aber für knapp 70,- € sollten Sie ein komplettes System bekommen.) Das praktische am RasPi ist, dass Sie ihn in die Tasche stecken können.

Wenn Sie dagegen die maschinennahe Programmierung auf einem Intel- oder AMD-Prozessor mit IA_x86-Architektur¹ durchführen wollen, können Sie sich die **GCC** (kurz für **GNU Compiler Collection**) kostenlos herunterladen und installieren. Hierbei handelt es sich um eine Sammlung von Compilern für die verschiedensten Sprachen. Im Gegensatz zu vielen anderen Compilern erhalten Sie die GCC unter der sogenannten **GNU GPL** (Lizenz). Kurz gesagt bedeutet das, dass Sie hierauf kostenfreien Zugang haben, dass die Entwickler Ihnen darüber hinaus aber noch wesentlich mehr Rechte einräumen. Die GCC sind in verschiedenen Paketen enthalten, die

¹Das sind die Prozessoren, die zurzeit üblicherweise in Desktop-Rechnern verbaut werden.

eine etwas komfortablere Installation erlauben.

Windows-Nutzern sei hier das **MinGW** empfohlen, das zusätzlich zum GCC auch ein minimalistisches GNU installiert. Mit letzterem können Sie unter Windows auf der Konsole wie in einer Linux-Umgebung arbeiten. Alternativ können Sie auch auf Microsofts **Visual Studio** Professional oder Ultimate zurückgreifen, das Sie als Studierende kostenlos über das Dreamspark Programm erhalten. Davon gibt es noch die Community Edition, die grundsätzlich kostenlos von Microsoft angeboten wird. Beachten Sie aber bitte, dass Sie in diesem Fall unter Umständen Software entwickeln, die ausschließlich auf Windows Rechnern lauffähig ist. Für den Aufruf des GCC-Assembler Compilers müssen Sie abschließend noch die PATH-Variable von Windows um das /bin-Verzeichnis Ihrer MinGW-Installation erweitern.

MacOS-Nutzer brauchen eine derart umfangreiche Software nicht, da Ihr Betriebssystem bereits die entsprechenden Werkzeuge an Bord hat. Allerdings müssen Sie immer noch einen Assembler Compiler installieren, wozu Sie am besten ebenfalls auf die GCC zurückgreifen. Einsteiger werden zuvor noch **XCode** installieren müssen. Dabei handelt es sich um eine IDE, die Sie über den App Store herunterladen können. Suchen Sie anschließend im Netz nach einer Anleitung, wie Sie die GCC in XCode einbinden können. Unter MAC OS X wählen Sie 2010 im XCode

Menü -> Preferences -> Downloads den Eintrag „Command Line Tools“ und hatten nach dem Abschluss des Downloads die GCC vollständig installiert. Die Installation sollte also auch Einsteigern problemlos möglich sein. Genau wie bei Microsofts Visual Studio sollten Sie allerdings daran denken, dass Sie bei der Nutzung von XCode unter Umständen Software entwickeln, die ausschließlich auf Apple-Rechnern lauffähig ist.

Warum es hier keine Informationen für **Linux**-User gibt? Weil Linux-User im Regelfall selbständig genug sind, um sich diese Informationen selbst zu beschaffen. Sollten Sie Linux-Neuling sein, empfehle ich Ihnen die Linux Einführung auf der Plattform edX. Dieser Kurs kann wie die meisten Kurse auf edX auch kostenlos belegt werden und wird im Gegensatz zu den meisten Kursen kontinuierlich angeboten; Sie sind hier also nicht an einen bestimmten Zeitraum gebunden. Er wurde von der FSF entwickelt und ist nach anfänglichem Marketing für die FSF eine fundamentale und sehr nützliche Einführung in das offene und freie Betriebssystem. Leider kommt GNU ein wenig zu kurz, aber Sie erhalten zumindest die entsprechenden Links.

Wenn Sie einen RasPi erworben haben, dann gibt es eine Einführung in Linux im sogenannten **Raspberry Pi Educational Manual**

http://vx2-downloads.raspberrypi.org/Raspberry_Pi_Education_Manual.pdf, das zwar für die erste Version des RasPi entwickelt wurde, aber auch bei der aktuellen Version genutzt werden kann, da diese weitgehend abwärtskompatibel ist. Darin ist neben der Einführung in die Grundlagen verschiedener Arten der Programmierung auch eine Einführung in die Administration von Linux enthalten. Die sollten Sie als RasPi-Nutzer auf jeden Fall durcharbeiten.

Außerdem sollten Sie noch ein gutes Lehr- und Nachschlagewerk für die Assemblerprogrammierung erwerben. Hier gibt es allerdings auch einige Bände, die Sie legal kostenlos herunterladen dürfen. Grundsätzlich möchte ich Ihnen für diese Fälle die folgende Seite ans Herz legen:

<http://hackershelf.com/topics/>

Suchen Sie hier bitte unter dem Suchbegriff **assembly**, nicht **assembler**, da im Englischen die Bezeichnung nicht **Assembler Language** sondern **Assembly Language** lautet.

Persönlicher Tipp: Greifen Sie dort zu „**Programming from the Ground up**“ von Jonathan Bartlet.

Sie wollen wissen, was mit GNU gemeint ist? Gute Frage. Aber die Antwort lautet: Lesen Sie es selbst nach: <https://www.gnu.org/> Es geht hier letztlich um eine der wichtigsten Institutionen, die Sie als angehende InformatikerInnen kennen sollten: Die **FSF** (kurz für Free Software Foundation). Ob nun die **GI** (kurz für Gesellschaft für Informatik), **IEEE**, die **FSF** oder **ITU-T** die absolute Spitzenposition bezüglich der Bedeutung für InformatikerInnen einnimmt, kann ich Ihnen nicht sagen, aber sie sind allesamt außerordentlich wichtig. Beschäftigen Sie sich deshalb damit.

Ergänzungen zu C und C++

Leider befand sich zum Zeitpunkt, als diese Zeilen geschrieben wurden noch kein für Einsteiger empfehlenswertes Buch auf hackershelf.com. Allerdings gibt es für die Programmierung in C ein Standardwerk, das u.a. vom Entwickler der Sprache verfasst wurde: „**The C Programming Language**“ von **Ritchie** und **Kernighan**.

3.2 Pascal - Eine weitere imperative Sprache

Um in Pascal zu programmieren installieren Sie bitte den „Free Pascal Compiler“ sowie die IDE „Geay“.

3.3 Scheme - Eine funktionale Programmiersprache

In einführenden Kursen wird Scheme regelmäßig als interpretierte Sprache vermittelt. Ein bekannter kostenloser Interpreter ist „Petite Chez Scheme“. (Bitte nicht mit Chez Scheme verwechseln; dabei handelt es sich um eine kostenpflichtige Version, die neben dem Interpreter einen Compiler beinhaltet.)

3.4 PROLOG - Logische Programmierung

Im Gegensatz zu den bisher aufgeführten Sprachen ist PROLOG eine interpretierte Sprache. Ein kostenloser Interpreter ist „SWI-PROLOG“, den Sie bitte ebenfalls installieren.

3.5 Java - Vorbereitung für die Programmierung in Java

Wichtig für Wiederholer: Stellen Sie sicher, dass Sie auf jedem Rechner die gleiche Version von Java installiert haben. Zwar ist Java weitgehend abwärtskompatibel, aber die Entwickler der Sprache erklären immer wieder einzelne Bestandteile für veraltet. Wenn Sie dann noch eine alte JDK verwenden, bedeutet das, dass Sie ggf. aktuellen Code nicht programmieren können und veraltete Vorgehensweisen vom Compiler akzeptiert werden. Gerade bei Hausaufgaben führt das dann schnell zu Problemen.

Für Java gibt es zunächst zwei Anbieter von SDKs. Zum einen können Sie eines der offiziellen SDKs von Oracle herunterladen, zum anderen gibt es das OpenJava, das unter einer anderen Lizenz entwickelt wird.

Auf der Webpage des Entwicklers Oracle gibt es Java2SE, Java2EE, Java for Mobile und noch ein gutes Dutzend weiterer Java Downloads. Das was Sie für den Einstieg brauchen ist **Java2SE**. Zum Grund dafür kommen wir am Ende dieses Abschnitts, für die Installation brauchen Sie es nicht zu wissen.

Nachdem Sie sich mit den Nutzungsbedingungen einverstanden erklärt haben, können Sie auf eine Reihe an Downloads zugreifen, die sich dadurch unterscheiden, ob Sie nun einen 32- oder 64-Bit-Prozessor haben, welches Betriebssystem Sie nutzen und ob Sie die Installationsdatei vollständig (offline) oder nur zum geringen Teil (für eine online-Installation) herunterladen wollen. Wenn Sie sich die Zeit nehmen, genau hinzusehen, dann sollten Sie hier die für Ihr System optimale Version wählen können. Im Grunde ist es bei Windows, Linux oder MacOS aber egal, welche Versi-

on (32/64 Bit bzw. online/offline) Sie wählen, so lange das Betriebssystem und der Prozessor stimmen.

Einen Buchtipp für die Programmierung in Java kann an dieser Stelle nicht gegeben werden, was auch daran liegt, dass mit jeder neuen Version wieder essentielle Änderungen in die Sprache eingeführt werden. Dazu kommt, dass Java durch seinen Middleware-Charakter ohnehin derart umfangreich ist, dass kein Buch existieren kann, das auch nur größtenteils vollständig ist. Dazu ist das Framework schlicht zu umfangreich. Sie glauben mir nicht? Dann werfen Sie einen Blick in die Java API; das ist die vollständige Java Dokumentation: <http://docs.oracle.com/javase/8/docs/api/> Und dort sind lediglich all die Klassen enthalten, die von Oracle selbst angeboten werden. Am besten setzen Sie in Ihrem Browser auch gleich ein Lesezeichen auf diese Seite, denn beim Lernen von und Arbeiten mit Java werden Sie wohl nichts so oft brauchen wie diese Seite. Sollte inzwischen die nächste Version von Java veröffentlicht worden sein, dann ändern Sie bitte die entsprechende Ziffer im Link ab.

Bei dieser Gelegenheit lernen Sie auch schon die nächste wichtige Abkürzung kennen: **API** steht kurz für **Application Programming Interface**, was als **Programmierschnittstelle** übersetzt wird. Eine API galt als ein typisches Kennzeichen objektorientierter Programmiersprachen, doch das ist falsch: Eine Vielzahl an Lösungen wurden bereits von anderen Entwicklern programmiert und intensiv getestet, sodass Sie sich darauf konzentrieren können, die Spezialfälle zu programmieren, die bei Ihrer Software auftreten. Tatsächlich finden Sie so etwas jedoch selbst für Assembler. Der Begriff **Klassenbibliothek** wird des Öfteren als Synonym für eine API verwendet. An sich steht dieser Begriff aber allgemeiner für Sammlungen von Klassen, die Lösungen für häufig auftretende Probleme beinhalten.

Da der Begriff der **Klasse** untrennbar mit Java verbunden ist, hier eine kurze Erklärung, die zwar viel zu simpel ist, aber für den Anfang genügen soll: In Java bestehen Programme wie in alle objektorientierten Programmiersprachen aus sogenannten Modulen, die in Java als **Packages** und Klassen bezeichnet werden. Jedes dieser Module beinhaltet einen Programmteil, der für sich genommen bereits ein einen sinnvollen Teil des Gesamtprogramms erfüllen kann. Um bei besonders großen Projekten mehr Übersicht zu erhalten werden dort mehrere Klassen in einem Package gesammelt. Packages können Sie sich in Java wie Verzeichnisse bei einem Betriebssystem vorstellen. Genau wie dort kann ein Package mehrere Packages enthalten, die sowohl Klassen als auch Packages enthalten können.

Anschließend müssen Sie u.U. unter Windows (ähnlich wie bei der Vorbereitung für die maschinennahe Programmierung) noch die PATH-Variable

erweitern. In diesem Fall müssen Sie dort das Verzeichnis angeben, in dem u.a. die Datei `javac` liegt. Prüfen Sie anschließend über den Befehl `javac -v` in der Eingabeaufforderung, ob Sie die `PATH`-Variable richtig gesetzt haben. Anmerkung für einen späteren Umstieg auf Linux: Dort heißt die „Eingabeaufforderung“ **Terminal**, **Konsole** oder **shell**, wird aber auch als `bash`, `sh`, `z-sh` und ähnliches bezeichnet, wobei das teilweise konkrete Programmnamen sind, die ein Terminal bzw. eine Konsole starten.

Jetzt aber zur Antwort auf die Frage, was denn die verschiedenen Java-Versionen unterscheidet. Hier werden wir nicht zu sehr ins Detail gehen, daher nur so viel:

- Warum steht da eine 2 in Java2SE?
Obwohl mit jeder neuen Version (Java ist im Moment, in dem diese Zeilen geschrieben werden bei Version 8 angelangt) essentielle Änderungen an der Sprache durchgeführt werden, entschieden sich die Entwickler nach Version 2 die Zahl 2 im Namen zu behalten. Das ist alles.
- Wir brauchen ja das SDK. Aber was ist die JRE?
Sie wissen bereits, dass Sie ein Softwareentwicklungskit zum Entwickeln von Software nutzen können. Dieses beinhaltet eine Laufzeitumgebung (in diesem Fall die JRE, kurz für Java Runtime Environment), die dazu dient, Java Programme auf einem Computer laufen zu lassen, auf dem kein JDK installiert ist.
Laufzeitumgebungen gibt es für eine Vielzahl von Programmiersprachen. Es handelt sich dabei um so etwas wie eine Übersetzungssoftware, die es dem Betriebssystem ermöglicht, Programme in der jeweiligen Sprache auszuführen.
Und richtig verstanden: Kein Programm wird direkt von einem Betriebssystem ausgeführt, sondern das Betriebssystem startet gegebenenfalls die Laufzeitumgebung für eine Sprache, in der das zu startende Programm dann ausgeführt wird. Ein häufiger Irrtum von Nutzern besteht dagegen darin, anzunehmen, dass beispielsweise unter Windows Dateien mit der Endung `.exe` von Windows selbst ausgeführt werden; vielmehr gilt auf Dateien im `.exe`-Format das gleiche, was für alle Programme gilt, die ausgeführt werden sollen: Es muss auf dem Rechner eine Laufzeitumgebung (oder ein Interpreter) für die entsprechende Sprache installiert sein, sonst kann das Programm nicht ausgeführt werden. Einzige Ausnahme: Ein Programm liegt in Maschinensprache vor. Dann und nur dann kann es direkt vom Prozessor ausgeführt werden.
- Was bedeutet dieses SE in Java2SE? Was ist dieses Java2EE? Und was

sollen all die anderen Versionen?

- **Java SE** (kurz für Standard Edition) ist gewissermaßen der Kern der Sprache/der Middleware. Es enthält alles, was Sie benötigen, um Java Programme zu entwickeln bzw. um sie auf einem Rechner laufen zu lassen.
- **Java EE** (kurz für Enterprise Edition) ist eine erweiterte Fassung von Java SE, die in Unternehmen eingesetzt werden kann, bei denen es wenigstens einen zentralen Server gibt, über den die Kommunikation von Java Anwendungen koordiniert wird.
- **Java ME** (kurz für Micro Edition) ist dagegen eine höchst effiziente Version, die auf Geräten mit geringen Kapazitäten zum Einsatz kommen kann. (Alte Rechner, Smartphones, usw.)
- **Java FX** (kurz für Effects) beinhaltet Klassen, mittels derer Internetapplikationen entwickelt werden können.

Nachdem Sie das JDK heruntergeladen und installiert haben, brauchen Sie noch eine Möglichkeit, um Java Programme einzugeben oder zu ändern, denn das JDK enthält keinen eigenen Editor. Hier genügt für die ersten Schritte ein **einfacher Texteditor**. Verwechseln Sie das bitte nicht mit einer **Textverarbeitung**: Ein Texteditor zeigt Ihnen den eingegebenen Text standardmäßig ohne irgendwelche Hervorhebungen wie Fettdruck und ähnliches. Im Gegensatz dazu zeigt Ihnen eine Textverarbeitung Texte mit Hervorhebungen, bei denen dann der Teil des Quelltexts ausgeblendet wird, über den diese Hervorhebungen erzeugt werden.

Standardmäßig hat jedes Betriebssystem mindestens einen Texteditor an Bord. Bei Windows finden Sie diesen unter Zubehör bzw. indem Sie bei Windows 8 in der „Kachelansicht“ das Wort Editor eintippen. Bei älteren Windows-Versionen können Sie das Suchfenster im Windowsmenü nutzen.

Als ersten Schritt zu mehr Komfort gibt es Texteditoren mit **Syntaxerkennung**. Beispielsweise den Notepad++. Solche Texteditoren heben automatisch syntaktische Fehler hervor, indem sie z.B. eine rote Linie unterhalb fehlerhaften Stellen anzeigen. Sie ändern dabei aber im Gegensatz zu Textverarbeitungen nichts am Quellcode.

Fortgeschrittene werden Ihnen jetzt empfehlen, doch sofort zu einer IDE wie **Eclipse** zu greifen, da die doch so viel besser zum Programmieren seien. Doch bitte lassen Sie das vorerst: Wie schon zuvor geschrieben haben selbst kostenlose IDEs inzwischen einen derartigen Umfang an Komfortfunktionen, dass sie für Einsteiger eher verwirrend als hilfreich sind. Es ist

aber richtig: Nach spätestens sechs Monaten sollten Sie auf jeden Fall damit beginnen, eine IDE zu nutzen.

3.6 HTML, PHP und MySQL – Vorbereitung für die Entwicklung verteilter Anwendungen

Wenn Sie eine Webpage entwickeln wollen, dann benötigen Sie dazu folgende Dinge:

1. Einen einfachen Texteditor wie **Notepad++**.
2. Ein Softwarepaket, das Ihnen die nötige Infrastruktur für eine web-basierte Anwendung bietet. Beispiele: **XAMPP** oder **EasyPHP**
3. Eine Software, die es Ihnen ermöglicht, Dateien auf einen Webserver zu übertragen. Ein einfaches kostenloses Programm ist **FileZilla**.

Im Gegensatz zur imperativen Programmierung müssen Sie sich hier nur um wenige Dinge kümmern, obwohl hier tatsächlich wesentlich mehr Installationsschritte zu erledigen sind, als das bei C, C++ oder Java der Fall ist. Denn all die nötigen Schritte übernimmt hier die Installationsroutine von XAMPP bzw. EasyPHP.

Da Sie alle drei Programme im Netz legal kostenlos beziehen können und die Installation keine fortgeschrittenen Kenntnisse erfordert, sei dazu an dieser Stelle nichts weiter gesagt.

Sie sollten allerdings noch wissen, welche Sprachen und Strukturen hier zum Einsatz kommen. Wie Sie zu Beginn des Kapitels über verteilte Anwendungen erfahren werden, werden Sie diese Punkte später in Veranstaltungen wie „Einführung in relationale Datenbanken“ ausführlich kennen lernen.

1. Zunächst benötigen Sie für eine Webanwendung einen Server, der die Daten der Seite vorhält und über das Internet erreichbar ist.
An dieser Stelle empfehle ich Ihnen die Installation von EasyPHP. EasyPHP wird (genau wie XAMPP) Ihren Rechner so vorbereiten, dass der Apache Server auf Ihrem Rechner läuft und als Webserver genutzt werden kann. Aus Sicherheitsgründen sollten Sie diesen Server aber nur so lange laufen lassen, wie Sie ihn zum Testen benötigen. Denn wenn Sie keine fortgeschrittenen Kenntnisse in IT-Sicherheit und Serveradministration haben, werden Sie sonst mit höchster Sicherheit Sicherheitslücken auf Ihrem Rechner einrichten.

2. Dann benötigen Sie eine Sprache, mittels derer Sie die Elemente Ihrer Webanwendung definieren können. Über diese Sprache regeln Sie also, wie die einzelnen Bestandteile der Webpage zu einzelnen Ansichten verbunden werden. Hier greifen wir zu **HTML**, für das Sie im Gegensatz zu C, C++ oder Java keinen Compiler oder Interpreter installieren müssen: Den HTML-Interpreter finden Sie bereits in Form eines Webbrowsers auf praktisch jedem Rechner vor.

HTML hat allerdings einen entscheidenden Nachteil für unsere Zwecke: Die Sprache ist durchgehend statisch. Wenn Sie also Änderungen auf der Seite einführen wollen, die beispielsweise von den Eingaben des Nutzers abhängen, dann können Sie das mit HTML alleine nicht erreichen.

3. Deshalb folgt als nächstes eine imperative Sprache, wobei wir hier **PHP** wählen, das im Gegensatz zu C und ähnlichen dynamisch typisiert ist. Sie werden sich erinnern: Das bedeutet, dass der Interpreter der Sprache vieles für Sie übernimmt, dass Sie also weniger Details selbst programmieren müssen. Dafür müssen Sie aber umso genauer bzw. umso mehr darüber Bescheid wissen, wie der Interpreter genau arbeitet.

PHP wird von vielen Webservern interpretiert. In unserem Fall übernimmt das der Apache Server, der Teil der Easy-PHP- bzw. der XAMPP-Installation ist.

Mittlerweile wird allerdings JavaScript mehr und mehr zu einem ernstzunehmenden Konkurrenten von PHP, weil es wesentlich mehr Möglichkeiten bietet und eine Vielzahl Beschränkungen dort nicht existieren. In HTML5 ist es als Standard für die Programmierung der Funktionalität einer Webanwendung eingestellt, sodass Sie hier zum Testen nicht unbedingt einen Webserver benötigen.

4. Außerdem brauchen wir noch eine Möglichkeit, um Daten langfristig zu speichern. Denn der Apache Server stellt zwar unsere Webpage bereit, HTML realisiert mit CSS die Darstellung und PHP realisiert die dynamische Nutzbarkeit der Anwendung, aber wenn wir nur diese Sprachen nutzen, können wir Nutzereingaben nicht dauerhaft speichern und wieder verwenden.

Gerade wenn wir eine Webanwendung entwickeln, über die wir Verträge abschließen wollen (egal, ob es dabei um ein browserbasiertes Spiel, einen Webshop oder was auch immer geht), genügt das nicht. Und das führt uns direkt zu einer Datenbank. Datenbanken werden über eigene Sprachen angesprochen, die häufig ein SQL vorkommt. SQL steht kurz für **Structured Query Language**, übersetzt sind das

also standardisierte Abfragesprachen. Und hier ist **MySQL** die Sprache, die wir im Kurs verwenden.

Aber auch hier brauchen Sie wieder keinen Interpreter oder Compiler herunterladen; diese Aufgabe übernimmt bereits XAMPP bzw. EasyPHP für Sie.

Trotz der komfortablen Installation z.B. durch EasyPHP müssen Sie sich all diese Komponenten merken, weil Sie bei einer professionellen Entwicklung all diese Komponenten selbst installieren und konfigurieren müssen. Und schließlich geht es im Studium darum, dass Sie sich auf eine professionelle Tätigkeit vorbereiten.

3.7 Textverarbeitung mit LaTeX

Um umfangreiche wissenschaftlichen Texte zu erstellen wird häufig die Textverarbeitung LaTeX verwendet, die aus Sicht der Programmierung eine Markup Language ist. Ein guter Einstieg, um etwas über den Hintergrund zu erfahren und Teil der Community zu werden ist <http://www.latex-project.org/>. Installieren Sie bitte die Software TeXstudio, um mit der Programmierung in LaTeX zu beginnen.

3.8 Alle - Vorbereitung für die Teamarbeit

Wie im ersten Kapitel aufgeführt ist die Arbeit an einer Software heute eine Aufgabe, die in Teams durchgeführt wird. Und hier müssen Sie Werkzeuge nutzen, die über einen Server koordiniert werden, um auch nur ansatzweise effizient zu sein. Wenn sie dagegen die Vorstellung haben, Ihren Teil der Software zu entwickeln, dann eine Kopie davon an Ihre Teamkollegen zu verschicken und sich anschließend die Kopien der Arbeitsergebnisse der Kollegen zu besorgen, dann sollten Sie sich schleunigst von diesem Dilettantentum verabschieden! So arbeiten ausschließlich Personen, die nichts aber auch gar nichts von Softwareprojekten verstehen.

Die aktuell effizienteste Möglichkeit, um ein solches Softwareprojekt über eine SCM zu verwalten heißt **Git**. Und diese Möglichkeit ist nicht nur außerordentlich effizient und praktisch, sondern obendrein noch kostenlos. Deshalb werden wir Sie hier von Anfang an einsetzen, damit Sie am Ende Ihres Studiums nicht einmal ansatzweise auf die Idee verfallen, Änderungen an Softwareprojekten per Kopie zu verteilen.

Im Arbeitsleben werden Sie allerdings häufig auf eine ältere Form des SCM treffen. Die Bezeichnung dafür lautet **SVN** bzw. **Subversion**. Subversion hat deutliche Nachteile gegenüber Git, ist aber immer noch weit verbreitet,

weil viele Nutzer schlicht nicht willens sind, sich in die Änderungen einzuarbeiten, die mit einem solchen Umstieg verbunden sind. Media Systems Studierende sollten sich deshalb in jedem Fall später (z.B. in der Veranstaltung Software Engineering) auch in Subversion einarbeiten.

Wie schon im ersten Kapitel beschrieben besteht der große Vorteil von Git darin, dass Sie auf das Repository keinen Zugriff haben müssen, um an der Software zu arbeiten; Subversion geht im Kern davon aus, dass Sie sich in einem Unternehmensnetzwerk befinden und deshalb ständigen Zugriff auf das Repository haben.

Neben Git gibt es noch eine Webplattform mit dem Namen **GitHub**. Git ist an Software das einzige, das Sie installieren müssen, um ein SCM zu beginnen. Für die Speicherung des Repository können Sie dann einen beliebigen Rechner nutzen, so lange dieser über das Internet oder ein anderes Netzwerk regelmäßig erreichbar ist. GitHub selbst ist ein Service, der Ihnen anbietet, dass Sie dort Ihre Repositories lagern. Ähnlich wie andere Cloud-Dienste gilt auch hier, dass Sie bei einem kostenlosen Account keine privaten Repositories erhalten. U.U. gibt es hier aber für Studierende Sonderangebote.

Wichtig: Sollten Sie GitHub (oder einen ähnlichen Dienst) nutzen und Ihre Repositories öffentlich sein, dann bedeutet das auch, dass Ihre Projekte für jedermann/-frau frei zugänglich sind. Nutzen Sie dagegen Git mit einem Server, für den Sie Zugangsbeschränkungen erlassen können, dann gilt das natürlich nicht (automatisch).

Das praktische bei Git ist, dass Sie gar keinen Server für die Repositories brauchen, um anzufangen: Sobald Sie die Software heruntergeladen (<https://git-scm.com/>) und installiert haben, können Sie jedes beliebige Verzeichnis unter die Versionskontrolle stellen und können damit arbeiten, als gäbe es bereits ein Repository. Sobald Sie den nötigen Web- bzw. Cloudspace haben, können Sie Ihr/e Projekt/e dort in ein Repository einbinden, bzw. aus ihrem/n Projekt/en heraus ein Repository anlegen.

Die Einführung in die Arbeit mit Git auf der Webpage git-scm.com ist sehr gut, weshalb an dieser Stelle keine weiteren Erläuterungen erfolgen. Nur so viel: Arbeiten Sie sich dort ein, damit Sie von Beginn an damit arbeiten können.

3.9 Alle – Nutzung des Netzlaufwerks zur Speicherung eigener Daten

Leider wissen viele Studierende nicht, dass ihnen in der Hochschule ein Netzlaufwerk zur Verfügung steht, auf dem Sie alle Daten speichern können und speichern deshalb Ihre Ergebnisse „auf dem Rechner an dem Sie in der Hochschule sitzen. Dabei werden die Daten aber gerade nicht auf dem Rechner gespeichert, an dem sie sitzen, sondern in ihrem Rechnerprofil. Und jedes Mal, wenn Sie sich in der Hochschule an einem Rechner anmelden wird dieses Profil über das Netzwerk auf den Rechner übertragen, an dem Sie sich anmelden. Wir hatten schon Studierende, die deshalb eine halbe Stunde und länger an ihrem Rechner saßen, bis sie endlich den Desktop sahen. Aber das liegt nicht etwa daran, dass die Rechner oder das Netzwerk langsam wären, sondern einzig und alleine daran, dass diese Studierenden zum Teil 30 und mehr Gigabyte an Daten in Ihrem Rechnerprofil gespeichert hatten. Nutzen Sie deshalb bitte für alle Arbeiten am Rechner Ihr Netzlaufwerk. Denn die Daten, die dort gespeichert werden werden erst dann auf Ihren aktuellen Rechner übertragen, wenn Sie sie benötigen. Und keine Sorge: Die Daten werden nicht über das WLAN übertragen, sind also im Regelfall so schnell auf Ihrem Rechner, als hätten Sie sie auf einem USB-Stick gespeichert.

An dieser Stelle auch ein Hinweis auf die Nutzung des WLANs: Leider verstehen viele Studierende nicht, dass jeder Aufruf einer Webpage eine gewisse Datenmenge über das Netzwerk überträgt, über das sie mit dem Internet verbunden sind. Und wenn nun fünfzig Studierende gleichzeitig ein Dutzend YouTube-Videos starten oder den neuesten Client für World of Warcraft herunterladen, dann bedeutet das eben, dass das Netz bis an die Grenze ausgelastet ist. Das verstehen Sie nicht? Nun, es ist genau wie im Straßenverkehr: Wenn jede/r mit dem Auto zur Arbeit fährt, dann kommt eben keine/r mehr richtig voran. Und dieser Vergleich entspricht genau dem, was beim Surfen im Netz das Problem ist: Die meisten Nutzer verhalten sich, als wenn sie alleine auf der Welt wären und als wenn Ressourcen unbegrenzt vorhanden wären. Wenn es dann einen Engpass gibt, dann heißt es immer: Das Netz ist schlecht, die Stadt soll mehr Straßen bauen, usw. usf. Doch die Ressourcen sind begrenzt. Ja, das gilt auch in den LTE-Netzen, egal was der Verkäufer Ihres Handy-Providers Ihnen versprochen hat.

3.10 Änderungen

- 9. März 2016: Hinzufügen des Abschnitts zur Vorbereitung für die Programmierung der Markup Language/Textverarbeitung LaTeX.

Kapitel 4

Ausgewählte Programmiersprachen

In diesem Kapitel stelle ich Ihnen einige Programmiersprachen vor und erkläre, wo die Unterschiede liegen.

Der Begriff der höheren Programmiersprachen wird heute eigentlich nur noch am Rande verwendet, weil die Abgrenzung zur maschinennahen Programmierung (und genau dafür steht der Begriff) zum Normalfall geworden ist. Sollten Sie also über diesen Begriff stolpern und sich fragen, was denn eine höhere Programmiersprache ist, dann merken Sie sich einfach: Es ist eine Abgrenzung, die für uns weitestgehend irrelevant geworden ist.

4.1 Nach B kam C

Die mangelnde Verständlichkeit von maschinennahen Programmen führte dazu, dass Programmiersprachen entwickelt wurden, die Befehle und Zeilenstrukturen beinhalteten, die leichter lesbar waren.

Die Zeile

```
if (a < b) then print "a ist kleiner als b"
```

dürfte auch von Menschen lesbar sein, die lediglich über grundlegende Englischkenntnisse, aber kaum über Computerkenntnisse verfügen.

Im Gegensatz dazu dürfte die Zeile

```
CMP R6 MSP
```

selbst bei denjenigen unter Ihnen für Stirnrunzeln sorgen, die bereits Projekte in Java oder C++ entwickelt haben. (Hier handelt es sich um eine maschinennahe Programmzeile, die bei einem ARM-Prozessor einen Vergleich zwischen zwei Zahlen durchführt.)

Eine dieser höheren Sprachen wurde von ihrem Entwickler **Dennis Ritchie** schlicht **C** genannt. Es gab vorher unter anderem eine Sprache namens **B**, die als Vorlage für **C** diente. Vor **B** gab es **BCPL**. Und vor **BCPL** gab es **Algol**. Die Informatiker der Anfangszeit waren weniger an Marketing interessiert, weshalb Bezeichnungen wie Java, Ruby, Python usw. erst ab den 90er Jahren üblich wurden. Vorher wurden häufig einzelne Buchstaben oder Abkürzungen wie im Falle der Sprache **PROLOG** genutzt, was schlicht für programmable logic steht.

C ist bis heute eine sehr wichtige Sprache, weil sie dafür entwickelt wurde, um **Betriebssysteme** zu entwickeln und dabei möglichst wenig maschinen-nah programmieren zu müssen. Zusätzlich können Sie mit ihr grundsätzlich jede Form imperativer Programme entwickeln. Für Sprachen, die wie **C** für alle möglichen Zwecke eingesetzt werden können, wird die Bezeichnung **general purpose programming** (kurz **GPP**) verwendet.

Das Buch „**The C programming language**“ von **Kernighan** und **Ritchie** ist eines der ersten Bücher zur Einführung in die Programmierung mit **C**. Es ist eher schwer zu nutzen, aber wenn Sie sich durch diesen Band durchgearbeitet haben, dann beherrschen Sie die Grundlagen der imperativen Softwareentwicklung, die InformatikerInnen beherrschen müssen.

Wenn Sie in einer Statistik nachsehen, wie viele Programmierer **C** nutzen, dann werden Sie feststellen, dass diese einen immer geringeren Anteil aller Programmierer ausmachen. Das hat damit zu tun, dass **C** für die Entwicklung verteilter Anwendungen relativ wenig Unterstützung anbietet. Aber denken Sie deshalb nicht, **C** sei belanglos geworden; es gibt schlicht wesentlich mehr Bereiche, in denen heute programmiert wird, als in den 70er Jahren, in denen **C** entwickelt wurde.

Eine **verteilte Anwendung** ist nichts anderes als ein Programm, das auf mehreren miteinander vernetzten Rechnern aktiv ist. Die Probleme, die dabei durch die Kommunikation zwischen den Rechnern entstehen sind eines der anspruchsvollsten Themen, mit denen Sie sich auseinander setzen können.

Kontrolle

Höhere Programmiersprachen sind Programmiersprachen, die für Menschen leichter lesbar sind, als das bei Assembler der Fall ist. **C** ist hier einer

der wichtigsten Vertreter, auch wenn es insbesondere bei der Entwicklung von verteilten Systemen eher nicht eingesetzt wird.

4.2 C++ : C mit Objektorientierung

Auch wenn höhere Programmiersprachen übersichtlicher und verständlicher als maschinennahe Programmiersprachen sind, ändert das nichts daran, dass irgendwann der Punkt erreicht ist, an dem auch sie nicht genug Übersichtlichkeit bieten. Vielen Informatikern war das bereits in der Frühzeit der Programmierung klar. Seit Mitte der 50er Jahre wurden deshalb immer neue Konzepte erarbeitet, die dann die Basis für verschiedene Sprachen bildeten, die mehr Strukturierungsmöglichkeiten beinhalten, als das bei C der Fall ist.

Für die Zwecke dieser Einführung soll es genügen, wenn Sie wissen, dass C++ der Sprache C entspricht, aber zusätzlich Möglichkeiten zur objektorientierten Programmierung bietet. Wenn nun von **objektorientierter Programmierung** die Rede ist, dann gibt es das Problem, dass es hierfür zwei Interpretationen gibt, die zu gänzlich unterschiedlichen Programmierstilen führen:

4.2.1 Objektorientierung nach Alan Kay

Alan Kay war ein Forscher am MIT, der den Begriff der Objektorientierung mitprägte aber diese Bezeichnung später als einen großen Fehler bezeichnete. Denn was er meinte war eine Programmierung, bei der der Fokus auf dem **Nachrichtenaustausch zwischen virtuellen Objekten** liegt. Wohlgemerkt, der Fokus liegt auf dem Nachrichtenaustausch, nicht auf den Objekten selbst.

Wenn Sie sich jetzt daran erinnern, was das wichtigste bei einem Computer ist (die Datenübertragung zwischen den Komponenten des Rechners), dann verstehen Sie auch, warum dieses Konzept der logische Schluss ist. Wenn Sie dann noch an den Aufbau des Internet denken, dann können Sie sich vorstellen, wie grundsätzlich und vorausschauend dieses Konzept ist. Und nochmal: In diesem Bereich kommen wir nur dann zu sinnvollen und effizienten Programmen, wenn **InformatikerInnen** und **NachrichtentechnikerInnen** zusammen arbeiten.

Aufgabe:

Können Sie jetzt nachvollziehen, warum Kay die Bezeichnung Objektorientierung als großen Fehler bezeichnet hat?

Dieser Begriff suggeriert, dass die virtuellen Objekte das wichtige sind und

lassen naive ProgrammiererInnen die Bedeutung des Nachrichtenaustauschs vergessen. Da wäre der Begriff des **Message Sending** wesentlich passender gewesen. Aber so ist das eben, wenn ein neues Konzept entwickelt wird; da wird eine einprägsame Bezeichnung genutzt und dann gerät alleine dadurch das eigentliche Konzept in Vergessenheit.

Aus diesem Grund sind auch praktisch alle Programmiersprachen, die im Internet zum Einsatz kommen für dieses Einsatzgebiet praktisch nicht geeignet: Für die Probleme beim Nachrichtenaustausch, namentlich zeitliche Verzögerungen und Verluste bieten sie im Regelfall nur beschränkte Lösungsmöglichkeiten, was dementsprechend eher zu mittelmäßigen Programmen führt. Und hier gilt wieder, dass InformatikerInnen und NachrichtentechnikerInnen leider kaum zusammen arbeiten. Täten Sie das, dann würden just die Probleme wesentlich besser gehandhabt werden, die beim programmierten Nachrichtenaustausch auftreten: Die NachrichtentechnikerInnen würden die Probleme bei der Datenübertragung sinnvoll lösen und die InformatikerInnen würden die Probleme bei der Softwareentwicklung sinnvoll lösen.

Eine Sprache, die Message Sending bzw. Objektorientierung nach Kay umsetzt, heißt **Erlang**. Es handelt sich hier um eine Sprache, die mehrere Paradigmen unterstützt. Richtig gelesen: Es gibt Sprachen, die mehrere **Paradigmen** umsetzen. Und tatsächlich ist das bei den meisten Sprachen der Fall. **Java** war beispielsweise bis zur Version 7 eine rein imperative und klassenbasiert objektorientierte Sprache. Seit Version 8 beinhaltet Sie mit der funktionalen Programmierung aber auch ein Konzept der deklarativen Programmierung. Die Version 9 wird kein neues Paradigma einführen, sondern es wird eine massive Restrukturierung geben, die den Speicherbedarf von Java-Programmen deutlich reduzieren wird, die aber auch bei der Programmierung in Java Folgen haben wird.

4.2.2 Objektorientierung nach Lieschen Müller

Damit kommen wir jetzt zu dem, was heute üblicherweise unter Objektorientierung verstanden wird:

Anstelle eines Programms entwickeln wir im Kern lauter kleine Programme, die jeweils einen gewissen Funktionsumfang anbieten und grundsätzlich als **Klassen** bezeichnet werden. Soll eine bestimmte Funktionalität genutzt werden, erhält die Klasse, die sie enthält einen entsprechenden Befehl, was dann als Methodenaufruf bezeichnet wird. Bitte beachten Sie: Ein Methodenaufruf ist mehr als nur ein einfacher Befehl, aber zu den Unterschieden kommen wir bei der Einführung in die imperative Programmierung, wenn wir uns die sogenannten Funktionen ansehen.

Und auch wenn Klassen, Methoden und Methodenaufrufe zentrale Themen der objektorientierten Programmierung sind, hat dieses Verständnis ungefähr so viel mit Objektorientierung zu tun, wie das Verleimen zweier Holzleisten mit dem Tischlerhandwerk: Es gibt noch wesentlich mehr, was Sie verstanden haben müssen, um wirklich zu verstehen, was Objektorientierung ist.

Das ist auch der Grund, warum man mit der Einführung in die Objektorientierung bereits eine vollwertige Vorlesung für ein oder zwei Semester füllen kann.

Kontrolle

Wenn C Programme zu umfangreich werden, kann man auf C++ zurückgreifen, da es den gleichen Umfang an Befehlen und Strukturen bietet, aber mit der Objektorientierung weitere Strukturierungsmöglichkeiten anbietet. Java ist ebenfalls in diesem Sinne eine objektorientierte Sprache. Beachten Sie bitte, dass bei allen dreien Objektorientierung nicht im Sinne von Alan Kay umgesetzt und angewendet wird, auch wenn das durchaus möglich wäre.

Darüber, was das im Detail bedeutet und wozu es gut ist, haben Sie jetzt noch nichts erfahren. Zerbrechen Sie sich da also bitte nicht den Kopf. Es braucht im Regelfall mehrere Jahre, um diese Punkte verinnerlicht und weitgehend verstanden zu haben.

4.3 Java – C++ ohne maschinennähe

C und damit C++ bieten wie beschrieben die Möglichkeit recht nah an der Maschine zu programmieren, auf der ein Programm laufen soll. Das bringt einen großen Nachteil mit sich: Angreifer können durch geschickt entwickelte Programme in laufende Prozesse eingreifen. Außerdem muss ein C bzw. C++ Programm individuell auf jeden Prozessor zugeschnitten werden, auf dem es laufen soll. Bei der Vielzahl an Prozessoren, die heute in mobilen Endgeräten zum Einsatz kommt ist aber genau dieser letzte Punkt ein ernstes Problem: Nicht nur müssen die Entwickler die Software für jeden Prozessor anpassen, sie müssen insbesondere die Details jedes dieser Prozessoren kennen, sonst entwickeln Sie im besten Falle ineffiziente, im schlimmsten Fall leicht angreifbare Programme. Und wer möchte schon, dass die neueste App ein Einfallstor für Viren und Trojaner wird?!

Deshalb wurde u.a. **Java** entwickelt: Zu einem Zeitpunkt, zu dem nicht nur für Frau Merkel das Internet Neuland war (Anfang der 90er Jahre) entwickelte ein Team bei **Sun Microsystems** diese neue Sprache zusammen mit

einem passenden mobilen Endgerät. Um Entwicklern die Umgewöhnung zu erleichtern, wurden viele Konventionen und Regeln in Java so umgesetzt, wie das bereits in C und C++ der Fall war.

Wenn Sie also bislang dachten, **Apple** sei das Unternehmen, das (mit dem iPhone) das erste Smartphone entwickelt hat, dann liegen Sie schlicht falsch. Hier wie in mehreren anderen Fällen hat Apple (genau wie **Microsoft**) ein Konzept, das andere bereits zuvor ausgearbeitet hatten schlicht zum richtigen Zeitpunkt in einem Produkt umgesetzt und es zum Verkauf angeboten, als es ausreichend Menschen gab, die bereit waren, dafür Geld auszugeben. Das gleiche gilt für grafische Nutzeroberflächen und die Bedienung eines Computers mit der Mouse. Die wurden ebenfalls nicht von Apple entwickelt, sondern von einem Unternehmen namens Xerox Parc. Allerdings kam dort (im Gegensatz zu Steve Jobs, der das Gelände besuchte) niemand auf die Idee, dass mit so etwas Geld verdient werden könnte.

Übrigens ist auch die Möglichkeit, ein Javaprogramm unverändert auf unterschiedlichen Systemen zu nutzen ein Kriterium der Objektorientierung. Dabei spricht man von **Portabilität**.

Insbesondere bei Entwicklern, die vorrangig in C oder C++ aber auch in anderen imperativen Sprachen entwickeln, herrscht bis heute das Vorurteil vor, Java sei eine viel zu langsame Sprache und deshalb überflüssig, ja generell sei **Objektorientierung** unsinnig.

Hier sollten Sie sich merken, dass es im Regelfall keine unsinnigen Sprachen gibt; **Sprachen werden entwickelt, um einen bestimmten Zweck zu erfüllen**. Ist dieser Zweck tatsächlich nützlich und ist die Sprache sinnvoll und für den Zweck effizient konzipiert, dann wird sie im Regelfall einige Jahrzehnte verwendet. Wer dann einer solchen Sprache die Sinnhaftigkeit abspricht zeigt damit lediglich, dass er den Zweck nicht versteht. Und natürlich kann Java nicht die Geschwindigkeit einer Sprache wie C++ erreichen: Java übernimmt die Arbeit, jedes Programm auf einer möglichst großen Anzahl von Rechnern und Smartphones laufen zu lassen. Das bedeutet einen teilweise höheren Aufwand und damit laufen diese Programme in Java langsamer als in C++, wenn es fähige C++-ProgrammiererInnen in C++ umsetzen. Andererseits müssen diese eben auch sehr fähig sein und umfangreiche Kenntnisse über die Unterschiede zwischen rund dreißig Betriebssystemen und Prozessoren kennen und sich kontinuierlich in neue Systeme einarbeiten. Da das kaum jemand leisten kann, der als Entwickler bezahlbar ist, werden die meisten Spiele nur für ein System entwickelt oder sie sind nicht gut auf die einzelnen Systeme angepasst.

Einige von Ihnen werden jetzt einwenden, dass es doch von **Steam** eine

Plattform gibt, auf der Spiele unabhängig vom System laufen. Hier gilt das gleiche, was schon bei Java gilt: So lange diese Spiele nicht individuell für jede Plattform entwickelt werden, kann auch nicht die volle Palette an Möglichkeiten genutzt werden, die das System bietet. Also werden einige Spiele auf dieser Plattform langsamer laufen als wenn Sie speziell an das System angepasst wären.

Die meisten Spieleentwickler nutzen heute allerdings keine Programmiersprache mehr, sondern sogenannten **Game Engines**. Das sind Softwarepakete, die bereits eine Vielzahl an **Bibliotheken** beinhalten, sodass die Mitglieder von Entwicklerstudios sich nur noch auf den Ablauf des Spiels konzentrieren müssen und ein Team von Designern für die Grafik und den Sound benötigen. Um mit einer Game Engine zu arbeiten brauchen Sie deshalb kein Informatikstudium mehr abschließen. Im Gegenteil: Da diese Softwarepakete genau das übernehmen, was fähige InformatikerInnen tun, gibt es in der Spielebranche nur wenige Stellen für vollwertige InformatikerInnen. Im Gegenteil: Als Absolvent z.B. von Media Systems ist die Nutzung einer Game Engine eigentlich ein Rückschritt: Das System übernimmt nicht nur vieles, was Sie sonst umsetzen müssten, es verhindert auch vieles, das Sie kennen und schätzen gelernt haben.

Dagegen müssen Sie anspruchsvolle Aufgaben als (Medien-)InformatikerIn erfüllen können, um eine Game Engine zu entwickeln oder Ihren Funktionsumfang zu erweitern. In Ihrem Studium können Sie das später ausprobieren: Sie werden eine Game Engine namens **Blender** kennen lernen. Diese wird von den meisten Studierenden abgelehnt, da die Nutzeroberfläche nicht wie die von vielen Game Engines oder Programmpaketen für Computergrafik aussieht. Tatsächlich ist Blender die wahrscheinlich beste Game Engine für (Medien-)InformatikerInnen: Da Sie hier alles und ohne Beschränkung erweitern oder verändern können und da Blender vollständig kostenlos und frei verfügbar ist, können Sie genau das tun, was Sie später als professionelle EntwicklerIn tun müssten. (Zum Vergleich: Wenn Sie keine akademische Lizenz erhalten, dann zahlen Sie für Engines wie Maya 3D mehrere tausend Euro. Doch selbst wenn Sie die Software erhalten, dürfen Sie daran nahezu nichts ändern.)

Kontrolle

Java ist eine imperative und klassenbasierte objektorientierte Programmiersprache, deren Programme leicht auf andere Systeme portiert werden können. Es unterstützt zusätzlich seit Version 8 die funktionale Programmierung und damit ein deklaratives Paradigma.

4.4 Verteilte Anwendungen

Die drei Sprachen, mit denen wir uns bislang beschäftigt haben setzen nicht voraus, dass unser Rechner sich in einem Netzwerk befindet, und dass es möglich ist, Daten mit den anderen Rechnern dieses Netzwerks auszutauschen. Nun wissen Sie aber, dass heute annähernd jedes computerbasierte System (also auch Smartphones) zumindest zeitweilig vernetzt ist. Wie Sie durch die einleitenden Kapitel wissen, wurden Rechner im Regelfall schon immer vernetzt und nur im Heimbereich hatten Nutzer einen Computer ohne Netzzugang. (Hieraus resultiert auch die veraltete Unterteilung in Heimcomputer und PCs.)

Wenn wir nun ein Programm entwickeln wollen, das vernetzte Rechner nutzen soll oder sogar nur bei vernetzten Rechnern einsetzbar sein soll, dann müssen wir die Strukturen, die sich daraus ergeben auch in unseren Programmen integrieren. Ein solches Programm wird übrigens als **verteilte Anwendung** bezeichnet, vor allem wenn es genau genommen aus mehreren individuell agierenden Programmen besteht. Wir müssen dann (siehe Alan Kay und die Objektorientierung) beachten, dass Daten zwischen Rechnern transportiert werden müssen. Und das bedeutet, dass wir eine Absicherung für die Fälle schaffen müssen, in denen Daten nicht das Ziel (also einen anderen Rechner) erreichen oder in denen das Ziel aus irgendwelchen Gründen nicht versteht, was es mit diesen Daten tun soll. Wir müssen insbesondere bei Verbindungen über das Internet auch beachten, dass die Datenübertragung einen Zeitversatz hat, und dass wir keine genaue Zeitabstimmung zwischen den Rechnern realisieren können. Die genauen Ursachen und möglichen Auswirkungen verstehen Sie, wenn Sie Veranstaltungen zum Thema **Netzwerke** und **Nachrichtentechnik** besuchen. Es folgen in Kürze einige Beispiele, um Ihnen einen ersten Eindruck zu vermitteln.

Aus der dafür nötigen Denkweise resultieren auch zwei Begriffe: Server und Client. Die naive Vorstellung lautet hier, dass ein Server ein Rechner im Netz ist, der eine bestimmte Funktionalität anbietet, und dass ein Client ein anderer Rechner im Netz ist, der vom Server eine solche Leistung anfordert. Das ist allerdings nicht richtig; ein **Server** ist lediglich ein Programm, das eine bestimmte Funktion anbietet, und ein **Client** ist ein Programm, das eine Funktion abruft. Sie können also auf einem Rechner verschiedene Server und Clients betreiben, wobei bei Betriebssystemen in aller Regel eine Vielzahl an Servern und Clients aktiv ist. (Hier gibt es noch andere Programmarten über die wir aber erst im Rahmen von Veranstaltungen wie „Betriebssysteme“ reden werden.) Einsteiger, die aus der Apple- oder Microsoftwelt kommen sind häufig bei der Installation von Linux überrascht, dass Sie Mailserver und andere Server installieren können, aber Sie wissen

jetzt, warum das so ist.

Dennoch werden häufig einzelne Rechner als Server oder Client bezeichnet. Das ist insbesondere dann kein Problem, wenn ein solcher Rechner ausschließlich eine entsprechende Funktion im Netz übernimmt. Aber Sie wissen jetzt, dass Sie kein Netz benötigen, wenn Sie ein netzbasiertes Programm entwickeln wollen, weil Sie ja auf einem Rechner sowohl den Server als auch den Client betreiben können. Und ja: Sie können dann einen Datenaustausch zwischen Client und Server auf Ihrem Rechner praktisch genauso durchführen, als wenn beide auf unterschiedlichen Rechnern installiert und über ein Netz verbunden wären. Deshalb können Sie z.B. eine Webanwendung, die später im Internet nutzbar sein soll auf Ihrem Rechner entwickeln und sie dort auch testen, selbst wenn keine Internetverbindung vorhanden ist.

Machen Sie sich in solchen Fällen aber bewusst, dass Sie dann die zentrale Fehlerquelle bei verteilten Anwendungen ausblenden: Da Sie keine Daten über das Netz austauschen, wissen Sie nicht, ob die Anwendung am Ende auch tatsächlich so funktioniert, wie Sie sich das vorstellen: Die Datenübertragung kostet Zeit und diese Zeit ist deutlich höher, wenn die Daten über ein Netzwerk übertragen werden, als wenn Sie innerhalb eines Rechners übertragen werden.

4.4.1 Entwicklung von Webanwendungen – MySQL und PHP versus Ruby on Rails

Bevor wir an dieser Stelle weiter machen, hier ein wichtiger Hinweis: Bis vor wenigen Jahren entwickelten die meisten Softwareentwickler Anwendungen, die auf einem System liefen und die ein Netzwerk nur nutzten, um Nachrichten darüber auszutauschen. **Webanwendungen** haben wie alle **verteilten Anwendungen**, mindestens einen Server- und einen Clientteil, die tatsächlich auf getrennten Rechnern aktiv sind. Die einfachste Form von Webanwendungen kennen Sie wahrscheinlich unter dem Namen Internetseiten. Doch das sind nicht einfach nur Dokumente mit Bildern und Videos, die Sie sich auf Ihren Rechner bzw. Ihr Smartphone herunterladen können, sondern es sind immer öfter komplette Anwendungen. Allerdings werden diese Anwendungen zum Teil auf dem Server und zum Teil auf dem Client ausgeführt. Wenn Sie sich intensiver mit diesem Bereich beschäftigen, werden Sie Sprachen wie **PHP** kennen lernen, die ausschließlich als Server bzw. auf einem Webserver genutzt werden können. Allerdings ist dieser Ansatz veraltet: Aktuelle Sprachen wie **JavaScript** können sowohl als Server als auch als Client eingesetzt werden. Das ist allerdings für Einsteiger bzw. Erstsemester meist nur schwer umsetzbar, da Sie hier bewusst und gut begründet entscheiden müssen, auf welche Programm-

teile Nutzer Zugriff haben dürfen.

Wenn Sie also denken, die Programmierung in **HTML** (der am häufigsten eingesetzten Sprache für Webanwendungen) sei langweilig und man könnte damit nur einfache Internetseiten programmieren, dann liegen Sie falsch; das war in der Version 4.01 so, die Ende 1999 veröffentlicht wurde. Mit der Version 5, die im Herbst 2014 veröffentlicht wurde, ist dieses Thema endgültig passé: Basierend auf HTML 5 ergänzt um Sprachen wie PHP oder JavaScript können Sie Anwendungen entwickeln, die genau das gleiche leisten wie eine beliebige Anwendung, die Sie auf einem einzelnen Rechner nutzen können. Der Begriff Webanwendung ist im Grunde ein Synonym für den Begriff der verteilten Anwendung.

Zu Beginn dieses Kapitels haben Sie erfahren, dass im Grunde keine Sprachen existieren, die die zentralen Probleme angehen, die bei der Datenübertragung im Netz aufkommen. Der Grund besteht darin, dass für die meisten InformatikerInnen eben das System im Mittelpunkt steht, auf dem eine Software ausgeführt wird. Die Kommunikationswege dazwischen und der Zeitfaktor bei der Übertragung werden im Grunde immer nur als lästiges Übel angesehen oder gleich gänzlich ignoriert. Das gleiche gilt für die Nutzung von Webanwendungen durch Menschen.

Ein anschauliches Beispiel konnten Sie bei ebay in der Anfangszeit erleben: Damals hatten die Entwickler ignoriert, dass kurz vor Abschluss einer Auktion besonders viele Aufrufe und Gebote für ein Angebot erfolgten. Also konnten die Server gar nicht alle Angebote „sofort“ verarbeiten. Dementsprechend wurde es zu einer Art Glücksspiel, ein Gebot kurz vor Versteigerungsschluss abzugeben: Unter Umständen wurde Ihr Gebot scheinbar ignoriert, weil es erst nach Auktionsende von den ebay-Servern verarbeitet werden konnte. Dieser Fehler im System wurde inzwischen soweit als möglich bereinigt. Dies ist außerdem ein Beispiel für die Bedeutung des Begriffs **Skalierbarkeit**.

Damit wieder zurück zu den eingangs genannten Sprachen: **Ruby on Rails** ist nicht die Sprache selbst, sondern ein Framework namens Rails, das Ruby so erweitert, dass Sie damit **Webanwendungen** entwickeln können.

Eine zweite Möglichkeit (und deutlich älter), um Webanwendungen zu entwickeln besteht in der Kombination aus drei Sprachen: **MySQL** ist eine Sprache, mit der Sie Datenbanken nutzen können und **PHP** ist eine imperative Sprache, mit der Sie die Funktionalität von Elementen einer Webanwendung programmieren können. Dazu kommt noch **HTML**, was eine **Markup Language** ist. Markup Languages sind Programmiersprachen mittels derer sich die Struktur von Anwendungen unabhängig von der Dar-

stellung und der Funktion programmieren lassen. HTML ist eine Markup Language mit der sich die Struktur einer Webanwendung und seit Version 5 die Bedeutung der Inhalte programmieren lässt.

Viele Softwareentwickler reden in Bezug auf Markup Languages vom sogenannten **Scripten**. Für diesen Begriff gibt es keine präzise Definition. Wenn ProgrammiererInnen ihn benutzen, dann geht es in aller Regel um etwas, das zwar programmiert werden muss, damit eine bestimmte Aufgabe erfüllt wird, das aber vom jeweiligen Entwickler keinerlei logisches Denkvermögen erfordert. Sie müssen im Falle vom Scripten also nur verschiedene Befehle aneinander reihen oder ineinander verschachteln, ohne sich weiter Gedanken darüber zu machen, wie die miteinander interagieren: Sie tun es schlicht nicht. Teilweise wird auch bei Konfigurationsdateien vom scripten gesprochen, obwohl hier (im Gegensatz zu HTML4.01 oder \LaTeX) sehr viel Grundlagenwissen nötig ist. Wenn Sie beispielsweise nicht genau wissen, was der Unterschied zwischen SSH und SSL ist, dann sollten Sie von der Konfiguration eines Servers die Finger lassen.

Haben Sie im ersten Semester eine Veranstaltung zum Webpage Development besucht, dann können Sie eine Webpage entwickeln, denn das ist gar nicht so schwer. Aber viele Konzepte und Abläufe werden Ihnen kaum klar werden. Wenn Sie allerdings die nötige Zeit investieren, dann werden Sie sich basierend auf dieser Veranstaltung die Grundlagen erarbeiten können, um eine vollwertige Webanwendung zu entwickeln.

Ein Tipp für den Fall, dass Ihnen jemand zu **Ruby on Rails** rät: Ja, es ist richtig, dass Rails ein großartiges Framework ist, mit dem Sie selbst komplexe Anwendungen für multinationale Konzerne entwickeln können. Aber Sie merken es schon an der Formulierung: Es ist für Einsteiger schlicht zu komplex und die Vielzahl an Optionen, mit denen Sie von Beginn an konfrontiert werden, lenkt Sie von den Punkten ab, die Sie als Einsteiger verinnerlicht haben müssen. Hier würde ich eher empfehlen, dass Sie sich in **HTML5** und **JavaScript** einarbeiten. Leider sind aber die meisten Anleitungen im Netz (selbst wenn dort die Rede von HTML 5 ist) immer noch Einführungen in HTML 4, bei denen praktisch alles ignoriert wird, was an Version 5 so großartig ist. Teilweise werden hier sogar Techniken vermittelt, die bereits in der Version 4 als schlampig galten. Der Grund ist recht simpel: Wie so oft erklären dort Menschen die Programmierung, die zwar die alte Version beherrschen, aber die schlicht zu faul oder dumm sind, um zu erkennen, dass Version 5 keine kleine Erweiterung um ein paar nette Effekte ist, sondern eine vollständige Überarbeitung, bei der Aspekte berücksichtigt wurden, die nirgends in Version 4 auftauchen und auch nichts mit dem zu tun haben, was in Version 4 vorhanden ist.

Sie fragen, was eine Datenbank ist? Wie so oft, wenn **InformatikerInnen** es mit gleichartigen Daten oder Abläufen zu tun haben, entwickeln Sie entsprechende Strukturen, die letztlich dazu dienen, Fehler zu reduzieren und Abläufe effizienter zu gestalten. **Datenbanken** sind eine weitere Lösung, die so entstanden ist: Wann immer es um große Mengen gleichartiger oder gleichartig strukturierter Daten geht, die nach verschiedenen Kriterien untersucht oder geändert werden müssen, wird eine Datenbank verwendet. **Relationale Datenbanken** bestehen dabei aus Tabellen, bei denen jede Spalte einem Kriterium entspricht und jede Zeile einem sogenannten **Datensatz**. Beispielsweise würde bei einer relationalen Kundendatenbank jede Zeile einem Kunden entsprechen und Einträge in den Spalten wären nach Aspekten wie Name, Vorname, Anschrift, usw. unterteilt.

Bei Webanwendungen dienen Datenbanken (wie bei allen Anwendungen) verschiedenen Zwecken. Zum einen wäre da die klassische Kundendatenbank. Dann gibt es Datenbanken, in denen Einträge auf den Webanwendungen verwaltet werden. Auch die Speicherung jedes Klicks und jeder Taste, die NutzerInnen gedrückt haben wird mit einer Datenbank realisiert. Aber es gibt noch wesentlich mehr Einsatzmöglichkeiten. Wie oben genannt geht es schlicht darum, große Mengen gleichartiger Daten in einer strukturierten Form aufzubewahren, um möglichst schnell darauf zuzugreifen.

Kontrolle

Beginnen Sie beim Webapplication Development mit einer Einführung in MySQL und PHP sowie HTML5. Wenn Sie die objektorientierte und funktionale Programmierung beherrschen, dann können Sie auch JavaScript anstelle von PHP verwenden. Allerdings ist hier ein häufiger Fehler, dass dann HTML nur noch dazu genutzt wird, die JavaScript-Anwendung zu starten, sodass sie in einem beliebigen Browser genutzt werden kann. Ruby on Rails ist ein sehr mächtiges Werkzeug aber für Einsteiger nur beschränkt empfehlenswert. Dazu kommt, dass Rails häufig in Kombination mit weiteren Frameworks verwendet wird, was den Einstieg zusätzlich erschwert. Außerdem ist es leider noch immer nicht so effizient wie JavaScript.

4.5 Konzepte bei der Programmierung

Wie schon mehrfach angeführt gab und gibt es zu jeder Zeit eine Vielzahl von Programmiersprachen, die jeweils für bestimmte Zwecke ausgelegt sind. Zum Teil sind die Unterschiede nur in wenigen Details begründet. Um Ihnen einen kleinen Überblick darüber zu verschaffen, was es noch für Sprachen gibt und wofür diese nützlich sind, folgt eine kleine und dementsprechend unvollständige Aufstellung. Zum Teil werden hier weitere Be-

griffe eingeführt, die für die Programmierung insgesamt wichtig sind.

4.5.1 Dynamisch versus statisch – Ruby und Python versus C und Java

Ruby und **Python** sind Programmiersprachen, die in Konkurrenz zu Java stehen. Der auffälligste Unterschied besteht darin, dass Ruby **schwach typisiert** ist. (Alternativ spricht man auch von **dynamischer Typisierung**.) Im Gegensatz dazu sind **C** und **Java** **stark bzw. statisch typisiert**. Beide Varianten haben Vor- und Nachteile. Und leider neigen die meisten Entwickler dazu, die Variante als schlecht zu bezeichnen, die sie als zweites kennen lernen. Wer das tut hat aber leider nichts mit professionellen **InformatikerInnen** zu tun, selbst wenn er/sie in einer Sprache bzw. einem **Paradigma** wirklich gut ist.

4.5.2 Typisierung von Daten

Bislang haben wir lediglich über Codierung gesprochen aber nicht über Typisierung. Wie Sie bereits wissen werden alle möglichen Daten, die sie vom Computer verarbeiten lassen in einer anderen Form gespeichert als die, in der sie angezeigt werden. Wenn wir nun von statischer oder starker Typisierung sprechen, dann bedeutet das, dass Sie bei einem Wert, den Sie programmieren so etwas Ähnliches wie eine Codierung festlegen. Der Typ eines Wertes, den Sie so vergeben wird entsprechend als **Datentyp** bezeichnet.

Einer der ersten Fälle, in denen Sie mit Typisierung zu tun bekommen ist das Rechnen mit ganzzahligen und ganzrationalen Zahlen. Diese werden nämlich vom Rechner unterschiedlich gespeichert: Fließkommazahlen werden nicht in der Form gespeichert, die Sie aus dem Mathematikunterricht kennen, aber eine Einführung in diese Materie überlasse ich den Kollegen der **Technischen Informatik**. Jetzt aber ein Beispiel für die möglichen Varianten, wie eine Programmiersprache mit ganzen Zahlen umgehen kann: ■

Wenn Sie die Zahl 5 programmieren, als Typ der Zahl ganzzahlig (**Integer**) festlegen und anschließend durch 2 teilen (oder jede andere Zahl ungleich ± 5 oder ± 1), dann ergibt sich bekanntlich eine ganzrationale Zahl. Je nach Programmiersprache gibt es nun unterschiedliche Möglichkeiten, was dabei passiert:

1. Bei statisch typisierten Sprachen erfolgt in aller Regel eine Fehlermeldung, denn Sie haben definiert, dass Ihre Zahl ganzzahlig ist. Also muss das Ergebnis ebenfalls ganzzahlig sein. Es gibt dennoch

Möglichkeiten, um eine solche Aufgabe in einer solchen Sprache lösen zu lassen. Dazu sind jedoch zusätzliche Programmzeilen nötig.

2. Wenn keine Fehlermeldung erfolgt, ist das das Ergebnis häufig nicht das, was Sie erwarten. In den Fällen, wo die Sprache vorsieht, dass das Ergebnis als ganzzahliger Wert gespeichert wird, wird nun entweder auf- oder abgerundet.

ABER! Ob auf- oder abgerundet wird, das hat nichts mit den Rundungsregeln zu tun, die Sie aus der Schule kennen: Es gibt also vier Möglichkeiten, wie eine Programmiersprache damit umgeht, wenn Sie eine Division von zwei ganzzahligen Werten einprogrammieren und bei der keine Ganzzahl berechnet wird.

- (a) In der Sprache wird stets abgerundet:
 $5 : 2 = 2$.
 $-5 : 2 = -3$,
denn $-2,2$ abgerundet ergibt -3 und nicht -2 !
- (b) In der Sprache wird stets aufgerundet:
 $5 : 2 = 3$
 $-5 : 2 = -2$,
denn $-2,2$ aufgerundet ergibt -2 und nicht -3 !
- (c) In der Sprache sind die Rundungsregeln enthalten, die Sie aus dem Mathematikunterricht der Schule kennen. Das ist der seltenste Fall.
- (d) Die Sprache gibt in solchen Fällen eine Fehlermeldung oder eine **Exception** aus. Als EntwicklerIn müssen Sie dann das Programm entsprechend korrigieren.

Wichtig:

Eine Exception ist KEINE Fehlermeldung. Es ist vielmehr ein Komfortfaktor einzelner Programmiersprachen, der sie darauf hinweist, dass Ihr Programm in bestimmten Fehlern nicht so ablaufen wird, wie Sie das wahrscheinlich erwarten. Je nach Komfort der jeweiligen Sprache gibt es auch Exceptions, die auf Situationen hinweisen, die durchaus wie gewünscht verlaufen können, wo Ihnen die Programmiersprache also quasi den Tipp gibt, zu prüfen, ob Sie dieses Verhalten so haben wollen oder ob das nicht doch ein logischer Fehler ist. In Java haben Sie sogar die Möglichkeit, eigene Exceptions zu programmieren, um bestimmte Ausnahmen ganz bewusst anders verarbeiten zu lassen als das sonst der Falle wäre.

3. Bei **dynamisch typisierten Sprachen** wird der Datentyp je nach Bedarf automatisch von der Programmiersprache angepasst. Hier pro-

programmieren wir also in aller Regel nicht den Datentyp. Generell steht der Begriff des **Typecasting** für eine solche Anpassung, die in einigen statisch typisierten Sprachen möglich ist.

Typecasting gibt es noch für andere Datentypen, es ist also nicht nur auf die Umwandlung des Datentyps bei einer Zahl beschränkt, sondern bei allen denkbaren virtuellen Objekten.

Aber auch beim Typecasting ist das Ergebnis nicht automatisch das, was Sie denken. Das hat wiederum mit der Speicherung von Daten zu tun. Wie Sie wissen basiert die Speicherung von Daten in einem Computer auf Zahlen der Basis 2. Und damit basiert die Speicherung von Nachkommastellen auf Potenzen von $\frac{1}{2}$. In Informatikveranstaltungen werden Sie dazu einige Beispiele rechnen, hier seien nur zwei genannt: $\frac{5}{2} = 2 + 1/2$. Binär lässt sich das in der Form $[10, 1]_2$ darstellen: $(1 \cdot 2) + (0 \cdot 1) + (1 \cdot \frac{1}{2})$. Versuchen wir einmal, die Zahl 0,3 als Binärzahl darzustellen:

$$\begin{aligned}
 & (0 \cdot 1) + (0 \cdot \frac{1}{2}) + 0,3 \\
 &= (0 \cdot 1) + (0 \cdot \frac{1}{2}) + (1 \cdot \frac{1}{4}) + 0,05 \\
 &= (0 \cdot 1) + (0 \cdot \frac{1}{2}) + (1 \cdot \frac{1}{4}) + (0 \cdot \frac{1}{8}) + 0,05 \\
 &= (0 \cdot 1) + (0 \cdot \frac{1}{2}) + (1 \cdot \frac{1}{4}) + (0 \cdot \frac{1}{8}) + (0 \cdot \frac{1}{16}) + 0,05 \\
 &= (0 \cdot 1) + (0 \cdot \frac{1}{2}) + (1 \cdot \frac{1}{4}) + (0 \cdot \frac{1}{8}) + (0 \cdot \frac{1}{16}) + (1 \cdot \frac{1}{32}) + 0,01875 \\
 &= \dots
 \end{aligned}$$

Aber es gibt kein endgültiges Ergebnis. Dementsprechend kann weder der Computer noch die Programmiersprache eine Zahl wie 0,3 richtig darstellen. Er könnte sie als $3 \cdot 10^{-1}$ darstellen, aber da das nur eine begrenzte Anzahl an Spezialfällen aller möglichen ganzrationalen Zahlen löst, ist das keine Lösung, die wir immer nutzen können.) Also wird eine Zahl wie 0,3 nur annäherungsweise gespeichert. Und wenn sie dann ausgegeben wird, kann so etwas wie 0,3000010002 oder 0,298991 herauskommen. Auch hierfür gibt es Lösungen, die aber auch wieder bedeuten, dass Sie zusätzliche Zeilen programmieren müssen.

Wenn wir mit der Programmierung in C beginnen, werden Sie häufig mit solchen Fällen zu tun haben. Sie werden dann (wie auch sonst grundsätzlich bei der Programmierung) Lösungen entwickeln müssen, damit der Rechner die Zahlen verwendet und speichert, die für Ihre Aufgabenstellung eine richtige Lösung darstellen.

Dies ist allerdings kein Beispiel dafür, was **InformatikerInnen** von ProgrammiererInnen unterscheidet: Gute ProgrammiererInnen wissen, dass es solche Probleme gibt, sie kennen die Ursachen und sie entwickeln Programme so, dass diese Probleme in allen Varianten gelöst werden. (Sonst sind es Dilletanten, was leider auf viele Quereinsteiger zutrifft.) Allerdings lernen InformatikerInnen in Ihrem Studium verschiedene systematische Methoden, um solche Probleme effizient anzugehen. Der entsprechende Bereich heißt **Praktische Informatik**, wobei die entsprechenden Veranstaltungen in aller Regel unter Titeln wie **Algorithmen und Datenstrukturen**, **Algorithmendesign** und **Algorithmen** angeboten werden.

Aber zurück zu Ruby: Wie gesagt handelt es sich hier um eine dynamisch typisierte Programmiersprache, während C, C++ und Java statisch typisiert sind. Es spielt keine Rolle, welche der beiden Varianten Ihnen lieber ist, das einzige, was eine Rolle spielt ist, dass Sie langfristig lernen, beide Formen der Typisierung zu beherrschen.

Eine weitere Programmiersprache, die neben Ruby in den letzten Jahren immer bekannter wurde und dynamische Typisierung bietet, ist **Python**.

Kontrolle

Es gibt dynamisch und statisch typisierte Sprachen. Der Unterschied besteht darin, dass bei den dynamisch typisierten Sprachen die Programmiersprache Methoden hat, um den Datentyp eines Wertes automatisch anzupassen. Das ist komfortabel, aber es ist nicht intuitiv. Denn während Sie bei einer statisch typisierten Sprache selbst programmieren müssen, wie ein Typcasting ausgeführt wird, müssen Sie bei jeder dynamisch typisierten Sprache genau wissen, wie diese einzelne Sprache das „automatische“ Typcasting durchführt. Wenn Sie das eine oder das andere nicht beherrschen, dann programmieren Sie den Computer nicht, um das zu tun, was er tun soll, sondern Sie programmieren Ergebnisse, die schlichtweg falsch sind. (Und glauben Sie mir, das wollen Sie nicht bei der Steueranlage eines Flugzeugs...)

4.5.3 Funktionen – Dynamisch versus statisch

Dieser Abschnitt dürfte auch für die meisten fortgeschrittenen unter Ihnen eine Überraschung beinhalten: Nicht nur Datentypen, sondern auch Funktionen können bei einzelnen Sprachen dynamisch während der Laufzeit eines Programms geändert werden.

Wichtig:

Bitte denken Sie jedoch nicht, dass das Programmieren einer Funktion eine Form der **funktionalen Programmierung** ist: Genau wie bei der **impera-**

tiven Programmierung nutzen Sie dort etwas, das als Funktion bezeichnet wird, um Abläufe aus dem Programm auslagern. Diese Art der Funktionsdefinition sorgt dafür, dass Sie den Inhalt der Funktion an beliebigen Stellen Ihres Programms verwenden können. Bei der funktionalen Programmierung ist eine Funktion dagegen eine Umsetzung des sogenannten **Lambda-Kalküls**, das wir uns erst bei der Einführung in die **deklarative Programmierung** ansehen werden.

Doch für die Einsteiger zunächst die Erklärung, was eine Funktion ist: Eine **Funktion** fasst in der Programmierung mehrere Programmzeilen zusammen und lässt sich über einen Bezeichner von beliebigen Stellen eines Programms aus aufgerufen werden.

Die meisten Programmierer kennen Funktionen dagegen nur als ein Mittel der Programmierung, das sich nicht ändern kann, während das Programm läuft. Das ist aber ein Irrtum, der darauf basiert, dass das bei Programmiersprachen wie C, C++ oder Java so ist.

Tatsächlich werden Funktionen während des Programmlaufs wie alle anderen Daten eines Programms im Speicher des Rechners abgelegt und bei Bedarf von dort geladen. Und weil der Speicher eines Rechners zu beliebigen Zeiten geändert werden kann, ist es natürlich auch grundsätzlich möglich beliebige Teile eines Programms abzuändern, das dort abgelegt wurde.

Kontrolle

Auch wenn die meisten bekannten Sprachen das nicht können, ist es grundsätzlich möglich, dass auch Funktionen eines Programms dynamisch programmiert werden.

4.6 First-class Objects

Die meisten Programmierneulinge lernen das Programmieren mit Variablen kennen und entwickeln dabei die Vorstellung, dass eine Variable nur einen Wert oder eine Menge an Werten (z.B. die sogenannten Arrays) sein kann. Tatsächlich kann aber auch eine Funktion der Wert einer Variablen sein. Ist das bei einer Programmiersprache der Fall, dann können Sie nicht nur eine Funktion mit einem Wert aufrufen, sondern Sie können eine Funktion quasi wie einen beliebigen Wert an eine andere Funktion übergeben. Das bedeutet, dass Sie dann die Möglichkeit haben, den Ablauf einer Funktion während eines Programmlaufs dynamisch anpassen können.

Diejenigen von Ihnen, die bereits imperativ programmiert haben werden

jetzt behaupten, dass das doch klar sei, weil so „schon immer“ der Wert einer Funktion an eine Variable übergeben wurde. Damit zeigen Sie, dass Sie den Absatz missverstanden haben: Dort steht, dass auch eine Funktion als ganzes und eben nicht nur der Wert, den sie berechnet in einer Variablen gespeichert werden kann. Warum das so ist werden wir uns ansehen, wenn wir klären, was genau eine Variable eigentlich ist.

Als Sammelbegriff für alles, was einer Funktion übergeben werden kann wird der Begriff des **first-class Object** verwendet. Wenn also die Rede davon ist, dass in einer Programmiersprache Funktionen first-class Objects sind, dann bedeutet das nichts anderes, als dass Sie in dieser Sprache eine Funktion genauso als Objekt an eine andere Funktion übergeben können, wie Sie das mit einer Variablen gewohnt sind.

Kontrolle

Im Gegensatz zur meist anzutreffenden Überzeugung von Programmierern spricht eigentlich nichts dagegen, auch Funktionen als Argumente an Funktionen zu übergeben. Und wenn eine Programmiersprache das unterstützt, dann reden wir davon, dass in dieser Sprache Funktionen first-class objects sind.

4.7 Zusammenfassung

In diesem Kapitel haben Sie einen Überblick erhalten, wie die drei Sprachen C, C++ und Java zusammen hängen und wo die Unterschiede liegen. Sie haben eine Vielzahl an Begriffen kennen gelernt, die bei der Programmierung von Hochsprachen von Belang sind.

Sie haben verstanden, dass es keine beste Sprache oder sinnlose Sprachen gibt, sondern dass Sprachen jeweils für eine bestimmte Problemstellung entwickelt wurden. Deshalb gibt es dann auch ganz unterschiedliche Arten (Paradigmen) des Programmierens und hier haben Sie konkrete Fälle kennen gelernt, um den Begriff des Paradigmas mit Leben zu füllen.

Sie wissen jetzt, dass Sie es gelegentlich mit einem bestimmten Programmierparadigma zu tun haben und teilweise lediglich mit einem Spezialfall, der so nur in einer einzelnen oder bei einigen wenigen Sprachen umgesetzt wird. Diese Kenntnisse sind ein weiterer Unterschied zwischen einem dilettantischen Quereinsteiger und einem ernstzunehmenden Softwareentwickler.

Danach haben Sie etwas über Konzepte erfahren, die C-, C++- und Java-ProgrammiererInnen nicht verstehen und die beispielsweise in Ruby, Py-

thon und JavaScript zum Einsatz kommen.

Kapitel 5

Grundlagen der Entwicklung verteilter Anwendungen

Verteilte Anwendungen sind Programme, die aus einer Vielzahl von einzelnen und unabhängigen Programmen bestehen, die jeweils auf unterschiedlichen Rechnern als Server bzw. Clients in einem Netzwerk Teilaufgaben erfüllen. Auch wenn Sie unabhängig voneinander agieren, sind Sie doch als Ganzes eine gemeinsame Funktionalität bzw. ein gemeinsames Programm. Erst wenn dieses letzte Kriterium gilt, wird das Ganze als eine verteilte Anwendung bezeichnet.

Wenn Ihnen das zu abstrakt klingt, dann stellen Sie sich ein Team von verschiedenen MitarbeiterInnen vor, die an verschiedenen Standorten für ein Unternehmen tätig sind. Die verschiedenen Server und Clients entsprechen den MitarbeiterInnen. Wenn Sie mit den Begriffen Server und Client Schwierigkeiten haben, dann lesen Sie bitte die entsprechenden Abschnitte der vorhergehenden Kapitel.

Die einfachste Möglichkeit, um eine verteilte Anwendung zu entwickeln besteht in einer Kombination, bei der HTML für die Struktur der Anwendung verwendet wird. Zusätzlich benötigen wir dann noch eine Sprache um die Funktionalität der Elemente dieser Anwendung zu programmieren und abschließend eine Sprache, um die langfristige Speicherung von Nutzerdaten sicher zu stellen. Das können Sie als Erstsemester mit entsprechendem Arbeitsaufwand gut schaffen.

„Echte“ verteilte Anwendungen werden dagegen erst im Masterbereich eines Informatikstudiums grundlegend behandelt, weil Sie für den Einstieg in diesen Bereich eine Vielzahl an Kursen absolviert haben müssen, die Teil eines Bachelorstudiums der Informatik und verwandter Studiengänge ist. Hier seien die wichtigsten dieser Veranstaltungen und Themenbereiche ge-

nannt:

- Diskrete Mathematik und Graphentheorie
- Grundlagen der technischen Informatik
- Nachrichten- und Kommunikationstechnik
- Netzwerke und Internetsicherheit
- Imperative Programmierung
- Objektorientierte Softwareentwicklung
- (Relationale) Datenbanken
- Mobile Apps und Responsive Design
- Medienrecht

Wie eingangs beschrieben können Sie einfache verteilte Anwendungen wie beispielsweise eine Webanwendung auch ohne fundierte Kenntnisse dieser Bereiche entwickeln. Leider kann es Ihnen passieren, dass Sie nach dem Studienabschluss in einem Unternehmen tätig werden, wo Sie es mit vermeintlichen Fachkräften zu tun haben, deren Kenntnisse der oben genannten Bereiche zumindest teilweise mangelhaft sind. Diese wissen dann z.B., dass es so etwas wie „das Internet“ gibt, wie sie eine Webanwendung programmieren können und ähnliches, aber ihnen fehlt das fundamentale Verständnis für die zugrunde liegenden Technologien. Das führt dann langfristig zu massiven Schwierigkeiten, denn sobald neue Technologien auf dem Markt erscheinen (HTML 5 ist da ein sehr gutes Beispiel) sind diese „Fachkräfte“ nicht im Stande die neuen Technologien sinnvoll in bestehende Projekte zu integrieren und was recht bald dazu führt, dass Sie einen neuen Arbeitgeber benötigen.

Bevor Sie hier weiterlesen: Haben Sie die einleitenden Kapitel zumindest überflogen und den Abschnitt zur Vorbereitung für die Entwicklung von verteilten Anwendungen durchgearbeitet? Wenn nicht, dann tun Sie das bitte, bevor Sie hier weitermachen. Im laufenden Kapitel wird zum einen vorausgesetzt, dass Sie die Inhalte der einleitenden Kapitel kennen und insbesondere die nötigen Installationen abgeschlossen haben. Unklare Fachbegriffe können sie anhand der Anhänge zu jedem Kapitel leicht nachschlagen.

Wenn Sie ein anderes Paket als EasyPHP nutzen, dann werden einige Ausgaben des Rechners bei Ihnen anders aussehen als hier geschildert. Eine zusätzliche Hilfe für diese Fälle kann leider im Rahmen dieses Kurses nicht

geboten werden.

Einführende Programmierveranstaltungen bereiten Sie in aller Regel darauf vor, Programme für einen Computer zu entwickeln. Das ist in dieser Veranstaltung anders: Hier lernen Sie, Programme zu entwickeln, die über das World Wide Web auf allen möglichen Endgeräten laufen können. Mit Endgeräten ist alles gemeint, was im Kern ein Computer ist, aber nicht unbedingt als solcher zu erkennen. Ein Beispiel sind Smartphones.

5.1 Internet, WWW und HTML

In einer Veranstaltung zur **Nachrichten- oder Kommunikationstechnik** werden Sie lernen, dass das, was allgemein als „das Internet“ bezeichnet wird in Wahrheit etwas ist, das mit dem Begriff „World Wide Web“ bezeichnet wird. Der Begriff **Internet** bezeichnet dagegen Verbindungen zwischen zwei Netzwerken auf globaler Ebene. Ein Beispiel hierfür wären die transatlantischen Kabel, die die Verbindung zwischen den Kommunikations- und Datennetzen in Europa und Amerika sicherstellen.

Somit ist das Internet also die Voraussetzung für das World Wide Web: Ohne diese Leitungen und Funkstrecken (z.B. über Satellit) könnten wir keine Daten in ferne Länder übertragen oder von dort erhalten. Und Psy Gangnam Style wäre nie so berühmt geworden. Das WWW ist damit die wahrscheinlich umfangreichste verteilte Anwendung, die es weltweit gibt.

Doch was ist nun das **World Wide Web** (kurz **WWW**)? Hier handelt es sich um Dateien, die auf Computern rund um die Welt gespeichert sind. Diese Daten sind in bestimmten Formaten verfasst und dienen dazu, mit einem entsprechenden Programm, Texte und multimediale Inhalte anzuzeigen. Das Format in dem diese Dateien verfasst sind ist eine von mehreren Programmiersprachen, die in aller Regel auf die beiden Buchstaben ML endet. Dieses ML steht für Markup Language. Eine **Markup Language** wird von den meisten Softwareentwicklern nicht als „echte“ Programmiersprache anerkannt. Sie können mit einer ML nämlich lediglich Elemente oder sogenannte Container definieren, die Texte oder Verweise auf beliebige Daten beinhalten. Interaktive Programme (also Anwendungen, die auf die Eingaben von Nutzern so reagieren, dass sich ihre Inhalte ändern) sind in HTML wie in beliebigen anderen Markup Languages nicht realisierbar.

An dieser Stelle müssen wir uns drei Dinge klarmachen:

1. Die Inhalte (also Texte, Bilder, Hintergrundmusik, Videos usw. usf.)

sind NICHT Teil einer Markup Language. Vielmehr gibt es innerhalb einer Markup Language verschiedene Möglichkeiten, um auf den Speicherort zu verweisen, an dem diese Inhalte sich befinden. Das kann auch durch eine weitere Programmiersprache passieren. Allerdings ist es meist möglich, Texte direkt innerhalb einer Markup Language einzutragen.

2. Markup Languages sind statisch, weil Sie ausschließlich dazu dienen, Strukturen zu definieren. Alles dynamische wird in einer zweiten Programmiersprache programmiert.
3. In einer Markup Language benutzen wir sogenannte Tags (sprich Täg), um zu definieren, was für eine Art von Element wir definieren wollen. In HTML5 benutzen wird beispielsweise das Tag `<main>`, um einen Bereich zu definieren, der zentral für unsere Webanwendung ist. Wie dieser Bereich dann später angezeigt wird, ist für die Programmierung in HTML vollkommen irrelevant.

Allerdings ist die Behauptung, dass eine Markup Language keine Programmiersprache ist schlichtweg falsch. Richtig ist dagegen, dass es weder eine imperative noch eine deklarative Programmiersprache ist. Richtig ist aber auch und insbesondere, dass Markup Languages außerordentlich wichtig sind, wenn wir es mit großen Softwareprojekten zu tun bekommen. In der objektorientierten Softwareentwicklung wird beispielsweise die **UML** (kurz für Unified Markup Language) genutzt, um übersichtlich darzustellen, aus welchen Komponenten ein Softwareprojekt besteht. Die UML ist aber noch zu wesentlich mehr zu gebrauchen, doch das würde jetzt vom Thema ablenken.

Markup Languages sind also ein wichtiges Werkzeug, wann immer wir ein umfangreiches Softwareprojekt realisieren wollen. Sie helfen uns, Fehler zu vermeiden und klare Strukturen im Projekt zu schaffen, wodurch die Teamarbeit deutlich erleichtert wird.

Programme, die Dateien in einer Markup Language, nämlich HTML anzeigen können kennen Sie, denn Sie nutzen Sie täglich: Es sind Webbrowser. Aber damit die Dateien von anderen Computern auf Ihrem Endgerät angezeigt werden können ist es noch nötig, dass sie von dort auf Ihr Endgerät übertragen werden. Wenn Sie also bislang dachten, dass es einen echten Unterschied zwischen einem Download und dem Besuch einer Webanwendung (bzw. einer Webpage) gibt, dann wissen Sie es jetzt besser: Alles, was in Ihrem Webbrowser angezeigt wird liegt mindestens so lange im Speicher des Endgerätes, wie es angezeigt wird. Denn für einen Computer macht es im Grunde keinen Unterschied, ob Daten „nur“ im (RAM-)Speicher oder

als sogenannte Datei auf einem Speicher wie einer Festplatte vorliegen.

5.1.1 HTTP und HTTPS

Damit Daten auf Ihr Endgerät übertragen werden gibt es verschiedene Verfahren, von denen Sie als Nutzer eines Gerätes nichts sehen. Diese Verfahren werden in Form sogenannter Protokolle vereinbart. Wie die einzelnen Protokolle aussehen, das ist Teil der Veranstaltung Netzwerke und Internetsicherheit. Für diesen Kurs genügt es, dass Sie wissen, was ein **Protokoll** ist. Eines dieser Protokolle übersehen Sie praktisch jedes Mal, wenn Sie eine Webanwendung aufrufen: Es handelt sich um das **HTTP**, das **HyperText Transfer Protocol**. Das ist ein Protokoll, das einzig dazu entwickelt wurde, um die Übertragung von Dateien zu organisieren, die Teil des WWW sind.

Übrigens, wenn in der Adresszeile des Browsers nicht HTTP, sondern **HTTPS** steht, dann bedeutet das, dass der Datenaustausch verschlüsselt durchgeführt wird. Vielleicht erinnern Sie sich daran, dass Ihr Browser Sie mit einer Warnung genervt hat, wonach eine Webpage ein ungültiges Zertifikat genutzt hat. Ein solches Zertifikat ist bei der verschlüsselten Datenübertragung essentiell: Es ist der Schlüssel, mit dem Ihr Browser die Daten verschlüsselt, die an eine bestimmte Seite übertragen wird. In einer Veranstaltung zur Netzwerksicherheit werden Sie lernen, was dabei im Hintergrund passiert und warum Browser manchmal ein gültiges Zertifikat nicht anerkennen.

Wichtig: Das, was Sie hier kennen lernen gilt genauso für Anwendungen im WWW. Mit den selben Grundlagen, mit denen Sie eine Webanwendung entwickeln, können Sie also auch ein Spiel oder eine beliebige andere Anwendung entwickeln, die auf jedem Endgerät läuft, das vernetzt ist und auf dem ein Webbrowser läuft. Unterschiede beim Betriebssystem wie zwischen **Android** und **iOS** oder **Windows** und **Linux** sind dann vollkommen belanglos.

Kontrolle

- Sie wissen jetzt, was der Unterschied zwischen Internet und WWW ist.
- Sie wissen, dass eine Webanwendung lediglich eine Ansammlung von Dateien ist, die über das Internet auf Ihren Rechner übertragen werden.
- Sie wissen, dass es Standards für diese Übertragung gibt, die in Form sogenannter Protokolle veröffentlicht und genutzt werden.

- Sie wissen, dass diese Dateien in einer von vielen Sprachen verfasst sind, die als Markup Languages bezeichnet werden.
- Sie wissen, dass die beiden Buchstaben ML im Namen einer Sprache für Markup Language stehen können.

Ausblick

Jetzt werden Sie mehr über die Programmiersprachen erfahren, die neben Markup Languages bei Webanwendung und Webanwendungen zum Einsatz kommen.

5.2 Funktionalitäten unserer Webanwendung

Der folgende Satz aus der Architektur ist für uns als SoftwareentwicklerInnen essentiell:

Form follows function!

Das bedeutet, dass zuerst die Funktion definiert werden muss, bevor es um gestalterische Fragen geht. Aber wie Sie ohne schon erkennen konnten kommt vor der Funktion die Struktur. Und zur Funktion gehört dann noch die Speicherung und Wiederverwendung von Nutzerdaten. Im Gegensatz dazu konzentrieren sich MediendesignerInnen auf das Design.

Zusammengefasst folgt daraus für uns:

1. Zuerst legen wir die Struktur fest,
2. dann legen wir für jedes Element der Struktur die Funktion(en) fest, die dieses Element anbieten soll.
3. Abschließend legen wir fest, welche Nutzereingaben wie verarbeitet und ob sie gespeichert bzw. wiederverwendet werden sollen.

Dagegen gilt hier:

Design ist für uns irrelevant.

Wenn Sie sich dagegen mit Design beschäftigen wollen, dann wechseln Sie bitte zu einem Studium des Medien- und/oder Kommunikationsdesigns;

dort lernen Sie das, was Sie suchen. Hier sind Sie dann schlicht und ergreifend am falschen Platz.

Denn wir werden uns hier grundsätzlich mit dem Entwurf und der Entwicklung von Webanwendungen beschäftigen. Üblicherweise wird bei vergleichbaren Kursen und Tutorien im Netz damit begonnen, einen Webshop in HTML4.01 zu entwickeln, selbst wenn die Anbieter behaupten, es handle sich um eine Einführung in HTML5. Im Gegensatz dazu erhalten Sie hier einen ersten Einblick in aktuelle Methoden bei der Entwicklung von Webanwendungen.

5.2.1 Erste Schritte zur Webanwendung

Und der erste Schritt dazu hat etwas mit Stift und Papier zu tun:

1. Notieren Sie, was Ihre Webanwendung tun soll.
2. Streichen Sie alles durch, was angibt, wie sie es umsetzen soll.
3. Notieren Sie dann, welche Elemente Nutzer für welche Funktion angezeigt bekommen sollen. (Auch das Anzeigen von Texten ist eine Funktion.)
4. Streichen Sie alle konkreten Textentwürfe durch. („Guten Tag, lieber Nutzer“ ist ein konkreter Text, „Hier Begrüßung des Nutzers einblenden“ dagegen nicht.)
5. Streichen Sie alles durch, was definiert, wie diese Dinge angezeigt werden sollen. (Zur Erinnerung: Wir machen Medieninformatik, nicht Mediendesign.)

Wenn Sie jetzt gleich zum nächsten Abschnitt weitergehen, dann machen Sie etwas falsch, denn bevor Sie nicht ausführlich überlegt haben, was Sie auf Ihrer Seite unterbringen wollen, brauchen wir uns über die Programmierung oder das Design keine Gedanken zu machen.

5.2.2 Projektdokumentation und Arbeitsumgebung

Und damit sind wir beim wichtigsten Arbeitsmittel beim Projektstart: Papier. Für eigene Notizen im Format A4 oder A5, für Gruppenarbeiten sollten sie auch Bögen im Format A3 oder besser noch A2 besorgen. Klebeband, Textmarker und Stifte brauchen Sie natürlich auch. Schließlich wollen wir keine Papierflieger bauen.

Das mag in der Zeit von Digitalisierung und Vernetzung seltsam klingen, aber da Papier nun wirklich kein Luxusgut ist und es nunmal leichter ist, mehrere Blätter auf einen Tisch zu legen, um zu vergleichen, was den Mitgliedern eines Teams am besten gefällt, führt an dieser Stelle kein Weg an Stift und Papier vorbei. Idealerweise nutzen Sie dabei wenigstens ein Dutzend unterschiedlicher Farben. So können Sie beispielsweise jedem Mitglied Ihres Teams eine Farbe zuordnen. Dann ist erkennbar, wer welche Einträge vorgenommen hat.

Bei der Programmierung werden Sie noch mit einem Repository arbeiten, aber für den Moment bleiben wir bei Stift und Papier.

Für alle, die bereits mit Software gearbeitet haben, die eine solche Arbeit z.B. via Tablet ermöglicht und für diejenigen, die das gerne tun würden: Der einzige Grund, aus dem ich diese Methoden hier nicht unterstütze ist der, dass es heute noch keine günstigen Arbeitstische gibt, die diese Methoden für Teams unterstützen. Bei rund einhundert Studierenden pro Semester reden wir hier über wenigstens 10 entsprechende Tische, bzw. deutlich mehr als 100.000,- € zuzüglich der Kosten für zwei Räume à 50 m², in denen diese Tische stehen. Denn Sie werden schlicht und ergreifend eine große Arbeitsfläche benötigen; da stellen Tablets aufgrund Ihrer beschränkten Formate eine zu große Beschränkung dar. Ansonsten bin ich hier voll und ganz auf Ihrer Seite und hoffe, dass die nötige Hardware sobald als möglich zu bezahlbaren Preisen auf den Markt kommt. Sie wollen ein Beispiel für das sehen, was ich meine? Dann sehen Sie sich den Film "Casino Royale" mit Daniel Craig an. Dort können Sie sehen, wie eine professionelle digitale und vernetzte Arbeitsumgebung für Teams aussehen kann. Allerdings fehlen dort Interfaces mit Tastatur und Maus, die fürs Programmieren unbedingt nötig sind.

5.2.3 Erste Übung

Die folgende Aufstellung ist für MS-Studierende gedacht, die in diesem Semester die Leistungsnachweise für PRG und Projekt 1 erwerben wollen. Für diese gilt:

Alle Schritte müssen in der genannten Reihenfolge bis zum 25. März 2016, 12.00 Uhr erledigt sein, sonst ist keine Teilnahme am Projekt 1 in diesem Semester mehr möglich.

Sollte die Veranstaltung in zwei Gruppen durchgeführt werden, dann gilt für die erste Gruppe die gerade genannte Frist, für die zweite Gruppe eine Frist bis zum 1. April 2016, 12.00 Uhr. (Nein, das ist kein Aprilscherz.)

1. Bevor Sie in einem Team mit einem Projekt beginnen, überlegen Sie für sich, was für ein Softwareprojekt Sie gerne umsetzen würden. Wichtig ist hier nicht, ob Sie sich vorstellen können, es umzusetzen, sondern dass Sie sich ein interaktives Programm überlegen, das Sie schon immer haben wollten.
 - Beschreiben Sie, was dieses Programm tut, bzw. wie Nutzer damit arbeiten, spielen oder wie auch immer interagieren.
 - Streichen Sie alles, was mit Design und konkreten Inhalten zu tun hat.
 - Es soll kein Roman dabei herauskommen: Kürzen Sie Sätze mit mehr als 10 Wörtern. Streichen Sie alle Adjektive. (Interessanterweise gibt es einen Literaturnobelpreisträger, der in seinen Romane die zweite dieser Vorgaben umgesetzt hat: Ernest Hemingway)
 - Formulierungen wie „Es ist eine AAA-roguelike X4-Sim mit Shooter-RPG-Elementen.“ gehören ins Marketing und zeigen, dass Sie nicht im Stande sind auszudrücken, was Ihre Anwendung ausmacht.
2. Wenn Sie Ihre Projektidee in ein oder zwei Stunden wie gerade beschrieben grob skizziert haben, dann arbeiten Sie sich an Ihrem Rechner in Git ein. (Beachten Sie dabei alles, was im Kapitel „Vorbereitung fürs Programmieren“ steht.)
3. Erstellen Sie in Ihrem Repository ein Verzeichnis mit dem Namen Dokumentation. Speichern Sie darin Ihren Projektvorschlag in einer Datei mit dem Namen `Projektidee.txt`. Wenn Ihnen nichts eingefallen ist, dann tragen Sie das in der Datei ein.
4. Erstellen Sie (wenn noch nicht passiert) ein Repository bei GitHub, mergen Sie Ihr lokales Repository damit (bzw. pushen Sie eben Ihr lokales Repository auf GitHub) und tragen Sie mich als Collaborator ein. Mein Username bei GitHub lautet `MarkusAlpers`.
5. Melden Sie sich bei Helios für die Leistungsnachweise PRG und Projekt 1 an.
Sollte die Anmeldung zurzeit nicht möglich sein, senden Sie mir bitte eine kurze Nachricht, damit ich in der Verwaltung alles nötige veranlassen kann.
6. Senden Sie mir an meine Hochschuladresse `markus.alpers@haw-hamburg.de` eine E-Mail **von Ihrer Hochschulmailadresse** (private Mailadressen werden nicht angenommen) mit folgenden Angaben:

- Als Bezug:
- Anmeldung Projekt 1 (MS)
- Als Inhalt die folgenden Zeilen (nur ausfüllen, also nicht explizit Nachname als Wort angeben, sondern Ihren Nachnamen usw. eintragen):
- Nachname, Vorname, Matrikelnummer
- Ihren Usernamen bei GitHub
- Den Namen Ihres Repositories

5.2.4 Mehr zu den Leistungsnachweisen für alle

Programmieren ist wie Fahrradfahren: Manche schaffen es auf Anhieb loszufahren und können relativ schnell an Wettrennen teilnehmen, andere brauchen Jahre, um auch nur den Einstieg zu schaffen. Es hat nichts mit auswendig lernen oder anderen Arten des „Lernens“ im Sinne von „büffeln“ zu tun, sondern beim Programmieren geht es (genau wie bei mathematischen Verfahren oder handwerklichen Tätigkeiten) fast ausschließlich darum, es wieder und wieder anzuwenden. Wenn Sie sich also nicht zu Hause hinsetzen und jede Woche mehrere Stunden selbst programmieren, dann ist Ihre Anwesenheit in diesem Kurs sinnlos und Sie werden keinesfalls einen Leistungsnachweis erhalten.

Umgekehrt hat Programmieren auf Hochschulebene sehr viel mit Planung und Konzeption zu tun. Wenn Sie also die Inhalte der Veranstaltung ignorieren und einfach drauflos programmieren, wird auch das nicht funktionieren. Viele Studierende suchen dazu im Netz nach Lösungen, die sie nicht verstehen. Am Ende sagen Sie dann, dass Programmieren langweilig oder sinnlos ist. Dabei könnten Sie mit dem, was wir in dieser Veranstaltung besprechen praktisch alles selbst programmieren, was Sie an fertigen Programmen kaufen können.

5.2.5 Mehr zu den Leistungsnachweisen für MS-Studis

Wenn Sie einen Abschluss in Media Systems erwerben wollen, dann gibt es unter anderem die beiden Leistungsnachweise `Projekt 1` und `Einführung ins Programmieren`. Diese beiden Leistungsnachweise erhalten Sie, wenn Sie das Projekt 1 erfolgreich abgeschlossen haben. Nachdem Sie die eben genannten Bedingungen erfüllt haben wird die gesamte Kommunikation zum Projekt über Ihr Repository erfolgen. Der weitere Ablauf wird so erfolgen, dass ich Ihren individuellen Leistungsstand regelmäßig (mindestens alle vier Wochen einmal) anhand dessen kontrolliere, was im Repository steht und Ihnen dort individuelle Hinweise gebe,

wie Sie weiterarbeiten können.

Laut Modulhandbuch sollen Sie im Projekt nachweisen, dass Sie die Inhalte der Veranstaltung erfolgreich in einer Gruppe umgesetzt haben. Als Arbeitsaufwand sind 80 Stunden vorgesehen. Das schließt nicht die Vor- und Nachbearbeitung der Vorlesung bzw. des Seminars ein. Damit ergeben sich, dass Sie sich außerhalb der Veranstaltung rund 10 Stunden pro Woche mit den Themen der Veranstaltung und der Durchführung des Projekts beschäftigen müssen.

Häufig werde ich aufgefordert, Ihnen genau Angaben dazu zu machen, was Sie machen sollen. Das ist aber so konkret nicht möglich: Programmieren hat etwas mit der Fähigkeit zu tun, die eigenen Vorstellungen zu abstrahieren und sie in einer Form auszudrücken, die ein Computer ausführen kann. Ob Sie das schaffen lässt sich aber nicht daran messen, ob Sie die Umsetzung in fünf oder in 500 Zeilen durchführen. Es ist im Grunde so ähnlich wie mit Fahrradfahren: Wenn jemand mehrere hundert Meter mit einem Fahrrad ohne Stützräder gefahren ist, dann ist klar, dass er/sie es kann. Würde ich das aber auf den Leistungsnachweis übertragen, dann würden viele von Ihnen, die eine ausreichende Leistung erbracht haben keinen Leistungsnachweis erhalten. Und das wäre unnötig demotivierend. Deshalb lege ich keine solche messbare Grenze fest.

Es gibt aber noch einen weiteren Grund: Für den Leistungsnachweis müssen Sie eine gewisse Qualität erreichen, weil Sie MedieninformatikerInnen sind. Würde ich eine Liste aller möglichen Qualitätskriterien aufstellen, dann wäre das Ergebnis ein Katalog von mehreren hundert Seiten, den Sie erst dann verstehen würden, wenn Sie mehrere Jahre programmiert hätten. Einen Teil davon finden Sie allerdings in diesem Buch. Und selbst wenn Sie davon nur einen Teil umsetzen, stehen Ihre Chancen gut, den Leistungsnachweis zu erwerben.

5.2.6 Mehr zum Leistungsnachweis für MT-Studis

In Ihrem Studiengang gibt es für die Veranstaltungen `Programmieren 1` und `Programmieren 2` einen benoteten Leistungsnachweis, der nach Bestehen eines Projekts ausgestellt wird, das Sie in PRG2 durchführen. Dieses Projekt machen Sie voraussichtlich bei Herrn Wagener. Zur Durchführung kann ich deshalb nichts sagen. Bei mir erlernen Sie die Grundlagen, über die es eine schriftliche Prüfung am Ende dieses Semesters geben wird. Sie werden also anders als in der Schule nur eine Prüfung für die gesamte Veranstaltung schreiben.

5.3 Technische Grundlagen verteilter Anwendungen

Sie wissen bereits, dass eine verteilte Anwendung aus mehreren Programmen besteht, die über ein Netzwerk¹ miteinander kommunizieren. In den nachfolgenden Abschnitten und Kapiteln dieses Buches werden wir Schritt für Schritt alles besprechen, das Sie benötigen, um Sie Ihre erste verteilte Anwendung starten können: Eine Webanwendung, zu der es Nutzerkonten gibt und deren Inhalte sich abhängig von den Eingaben der Nutzer ändern können.

Wie Sie schon aus dem Abschnitt zur Vorbereitung auf die Entwicklung von verteilten Anwendungen wissen, gliedert sich dieser Teil des Buches in die folgenden Bereiche:

- Festlegung, welche Funktionalitäten unsere Webanwendung enthalten soll.
- Betrieb eines Webserver und Zugriff durch einen Webbrowser.
- Einführung in die Markup Language HTML, um statische Webanwendung zu entwickeln.
- Bereitstellung dieser Webanwendung auf dem Webserver.
- Erweiterung der Anwendung durch die Nutzung von PHP, sodass wir eine dynamische Webanwendung erhalten.
- Einbindung einer Datenbank, die auf dem Webserver bereit gestellt wird, um so die Speicherung von Daten über individuelle Nutzer zu realisieren.

Wenn Sie diese sechs Punkte beherrschen, dann verstehen Sie die grundsätzlichen Abläufe, die bei jeder verteilten Anwendung vorkommen. Der einzige Unterschied zwischen Ihnen und einem professionellen Entwickler solcher Anwendungen besteht in zwei Punkten: Zum einen die langjährige Erfahrung, zum anderen ein wesentlich besseres Verständnis, wie diese Abläufe stattfinden und wie hochkomplexe Projekte sinnvoll realisiert werden können.

5.4 Erste Gehversuche mit Server und Client

Nachdem wir die Planung der Webanwendung begonnen haben, fahren wir also damit fort, den Webserver zu nutzen bzw. zu prüfen, ob er soweit

¹oder über ein Netzwerk von Netzwerken wie das beim „Internet“ der Fall ist

funktioniert, wie das für unsere Zwecke nötig ist:

Prüfen Sie bitte, ob EasyPHP den Server gestartet hat. Wenn das der Fall ist, öffnen Sie bitte einen Browser und geben dort in der Adresszeile (nicht im Suchfeld einer Suchmaschine!) das Wort `localhost` ein. Groß- und Kleinschreibung ist hier nicht von Belang.

Nun sollte eine Seite angezeigt werden, auf der am oberen Rand einige wenige Einträge mit Titeln wie „version“ erscheinen. Im Zentrum der Seite sollten drei Verzeichnisse angezeigt werden. Wenn Sie diese anwählen, werden Sie feststellen, dass Sie allesamt leer sind.

Wiederholen wir an dieser Stelle die Frage, was denn eigentlich der **Server** und was der **Client** ist (momentan haben wir jeweils nur einen): Der Server ist ein sogenannter Apache Server. Dabei handelt es sich um ein Programm, das mit Hilfe von Dateien, auf die es Zugriff hat, verschiedene Dateien generieren kann, die über ein beliebiges Netzwerk übertragen und von einem Browser als etwas angezeigt werden, das wir unter der Bezeichnung Webanwendung bzw. Webpage kennen. Das Programm, das also die Daten bzw. Dateien vom Server nutzt (eben der Client) ist dementsprechend der Browser, den wir nutzen.

Wenn es keine Datei gibt, die eine Webanwendung generiert, dann erzeugt ein Webserver Daten, die vom Client so angezeigt werden, wie Sie das von Ihrem Rechner beim Dateibrowser kennen: Da werden einerseits Dateinamen mit Endungen und andererseits Verzeichnisnamen angezeigt. Und für jeden dieser Einträge gibt es dann noch ein Symbol. Aber auch hier haben wir es wieder mit HTML-Dateien zu tun, denn sonst könnte der Browser sie nicht anzeigen.

Aber selbst wenn der Server auf eine Datei Zugriff hat, die definiert, wie eine Webanwendung aussehen soll, bedeutet das nicht, dass der Server basierend auf dieser Datei eine Webanwendung-Ansicht generiert. Vielmehr müssen Sie ihn so konfigurieren, dass er das tut.

Für den Fall, dass Sie sich wundern: Die Bezeichnung **localhost** ist eine Art Alternativbezeichnung für eine bestimmte IP-Adresse. Darüber können wir einige Server nutzen, die auf unserem Rechner aktiv sind.

Was eine **IP-Adresse** ist? Ach ja richtig, Sie hatten ja noch keine Veranstaltung über Netzwerke... Eine IP-Adresse ist in Netzwerken (und damit auch im Internet) so etwas wie eine Telefonnummer für Computer. Diese IP-Adressen sind also eine Möglichkeit, damit Computer über Netzwerke Daten austauschen können. Denn ohne IP-Adresse hätten Sie ja kei-

ne Möglichkeit, einen bestimmten Rechner anzusprechen. Egal, ob Sie das wollen oder nicht: Dieser Datenaustausch passiert, wenn Sie einen Rechner, ein Smartphone oder welches vernetzte Gerät auch immer einschalten². Im Gegensatz zu Telefonnummern sind IP-Adressen aber nicht automatisch fest einem Computer zugeordnet, sie sind also im Regelfall dynamisch zugeordnet.

Wenn sie nun im Netz surfen, dann geben Sie im Regelfall nicht die IP-Adresse eines Rechners ein, sondern den „Namen“ des Standorts einer Webanwendung. Solche Namen haben wie alles einen Fachbegriff: Die **URL** (kurz für Unified Remote Location) einer Webanwendung der HAW Hamburg ist beispielsweise `www.haw-hamburg.de` wobei Sie im Regelfall das `www.` am Anfang weglassen können. Eine andere URL der HAW Hamburg lautet beispielsweise `www.mt.haw-hamburg.de`. Der Begriff URL ist zwar gebräuchlich, aber im Kern falsch, denn auch wenn das L für Location bzw. Standort steht, können Sie aus der URL nichts über den Standort eines Servers aussagen. Der einzige Standort, den Sie aus einer URL ablesen können ist der Standort des Verwaltungsgremiums, das für die Vergabe von URLs zuständig ist. (In Deutschland ist das die DENIC.) Präziser ist da die Bezeichnung **URI** (kurz für Unified Remote Identifier).

Leider gibt es keine unterschiedliche Bezeichnung für eine einzelne Ansicht einer Webanwendung oder die Gesamtheit aller Webanwendung, die unter einer URL zu finden sind. Sie müssen also jeweils überlegen, ob nun eine einzelne Seite oder die Gesamtheit aller Seiten einer Webanwendung gemeint ist.

Und jetzt kommt etwas, dass viele Menschen nicht wissen: Wenn Sie eine solche URI in einen Webbrowser eingeben, dann schlägt dieser in einem Verzeichnis nach, welche IP-Adresse zu dieser URL gehört. Ein solches Verzeichnis ist ein Programm (hier spricht man auch von einem **Dienst**), denn es muss ja im Stande sein, eine Antwort zu senden. Und wie nennen wir solche Programme? Richtig, es ist ein Server, der als Domain Name Server, kurz **DNS** Bezeichnet wird. Warum dieser Server so heißt, was Domänen sind, wie er funktioniert usw. usf. ist auch wieder Teil jeder Veranstaltung über Netzwerke. Ob ein Dienst nun ein einzelnes Programm oder eine verteilte Anwendung ist, soll uns an dieser Stelle nicht weiter interessieren. Der Begriff wird vorrangig im Bereich der **Nachrichten- und Kommunikationstechnik** verwendet, während wir als (Medien-)InformatikerInnen je nach Situation von Programm, verteilter Anwendung, Server, Client oder ähnlichem sprechen.

²Wie das im Detail funktioniert und wann genau eine Datenübertragung begonnen wird ist Teil jeder Veranstaltung zu IT-Netzwerken.

Wie bei allen Servern gilt auch hier: Dieser Server kann sich auf Ihrem Rechner befinden oder auf einem beliebigen Rechner irgendwo im Netz. Als Frau von der Leyen vor einigen Jahren forderte man müsse bestimmte Seiten im Netz sperren, indem man bei Suchanfragen ein Stoppschild einblenden würde, bewies Sie damit, dass sie nicht einmal die einfachsten Strukturen des Internet kannte. Denn um diese Stoppschilder zu realisieren, hätte jeder DNS-Server weltweit angepasst werden müssen. Und nur wer keine Ahnung vom Unterschied zwischen nationalem und internationalem Recht hat, kann auf die Überzeugung verfallen, dass beispielsweise die Administratoren eines DNS-Servers in Timbuktu sich bei ihrer Arbeit nach den Gesetzen der Bundesrepublik Deutschland richten. Vielleicht glaubt er oder sie auch an den Weihnachtsmann; wir beschäftigen uns hier mit der Informatik und ihren Auswirkungen in der Realität, deshalb wissen wir es besser.

Aber zurück zu unserem einfachen Webserver, bzw. zum Aufruf localhost im Webbrowser. Der Begriff localhost ist auch eine URI. Das bedeutet, dass ein Webbrowser im Regelfall auf einem DNS-Server nachsieht, welche IP-Adresse zu dieser Adresse gehört. Im Gegensatz zu anderen URIs gibt es aber für localhost eine Konvention: Dieser URL ist eine statische IP-Adresse zugeordnet: Es ist 127.0.0.1 Geben Sie diese Zahlenkombination einmal in Ihrem Browser in die Adresszeile ein.

Wenn Ihr Browser jetzt eine Übersicht von Internetseiten präsentiert, wie das bei Google der Fall ist, dann hat Ihr Browser entweder gar keine Adresszeile mehr, was leider immer öfter der Fall ist oder Sie haben die 127.0.0.1 in das Suchfenster eingetragen. Im zweiten Fall sollten Sie nochmal ernsthaft in sich gehen und überlegen, ob Sie nicht doch lieber irgend etwas anderes studieren wollen, denn dann haben Sie einen sehr hohen Nachholbedarf, was die Grundlagen der Bedienung eines Computers angeht.

Nochmal zur Erinnerung bezüglich der Begriffe statisch und dynamisch: Wenn wir in der Informatik davon reden, dass etwas statisch ist, dann handelt es sich dabei um eine Konvention, bzw. eine Vereinbarung und nicht um eine Art Naturgesetz. Denn im Gegensatz zu dem, was jemand mit dem Begriff „statisch“ verbinden könnte, sind alle Daten dynamisch: Ein Computer kann nur mit Werten arbeiten, die er in die Register seines Prozessors lädt. Und der Inhalt eines solchen Registers kann nicht statisch sein. Das wiederum entspricht auch letztlich jedem Naturgesetz: Das einzig konstante ist die Veränderung.

Wichtig: Diese Art der IP-Adressen ist als **IPv4** bekannt, wobei das v für Version steht. Der nächste genutzte Standard für IP-Adressen lautet **IPv6**

und wird immer häufiger eingesetzt. Bei Webanwendungen ist aber zurzeit IPv4 noch Standard. Die Unterschiede zwischen den Versionen 4 und 6 sind auch wieder Teil jeder Veranstaltung zu IT-Netzwerken.

Kontrolle

- Sie wissen, wie Sie auf Ihren Rechner einen Webserver starten und Sie wissen auch, warum hier nicht automatisch eine Webanwendung erzeugt wird.
- Sie wissen, dass auf Ihrem Rechner sowohl Client als auch Server laufen können und Sie wissen, wie man diese Art von Server bzw. Client bei Webanwendungen nennt, bzw. welche Programme Sie dafür nutzen.
- Sie wissen, wie Sie auf die Einstiegsseite kommen, die der Webserver generiert.
- Sie kennen auch die Aufgabe eines DNS Servers, wissen was die Abkürzungen IP und URL bzw. URI bedeuten und wie diese zusammenhängen oder eben nicht zusammenhängen.
(Anmerkung: Die Bezeichnung DNS Server ist im Grunde doppelt gemoppelt, denn das S in DNS steht ja bereits für Server, aber dennoch ist es üblich, diese Server so zu nennen.)

5.4.1 Eine erste Webanwendung mit PHP

Erstellen Sie bitte das folgende Programm mit Hilfe eines Editors und speichern Sie es im Verzeichnis `C:/ProgramFiles(x86)/EasyPHP-DevServer-14.1VC11/data/localweb` unter dem Namen `phpinfo.php`. Wenn Sie nicht EasyPHP installiert haben oder das Verzeichnis von EasyPHP bei der Installation geändert haben, müssen Sie prüfen, wo Sie die Datei speichern müssen. Es ist auch möglich, dass der Name des Verzeichnisses sich bei einer späteren Version von EasyPHP ändern wird.

```
<html>
<body>
<?php
phpinfo();
?>
</body>
</html>
```

Rufen Sie nun die Webanwendung auf. In der Ansicht sollte neben den drei Verzeichnissen auch der Dateiname `phpinfo.php` erscheinen. (Sonst haben Sie beim Speichern einen Fehler gemacht.) Wählen Sie diesen Dateinamen einmal mit der Maus an, so wie Sie es bei einem beliebigen Link auf einer Webanwendung tun würden.

Sie erinnern sich an die Aussage vom Anfang, wonach die Entwicklung verteilter Anwendungen umfangreiche Kenntnisse aus vielen Bereichen benötigt? Die Seite, die Ihnen jetzt angezeigt wird, zeigt Ihnen die Konfiguration (also die Einstellung) aller Komponenten an, die alleine bezüglich der Darstellung von Elementen auf einer Webanwendung relevant sind. Das ist aber nur ein Teilbereich aller Konfigurationen, die bei der Entwicklung und beim Betrieb einer Webanwendung oder einer anderen verteilten Anwendung relevant sind. Und? Sind Sie schockiert über die Masse an Dingen, die Sie nicht verstehen? Gut, denn dann werden Sie hoffentlich nicht so überheblich wie die breite Masse an Entwicklern, die glauben, dass Sie Meister des Netzes sind, nur weil sie im Stande sind, eine Webanwendung zu entwickeln und zu betreiben. Denn dafür brauchen Sie so gut wie nichts von dem zu verstehen, was Ihnen gerade angezeigt wird. Aber bedenken Sie: All diese Einstellungen sind wichtig und müssen richtig konfiguriert sein, damit Ihre Webanwendung richtig funktioniert und um es Angreifern so schwer wie möglich zu machen, Ihre Webanwendung zu manipulieren.

Kontrolle

Sie wissen jetzt grundsätzlich, wie Sie eine Webanwendung auf einen Server übertragen können, und dass diese Übertragung bereits genügt, damit diese von Nutzern aufgerufen werden kann.

Sie wissen jedoch bislang nichts darüber, wie Sie eine Webanwendung programmieren müssen, um eigene Inhalte darauf zu präsentieren. Wir haben auch noch nicht über mögliche rechtliche Folgen einer Veröffentlichung von Inhalten auf einer Webanwendung gesprochen oder darüber, welche rechtliche Folgen es haben kann, dass Sie überhaupt eine Webanwendung ins Netz stellen. Doch bevor wir über weitere Details der Programmierung von Webanwendung sprechen, sollten wir dafür fünf Minuten aufwenden.

Ausblick

Nachdem Sie jetzt die Voraussetzungen geschaffen haben, um auf Ihrem System Webanwendungen zu entwickeln, wenden wir uns den Details der Softwareentwicklung in den drei Bereichen zu, mit denen Sie zu tun haben werden: Modell, Steuerung und langfristige Speicherung. Für jeden dieser

drei Teile verwenden wir hier eine eigene Programmiersprache: Für das Modell HTML in der Version 5, für die Steuerung PHP in der Version 5.6 und für die langfristige Speicherung MySQL in der Version 5.6. An dieser grundsätzlichen Aufteilung und der konzeptionellen Arbeit wird sich bei Webanwendungen nichts ändern, auch wenn Sie z.B. JavaScript anstelle von PHP oder MongoDB anstelle von MySQL verwenden.

Seit einigen Monaten ist PHP 7 auf dem Markt. Wenn Sie damit weiterarbeiten wollen, sollten Sie sich hier einarbeiten. Da ich mich zurzeit darum kümmere, dieses Buch fertig zu stellen habe ich mir die Änderungen noch nicht im Detail angesehen und werde das auch bis Ende des Semesters nicht tun.

5.5 Programmierung von Webanwendungen

Die bekannteste Markup Language ist **HTML**, die HyperText Markup Language. Seit dem Herbst 1999 wurde HTML in der Version 4.01 verwendet, seit Herbst 2014 steht HTML5 zur Verfügung. Beide Versionen haben nur gemein, dass HTML5-Dokumente auch HTML4-Code beinhalten können, und dass dieser wie bisher ausgeführt wird. Sonst nichts. Wer denkt, HTML5 sei nur eine Erweiterung von 4.01, hat es nicht verstanden. Es ist im Kern eine vollständig neue ML, die entwickelt wurde, um alle nur denkbaren multimedialen und auch interaktiven Inhalte verfügbar zu machen.

Die Kompatibilität zu Version 4 wurde eingeführt, damit es keine Probleme mit alten Webanwendung gibt. Insbesondere unterstützt HTML5 direkt die objektorientierte Softwareentwicklung und ist darauf ausgelegt im Verbund mit JavaScript jede Art von Programmen im Browser zu realisieren. Das bedeutet unter anderem, dass **Flash** bzw. **ActionScript** nunmehr überflüssig geworden ist. **ActionScript** ist die Sprache, in der Flash-Anwendungen programmiert werden und die zum Eigentum von Adobe gehört. Es ist eine Alternative zu JavaScript.

Wie gesagt definieren Sie in einer Markup Language lediglich Elemente bzw. Container, die Texte und Verweise auf Dateien enthalten. Die Darstellung des Inhalts dieser Container wird dagegen über sogenannte Cascading Style Sheets (kurz **CSS**) festgelegt und über die Einstellungen des Browsers auf dem Rechner eines Nutzers.

An dieser Stelle ein Hinweis für diejenigen von Ihnen, die bereits mit HTML programmiert haben: (Alle anderen lesen bitte beim nächsten Absatz weiter, da Sie das hier noch nicht verstehen können.) Sie haben wahrscheinlich

schon Attribute wie align genutzt. Das dürfen Sie unter HTML5 nicht mehr in dieser Form tun! Hier müssen Sie alles, was sich auf das Layout bezieht in CSS programmieren. Das mag ungewohnt sein, ist aber sinnvoll, weil auf diese Weise die Trennung zwischen der Definition von Elementen (was unter HTML passiert) und ihrer Darstellung (was unter CSS programmiert wird) eindeutig geklärt ist.

Um nun Programme wie beispielsweise Spiele im Browser zu entwickeln werden verschiedene Ansätze gewählt. Auch hier wurde durch die Einführung von HTML5 ein Standard eingeführt: Während bei Version 4 keine Programmiersprache als Standard vorgesehen ist, sieht Version 5 die Anbindung von Programmen in der Programmiersprache JavaScript als Standard vor. JavaScript ist eine der mächtigsten Sprachen, die Sie zurzeit nutzen können, um Anwendungen auf einem Rechner oder in einem Browser zu entwickeln. Auch aus diesem Grund werden Sie JavaScript immer öfter auf Webanwendung finden.

Eine Sprache, die früher häufig eingesetzt wurde und deshalb auch heute noch weit verbreitet ist, ist **PHP**. JavaScript ist zwar mächtiger und aus diesem Grund empfehlenswerter, aber es hat einen Nachteil für Sie, wenn Sie sich in die professionelle Teamarbeit einarbeiten wollen: Während Sie in PHP nichts programmieren können, das Sie in HTML oder CSS programmieren können, ist es möglich, nahezu eine vollständige Webanwendung in JavaScript zu entwickeln. Das führt dann bei Einsteigern schnell dazu, dass der JavaScript-Programmierer Teile übernimmt, die andere umsetzen sollten. Und die haben dann nichts mehr zu tun.

Ursprünglich wollten die Entwickler durch den Einsatz von Sprachen wie PHP und JavaScript nur erreichen, dass Webanwendung dynamisch werden. Eine **dynamische Webanwendung** ist schlicht eine Webanwendung, deren Inhalte sich z.B. durch Eingaben von Nutzern ändern können. Die Möglichkeiten, die HTML5 zusammen mit JavaScript bietet gehen weit darüber hinaus.

Zu guter Letzt brauchen Sie als Entwickler einer Webanwendung noch eine Möglichkeit, um Daten langfristig zu speichern. Hier kommen die sogenannten **Datenbanken** zum Einsatz: Eine Datenbank ist eine Ansammlung von Tabellen, in denen Daten in standardisierter Form abgespeichert werden. Im Gegensatz zu Dateien liegen Datenbanken dabei ständig im Speicher eines Rechners vor und können deshalb genutzt werden, ohne dass Sie zunächst von einer Festplatte oder einem USB-Stick geladen werden müssten.

Auch hier bekommen wir es mit Programmiersprachen zu tun, namentlich

mit den sogenannten Query Languages (kurz QL) oder Structured Query Languages (kurz **SQL**), was übersetzt so viel wie strukturierte Anfrage-Sprache bedeutet. Denn nichts anderes tun diese Sprachen: Sie ermöglichen es einem Nutzer in standardisierter Form Anfragen an eine Datenbank zu stellen und geben die Antwort der Datenbank in einer Form aus, die für Menschen lesbar ist. Die bekannteste dieser Sprachen ist **MySQL**.

Wichtig: SQL steht dagegen nicht für sequel (zu Deutsch: Nachfolger). Es ist aber im englischsprachigen Raum durchaus üblich, Abkürzungen nicht zu buchstabieren, sondern sie wie ein Wort auszusprechen. Deshalb wird SQL gelegentlich als sequel bezeichnet.

Während PHP so entwickelt wurde, dass es die Nutzung von MySQL direkt unterstützt, benötigt JavaScript eine Erweiterung dafür. Eine recht neue Erweiterung heißt **Node.js**. Dieser Ansatz eine Sprache über Erweiterungen zu ergänzen, anstatt diese von vornherein in die Sprache zu integrieren, wirkt auf den ersten Blick komplizierter als die vermeintlich einfache Kombination von PHP und MySQL, aber es ist einfach nur ein Ansatz, der es ermöglicht, die Sprache selbst nicht zu umfangreich werden zu lassen. Was das im Detail bedeutet können Sie in Veranstaltungen zum Software Engineering lernen.

5.5.1 MVC – Das Model View Controller Pattern

Diese Aufteilung der Programmierung einer Webanwendung entspricht einem Entwurfs- und Architekturmuster, das auch in der Wirtschaft angewendet wird. Dieses Muster wird als MVC (kurz für Model View Controller) bezeichnet. Pattern ist schlicht das englische Wort für Muster.

Solche Modelle spielen bei der Entwicklung großer Softwareprojekte eine essentielle Rolle, denn indem wir sie nicht wie einen Monolithen, sondern als Kombination von einzelnen Komponenten entwickeln, können wir leichter Fehlerkorrekturen durchführen und Änderungen einbauen. Außerdem können wir so die Arbeit leicht innerhalb eines Teams mit mehreren Dutzend Entwicklern verteilen.

Allerdings kommt es nur sehr selten vor, dass für jeden Teil eines Patterns auch eine eigenständige Programmiersprache genutzt wird. Das macht es dann gerade für Einsteiger schwer, zu verstehen, wo die Grenze zwischen Model und View und Controller liegt. Das ist bei der Kombination aus HTML5, CSS und PHP oder JavaScript anders: Hier können wir jedem Bestandteil des Patterns genau eine Sprache zuordnen, so wie das oben schon passiert ist. Die SQ-Sprache lassen wir hier mal außen vor; sie dient ja nur dazu, die langfristige Speicherung von Daten zu ermöglichen.

In unserem Fall ist HTML5 die Sprache für das Model, CSS ist die Sprache für den View und PHP bzw. JavaScript ist die Sprache für den Controller. Deshalb werden wir uns im Kapitel über HTML auch nur darum kümmern festzulegen, aus welchen Teilen wir eine Webanwendung zusammensetzen können. Aus dem gleichen Grund werden wir uns im Kapitel über CSS ausschließlich darum kümmern, wie die Elemente der Seite dargestellt werden sollen. Und im Kapitel über PHP bzw. JavaScript wird es dann ausschließlich darum gehen, wie wir Änderungen steuern können, die während der Nutzung der Webanwendung auftreten.

Diejenigen von Ihnen, die Media Systems studieren werden später in der Veranstaltung Software Engineering eine Vielzahl von Patterns kennen lernen. Leider bleibt dort kaum Zeit, sich einem dieser Design Patterns ausführlich zu widmen. Aber so haben Sie jetzt schon die Möglichkeit, den Nutzen eines solchen Patterns kennen zu lernen.

Kontrolle

- Sie wissen, dass Sie bei der Programmierung von Webanwendung vier Arten von Programmiersprachen nutzen, und dass Sie damit ein Design Pattern der Informatik umsetzen, den MVC:
 - Markup Languages dienen dazu, Container zu definieren, die die Inhalte Ihrer Webanwendung enthalten oder die auf Dateien verweisen, die als Teil einer Webanwendung angezeigt werden.
 - CSS dient dazu, die Darstellung der Inhalte zu gestalten.
 - Sprachen wie JavaScript und PHP werden dazu genutzt, um dynamische Webanwendung bzw. Webanwendungen zu realisieren.
 - SQL-Sprachen dienen dann dazu, um z.B. Nutzereingaben dauerhaft zu speichern, um Kataloge von Waren bereitzustellen oder andere große Mengen an Daten aufzubewahren.
- Ihnen ist klar, dass Sie damit wesentlich mehr realisieren können als „nur“ Webpages, auch wenn Sie noch nicht wissen, wie Sie das tun können. Ausblick

Jetzt werden Sie den nächsten Schritt der Web-Evolution kennen lernen: Das semantische Web.

5.6 Das semantische Web

Bis hierher haben Sie nur über Dinge gelesen, die Sie unter Umständen schon wussten. Doch nun kommen wir zu einem Thema, dass den meisten Menschen und leider auch vielen Webentwicklern nicht bekannt ist, obwohl es der nächste Schritt für die Nutzung des WWW ist. Die Rede ist vom semantischen Web. Ohne das Verständnis, was das semantische Web ist, können Sie mit HTML5, dem neuen Standard für Webanwendung nichts anfangen. Sehen Sie sich dazu bitte die folgenden Einleitung an: <https://plus.google.com/+ManuSporny/posts/FPwMGWhgQYh?cfem=1>
Sehen wir uns das mal im Detail an:

5.6.1 Syntax und Semantik

Den einen dieser Begriffe haben Sie wahrscheinlich in der Schule kennen und haben gelernt. Dabei stehen die beiden für ein ausgesprochen sinnvolles Konzept. Mit **Syntax** werden all die Aspekte einer Sprache bezeichnet, bei denen es darum geht, welche Buchstaben und andere Zeichen in welcher Reihenfolge notiert werden dürfen. (Denken Sie an so etwas wie Satzbau, Kommasetzung, usw.) Wenn also eine Sprache ohne Syntax auskommen könnte, gäbe es keine Absprache darüber, welche Wörter es gäbe und wie ein Satz aussehen kann. Und egal ob Sie diesen Stil nun mögen oder nicht, auch der folgende Satz folgt einer Syntax: „Ey Alter, was geht’n?“

Damit kommen wir zur **Semantik**. Mit Semantik bezeichnen wir alles, was uns sagt, welche Bedeutung etwas in einer Sprache hat. Es ist wichtig zu verstehen, dass zwei syntaktisch unterschiedliche Sätze in semantischer Hinsicht gleich sein können. So gibt es einen großen syntaktischen Unterschied zwischen „Ey Alter, was geht’n?“ und „Hallo Herr Schulz, wollen wir Essen gehen?“ aber semantisch ist dieser Unterschied eher gering. Das gilt genauso für Programmiersprachen: Vieles lässt sich in gänzlich unterschiedlicher syntaktischer Form programmieren und erfüllt doch die selbe Aufgabe.

Das führt uns auch nochmal zu der Antwort auf die Frage, warum bei es bei den Projekten der MS-Studierenden keine feste Vorgabe für den Umfang gibt: Da mit extrem unterschiedlichem Code identische Funktionalitäten zu realisieren sind und es nicht um die Form, sondern um die Funktion geht, sind Sie in der Wahl der Form recht frei. Da aber auch die Funktion von Element zu Element unterschiedlich ist, sind Sie auch in der Wahl der Formen recht frei. Nur die Programmiersprachen und die zu verwendende Version sind vorgegeben.

Ein fähiger Softwareentwickler ist somit wie ein Diplomat oder ein Dichter: Er kann mit Sprachen umgehen wie ein Künstler. Und das nicht, weil er ihre Regeln gelernt hat, sondern weil er sie nutzt, um Ideen und Konzepte in eleganter Form auszudrücken. Deshalb ist der erste Schritt zu einem guten Programm auch nicht der Griff zur Tastatur, sondern zu Stift und Papier, um Konzepte und Relationen zu skizzieren, sie zu überarbeiten und ein sinnvolles Ganzes daraus zu gestalten. Erst wenn dieser Schritt abgeschlossen ist, beginnt die eigentliche Programmierung.

Wieder zurück zum Begriff der Semantik: Es gibt einen sehr großen Unterschied zwischen der Semantik bei gesprochenen Sprachen und bei Programmiersprachen: Die Semantik einer Zeile einer Programmiersprache ist immer eindeutig: Sie besagt je nach gewähltem Paradigma entweder was der Computer tun soll oder wie er es tun soll. Bis auf HTML5 gibt es aber noch keine Programmiersprache, die auch aussagt, was das ganze im Sinne der Kommunikation von Menschen bedeutet!

Stellen Sie sich dazu die folgende Situation vor, die Schultz von Thun in seinem Standardwerk über zwischenmenschliche Kommunikation präsentiert: Ein Mann sitzt am Steuer seines Wagens und seine Frau sagt zu ihm: „Es ist grün.“ Dieser Satz lässt sich unterschiedlich interpretieren. Das ist das Problem mit der Semantik in gesprochenen Sprachen: Sie ist im Regelfall nicht eindeutig.

Die meisten Anfänger einer Programmiersprache verstehen deshalb nicht, dass es bei Programmiersprachen keine unterschiedliche Interpretation eines Programms gibt. Wenn es also bei einer Programmiersprache das Äquivalent zu „Es ist grün.“ gibt, dann hat dieser Programmbefehl genau eine Bedeutung. Daraus folgt auch, dass Informatiker in aller Regel versuchen, Dinge so eindeutig wie möglich zu formulieren. Es geht ihnen nicht darum, Dinge zu verkomplizieren, sondern darum, Missverständnisse zu vermeiden.

Woran Sie erkennen können, dass selbst erfahrene Programmierer häufig ignorieren, dass die Semantik von Programmiersprachen eindeutig ist? Ganz einfach: Immer wenn Sie den Satz „Warum macht der das denn nicht?!“ hören, ignoriert jemand diese fundamentale Tatsache. Aber da Programmierer eben auch nur Menschen sind, ist das vollkommen in Ordnung.

Aber was hat das mit dem WWW zu tun? Und wie passt hier der Begriff semantic web ins Bild? Dazu eine kleine Übung:

Aufgabe

- Suchen Sie im Internet (per google.de und duckduckgo.com) nach dem Wort Musik. Vergleichen Sie die Ausgaben der beiden Suchmaschinen.
- Überlegen Sie, warum Menschen mit den Ergebnissen dieser Suche unzufrieden sein könnten, egal welche Suchmaschine sie nutzen.
- Überlegen Sie, was Sie eingeben müssten, damit eine Suchmaschine Ihnen Angebote für CDs Ihrer Lieblingsband anzeigt.

Kontrolle

Sie haben eine erste Idee davon, dass das semantic Web etwas damit zu tun, welche Bedeutung Wörter und multimediale Inhalte im WWW haben.

5.6.2 Semantik und das WWW

Wichtig: Oben haben Sie eine Bedeutung des Wortes Semantik erfahren: Es ist die Bedeutung eines Programmbefehls, bzw. eines Teils eines Computerprogramms. Beim semantic Web geht es aber nicht darum, wie ein Computerprogramm auszuführen ist. Wenn Sie das verwirrt, denken Sie bitte daran, dass der Begriff der Semantik allgemein als „Bedeutung und Interpretation von etwas“ übersetzt werden kann. Wenn wir also vom semantic Web reden, dann geht es um die Bedeutung von etwas, das mit dem WWW zu tun hat, bzw. Teil des WWW ist.

Zurück zum eigentlichen Text:

Die Aufgabe oben war schwierig, aber durch die einleitenden Erklärungen zum WWW hatten Sie das Wissen, um die Ursache des Problems zu erkennen. Dort haben Sie erfahren, dass Webanwendungen in Markup Languages programmiert werden. Sie haben dort ebenfalls erfahren, dass Sie in diesen Sprachen lediglich Container definieren können. Und was ist das Problem mit Containern? Genau: Die meisten Menschen sehen nur die Verpackung aber nicht den Inhalt. Und das gleiche gilt für Container einer Markup Language: Suchmaschinen tun sich schwer damit, die Webanwendung zu finden, nach denen ein Nutzer sucht, weil sie keine semantischen, sondern ausschließlich syntaktische Informationen bekommen, wenn ein Benutzer eine Eingabe macht. Und umgekehrt stellen Webanwendung ebenfalls keine semantischen, sondern ausschließlich syntaktische Informationen zur Verfügung.

Es hat seit der Einführung von HTML4.01 mehrere Versuche gegeben, dieses Problem zu lösen. Ein Lösungsansatz besteht darin, sogenannte Labels zu programmieren. Damit befestigt der Entwickler einer Webanwendung

gewissermaßen Schilden an die Container. Aber auch das ist ja wieder nur eine syntaktische Information, also löst es das Problem nicht.

Kontrolle

Wenn Computer Dateien in einer ML per HTTP austauschen, dann haben Sie nicht die geringste Ahnung, womit sie es zu tun haben. Also müssen wir einen Standard haben, mit dem wir semantische Informationen in ML-Dateien unterbringen können. Denn nur dann können die Computer nach diesen Informationen suchen.

5.6.3 DOM und Microdata

Wenn Sie sich schon einmal mit der Entwicklung von Webanwendung beschäftigt haben, dann haben Sie vielleicht schon die Abkürzung DOM gesehen. Das Document Object Model (kurz DOM) ist die Basis für die Nutzung von JavaScript im Browser. Es geht hier schlicht darum, dass jeder Container einer ML von der Sprache JavaScript als ein eigenständiges Objekt behandelt werden kann. Die Sprache kann dann den Inhalt eines solchen Containers ändern. Über DOMs kann außerdem definiert werden, wie verschiedene Container voneinander abhängen. All das hilft uns aber bei der Suche nach einem semantic Web nicht weiter, weil es immer noch keine semantischen Informationen bereitstellt, sondern nur klärt, wie die Elemente einer Webanwendung voneinander abhängen.

Microdata sind nun gewissermaßen eine Erweiterung von DOMs, die mit HTML5 eingeführt wurden: Hier enthält ein Container Einträge, die in standardisierter Form angeben, welche Bedeutung ein Eintrag in einem Container hat. Wenn Sie also Microdata verwenden, dann bedeutet das, dass Sie eine Anschrift in HTML5 so einprogrammieren, dass der Browser erkennen kann, dass es eine Anschrift ist, welcher Teil die Straße ist, welche Textpassage der Name ist, usw. Ein Besucher dieser Webanwendung braucht dann nichts weiter zu tun, wenn er diese Anschrift in sein Adressbuch übernehmen will oder die Anschrift in einem Kartenprogramm suchen will: Durch die Microdata kann der Browser die entsprechenden Verlinkungen erzeugen und die nötigen Daten weitergeben. Mit Microdata erhalten wir also eine semantische Webanwendung.

Bitte denken sie nicht, dass das nur für Anschriften gilt. Auf der Seite <http://www.schema.org/docs/full.html> finden Sie einen Überblick über alle Arten von Microdata, die Sie verwenden können. Bitte versuchen Sie nicht, all diese Typen auswendig zu lernen; wie so oft in der Informatik ist auswendig lernen unsinnig, da kontinuierlich Änderungen erfolgen und es viel wichtiger ist, zu wissen, wo Sie etwas finden, als dass

Sie wissen, wie es im Detail aussieht.

Neben naheliegenden Inhalten wie Anschriften finden Sie hier auch Microdata für Veranstaltungen. Richtig gelesen: Wenn Sie innerhalb eines Containers einen Eintrag als solch einen Typ von Microdata programmieren, dann brauchen Sie auf der ganzen Seite nicht ein einziges Mal den Begriff *Veranstaltung* zu benutzen: Eine Suchmaschine kann alleine durch diese Typangabe erkennen, dass es sich um eine Veranstaltung handelt und nicht etwa um einen Leitfaden für die Planung und Durchführung einer solchen.

Neben den negativen Möglichkeiten sorgt das semantic Web auch dafür, dass Manipulationen durch Betreiber von Preisportalen und ähnlichen Webanwendung in Zukunft weniger erfolgreich sein werden: Da bei konsequenter Anwendung von HTML5 und Microdata eine Suchmaschine direkt erkennt, wo eine Seite ein Angebot z.B. für Flugreisen anbietet, werden Nutzer langfristig eine eigene Suche nach Flugreisen durchführen können. Deshalb werden dürften diese Portale mittelfristig wieder vom Markt verschwinden.

Und das Großartige bei all dem ist, dass es wie beim WWW kein gewinnorientierte Unternehmen gibt, dass für die Nutzung von HTML5 Geld verlangt.

Kontrolle

- Sie wissen jetzt, dass Sie in HTML5 eine fast schon unüberschaubare Menge an Informationen so verpacken können, dass Browser wissen, um was für Daten es sich handelt.
- Sie haben eine erste Vorstellung davon, wie umfangreich die Auswirkung des semantic Web ist.

5.6.4 Zusammenfassung

Sie haben jetzt einen ersten Überblick über Techniken und Technologien, die Sie nutzen können, um Webanwendungen entwickeln können. Wenn Sie dieser Einleitung aufmerksam gefolgt sind, dann haben Sie außerdem verstanden, dass es bei der Entwicklung einer Anwendung, eines Computerprogramms oder eine verteilten Anwendung nicht zuerst darauf ankommt, welche Programmiersprache Sie nutzen wollen, sondern welche Konzepte und Ideen Sie umsetzen wollen.

5.7 Übertragung der Dateien auf einen Webserver im Netz

Wenn Sie im Gegensatz zum hier beschriebenen Vorgehen bereits einen Webserver nutzen, der Ihre Webanwendung im Netz anbietet und über das Internet erreichbar ist, dann müssen Sie noch wissen, was für ein Programm Sie zur Datenübertragung nutzen können.

Hierzu müssen Sie zunächst wissen, was ein **Protokoll** ist und welche Protokolle es gibt.

Protokolle sind Vereinbarungen darüber, wie bestimmte Abläufe geregelt werden. Klingt kompliziert? Ist es nicht. Ein praktisches Protokoll ist die Begrüßung: Es kann sein, dass Sie jemandem zur Begrüßung die Hand schütteln, ihr ein einfaches Hallo sagen, oder wie auch immer Sie die Begrüßung handhaben. Wenn Sie mit einer anderen Person eine feste Vereinbarung treffen sollten, wie die Begrüßung ablaufen soll, dann entspricht das dem, was wir als Protokoll in einem Netzwerk bezeichnen. Sie habens noch nicht verstanden? Dann sehen Sie ein paar Folgen Star Trek.

Wie Sie bereits wissen, gibt es verschiedene Arten, wie man Daten darstellen kann. Zur Erinnerung: In der **Nachrichtentechnik** wird das über die sogenannte **Codierung** festgelegt. In der Informatik werden üblicherweise nur Codes genutzt, die sich als Codetabelle darstellen lassen oder die mittels einer mathematischen Funktion erfolgen. Vereinfacht ausgedrückt definiert ein Protokoll unter anderem, mittels welcher Codierung Daten zwischen Server und Client ausgetauscht werden.

Ein Protokoll bekommen Sie fast immer angezeigt, wenn Sie im Netz unterwegs sind. Es handelt sich um das **HTTP**, das Hyper Text Transfer Protokoll. Übersetzt ist das also ein Protokoll, das festlegt, wie Hypertexte (was auch immer das sein mag) von einem Client angefordert und an ihn übertragen werden.

Nun also zum Begriff **Hypertext**: Wie Sie wissen werden in Webanwendungen Inhalte angezeigt, von denen aus Sie über Links (Fachbegriff **Hyperlinks**) zu anderen Webanwendungen bzw. zu anderen Inhalten gelangen. Um zu betonen, dass ein Dokument aus Texten und aus Verbindungen zwischen entfernten Textpassagen besteht, wurde der Begriff des Hypertexts entwickelt. (Das ist heute eine Selbstverständlichkeit, aber versuchen Sie mal einen Link jemandem zu erklären, der bislang nur Bücher aber keine Webpages kennt. Denn bei der Nutzung eines Links springen Sie ja direkt an eine bestimmte Stelle, ohne erst nachschlagen und nach der entspre-

chenden Stelle im Buch suchen zu müssen.)

Nun aber zu zwei Protokollen, die Sie für die Datenübertragung auf einen Webserver kennen müssen: Das FTP und den SSL. Wenn Sie konzentriert gelesen haben, dann haben Sie bemerkt, dass HTTP es nicht ermöglicht, ganze Dateien beliebiger Dateiformate zu übertragen, sondern dass es nur dazu dient, die Inhalte von Webanwendungen anzufordern und zu übertragen.

Das **FTP**, kurz für File-Transfer-Protokoll löst nun genau diese Aufgabe: Es definiert, wie ein Client Dateien auf einen Server übertragen kann. Aus Sicherheitsgründen sollten Sie jedoch prüfen, ob dieses Protokoll für Ihre Zwecke ausreichend ist. Denn eine Datenübertragung per FTP ist öffentlich. Das bedeutet in Kurzform: Alle angeschlossenen Nutzer können sämtliche Daten kopieren, die Sie über ein Netzwerk übertragen. Ja! Natürlich gilt das auch für Ihre Passwörter. Und das gleiche Prinzip gilt auch für HTTP. Denn diese Dienste wurden entwickelt, um bestimmte Daten mit einer möglichst hohen Effizienz über Netzwerke zu übertragen und jede Form von Sicherheit reduziert im Regelfall die Effizienz einer Übertragung. In den Äußerungen mancher politisch interessierten Personen zeigt sich hier das mangelnde Verständnis für die Technologie: Denn das hat nichts mit einem Wunsch nach allumfassender Überwachung zu tun, das hat etwas mit der Anwendung von Naturgesetzen zu tun.

Wollen Sie dagegen weitestgehend sicherstellen, dass niemand die übertragenen Dateien ändern oder auch nur kopieren kann, dann sollten Sie eine Verbindung mittels **SSL**, kurz für Secure-Socket-Link wählen. Allerdings kann es sein, dass der Webserver diese Art des Datentransfers nicht unterstützt. Gerade bei Billigangeboten wird SSL-Unterstützung gerne als kostenpflichtiges Update verkauft. Ob SSL nun ein eigenes Protokoll oder die Erweiterung eines Protokolls ist, lassen wir hier einmal außen vor.

Ähnlich wie SSL bietet auch **HTTPS** eine sicherere Möglichkeit, um im Netz Daten zu übertragen. Beachten Sie aber bitte: Eine absolute Sicherheit gibt es auch in Netzwerken nicht. Der Sinn von Netzwerken bestand sehr lange auch ausschließlich darin, Verbindungen zu ermöglichen, nicht darin, die Vertraulichkeit der übertragenen Daten sicherzustellen. Über Details zu diesen Themen hören Sie mehr in der Veranstaltung Netzwerke und Internetsicherheit.

Kontrolle

Sie wissen jetzt, dass Sie für die Übertragung von Dateien auf einen Webserver ein FTP- oder ein SSL-Programm benötigen. Sie wissen auch, dass Sie

im Netz nicht nach FTP-Programm oder SSL-Programm suchen, weil Sie wissen, dass es für solche Programme eine andere Bezeichnung als das Wort Programm gibt.

Wie Sie ein solches Programm bedienen müssen ist nicht Teil dieser Veranstaltung, da auch hier erneut weitergehende Kenntnisse bei der Administration von Servern nötig sind, die mit der Entwicklung einer Webanwendung bzw. den Grundlagen der Programmierung nichts zu tun haben.

Kapitel 6

Einstieg in HTML 5

Dass Sie in diesem Kapitel die Grundlagen der Programmierung in HTML5 erlernen ist nach der Kapitelüberschrift klar. Aber Sie werden hier ebenfalls Konzepte kennen lernen, die wichtig sind, damit Ihre Seite tatsächlich ein Teil des semantic web ist. Dieses Kapitel ist auch für komplette Neueinsteiger in die Programmierung leicht zu bearbeiten, da Sie hier nur wenige abstrakte Grundlagen verinnerlichen müssen, die sonst in der Welt der Informatik und der Programmierung recht häufig vorkommen.

Wenn Sie mit Dateien arbeiten, sind Sie es in aller Regel gewöhnt, die Datei mit einem Programm zu öffnen. Die meisten von Ihnen kennen also nur die verschiedenen Ansichten, die einzelne Programme erzeugen, wenn Sie mit einem dieser Programme eine Datei öffnen. Bei einer Webanwendung sehen Sie beispielsweise verschiedene Elemente wie Bilder, Videos, Texte usw. Sie sehen dann aber in aller Regel nicht, wie diese Datei selbst aussieht. Doch genau darum geht es in diesem Kurs.

Wenn Sie das jetzt irritiert, dann machen Sie sich bitte folgendes klar: Wenn Sie eine pdf-Datei öffnen, dann generiert der pdf-Reader eine Ansicht. Das was Sie sehen ist also nicht die Datei selbst, sondern nur eine aus dieser Datei erzeugte Ansicht. Und alles, was Sie irgendwo von einem Computer angezeigt bekommen ist eine Interpretation einer oder mehrerer Dateien. Wenn Sie also lernen wollen, wie Sie Computer programmieren können, dann müssen Sie als erstes verstehen, dass Sie bislang immer nur eine Interpretation dessen gesehen (oder gehört) haben, was tatsächlich „auf dem“ Computer gespeichert ist.

In HTML kümmern wir uns dagegen nicht darum, wie eine Webanwendung aussehen soll, sondern ausschließlich darum, aus welchen Bestandteilen die Webanwendung bestehen soll. In anderen Worten: In diesem ganzen Kapitel wird es nicht ein einziges Mal darum gehen, wie die Elemente

Ihrer Webanwendung später aussehen werden. Wenn Sie hier also an irgend einer Stelle versuchen, die Darstellung zu programmieren, dann machen Sie einen Fehler.

Nachdem wir zuvor weitgehend geklärt haben, wie wir eine Webserver auf unserem Rechner in Betrieb setzen und kontrollieren können, kommen wir nun zum zweiten Teil dieser Einführung: Die Einführung in die Markup Language HTML. Am Ende dieses Abschnittes können Sie also statische Webanwendung erstellen. Danach werden wir über **CSS** sprechen: Cascading Style Sheets sind eine Möglichkeit, um auf unterschiedlichen Webanwendung gleiche Strukturen und Formate für Inhalte zu realisieren. Danach kommen wir zur Programmierung in PHP, womit Sie die Elemente Ihrer Webanwendung dynamisch und interaktiv programmieren können.

Aufgabe:

Unabhängig von dem, was in den einzelnen Abschnitten steht, programmieren Sie bitte für jedes Element, dass Sie auf Ihrer Webanwendung einbinden wollen einen eigenen HTML-Container, ohne sich über das Design Gedanken zu machen. Das ist der erste Schritt hin zu professionellen Anwendungen: So lange Sie es nicht schaffen, zuerst die Inhalte bzw. Inhaltsstruktur festzulegen und erst dann ein passendes Design auszusuchen sind Sie weit davon entfernt, professionelle/r InformatikerIn zu sein. Das bedeutet nicht, dass Design unwichtig wäre, sondern es geht darum, dass Sie sich auf Ihre zukünftige Aufgabe in Teams konzentrieren. Und so lange Sie nicht Design (z.B. Medien- und Kommunikationsdesign) studieren, bereiten Sie sich eben auch nicht darauf vor, als DesignerIn zu arbeiten. So lange Sie das nicht akzeptieren werden Sie ein Don Quichote der Medieninformatik bzw. der Medientechnik sein.

Ignorieren Sie also Aspekte wie Design, die Positionierung eines Elements auf der Seite, Hyperlinks, usw. usf. Auch für die einzusetzenden Text, Bilder usw. verwenden Sie bitte vorerst Platzhalter: Die Qualität Ihrer Arbeit als MedieninformatikerIn wird sich nicht in umfangreichen Texten und Bildergalerien zeigen, sondern in gut durchdachten und ausgearbeiteten Strukturen. Das Einfügen von Texten und Bildern sowie das Ausarbeiten eines guten Designs ist dann die Aufgabe anderer Mitarbeiter im Team.

6.0.1 Das ist HTML

HTML, kurz für Hyper Text Markup Language ist, wie es aus dem Namen hervorgeht eine **Markup Language**. Teilweise wird auch von einer statischen Skriptsprache gesprochen. Statisch bedeutet hier, dass es mit HTML nicht möglich ist, Inhalte während der Nutzung zu ändern. In der Anfangs-

zeit des WWW genügte das auch, weil es schon eine großartige Bereicherung darstellte, Texte und Bilder direkt über eine Datenleitung in wenigen Sekunden oder Minuten empfangen zu können, die sonst per Post erst nach einigen Tagen oder Wochen im Haus gewesen wären.

Heute dagegen, wo selbst das Format von Displays kaum noch standardisiert ist, benötigen wir weitergehende Möglichkeiten. Diese werden unter dem Begriff **Responsive Design** zusammengefasst: Das Design einer Webanwendung passt sich automatisch dem Gerät an, auf dem es angezeigt wird. Wir reden hier also über etwas, das in den Bereich der Informatik bzw. der Softwareentwicklung fällt und nicht in den Bereich dessen, was üblicherweise unter Design im Sinne von kreativer Gestaltung verstanden wird.

Deshalb muss es hier nochmal betont werden: Wenn Sie HTML programmieren und festlegen, wie ein Element aussieht oder wo es angeordnet werden soll, dann erzeugen Sie damit in den meisten Fällen eine Webanwendung, die auf einer Vielzahl von Nutzergeräten grausig aussehen wird, egal wie gut Ihr Webdesign sonst sein mag. Also lassen Sie das.

Dieser Kurs behandelt jedoch nicht die Feinheiten der Usability, zu denen Responsive Design gehört. Vielmehr ist das eine der Spezialisierungen, die MedieninformatikerInnen im Masterstudium wählen können.

6.0.2 Erster HTML-Quellcode

Das wichtigste bei der Programmierung einer Webanwendung ist die Antwort auf die Frage, aus welchen Einheiten sie bestehen soll und welche Funktion jede dieser Einheiten übernehmen soll. Erst danach überlegen Sie sich idealerweise unterstützt durch Designer, mittels welches Designs diese Funktion erkennbar sein soll. Ein typischer Einsteigerfehler besteht darin, sich zunächst über das Design Gedanken zu machen und dann mit dem Programmieren anzufangen. Das ist deshalb ein Fehler, weil dabei in aller Regel wichtige Funktionalitäten vergessen werden oder (noch schlimmer) es für Nutzer nicht erkennbar ist, wie eine Funktionalität genutzt werden kann. Wird dann entdeckt, dass eine Funktionalität fehlt, die wichtig ist, dann muss häufig das Design komplett verworfen und neu entwickelt werden. Oder es herrscht die Überzeugung vor, dass der Kunde schon blöd genug sein wird, sich nicht daran zu stören. Und glauben Sie mir: Bei der Prüfung für Ihre Leistungsnachweise haben Sie keinen naiven Kunden vor sich.

Deshalb nun die Frage: Welche Funktion soll unsere erste Webanwendung erfüllen?

Hinweis:

Diese Webanwendung ist ausschließlich für diejenigen gedacht, die zu Hause dieses Buch durcharbeiten wollen. Das gleiche gilt für *Hausaufgaben* und andere Übungen, die Sie hier finden. Es handelt sich hier nicht um Aufgaben, die Sie für einen Leistungsnachweis in Studienveranstaltungen von mir bearbeiten müssen. Vielmehr können Sie sich so unabhängig von meinen Studienveranstaltungen in das Thema einarbeiten.

Beginnen wir mit einer einfachen Seite, mit der wir einfach nur prüfen wollen, ob ein Text so angezeigt wird, dass wir mit der Darstellung zufrieden sind: Er sollte groß genug angezeigt werden, damit wir ihn lesen können. Außerdem sollte er an einer Position angezeigt werden, an der er von einem Nutzer unserer Seite bemerkt wird.

Das ist ziemlich viel Text, um daraus eine Prüfung abzuleiten, ob die Funktionalität erfüllt ist. Deshalb gibt es das Planungswerkzeug **Use Case**. Ein Use Case ist eine kurze Beschreibung, wer was mit einer Webanwendung tut und was dann passieren soll. Machen wir das doch gleich für unseren Fall:

Use Case 1: Der Nutzer liest einen Begrüßungstext. (Keine Interaktion.)

Sicher, das ist kein spannender Use Case, weil er keine Interaktion enthält. Vor allem können wir auch gar nicht prüfen, ob einzelne NutzerInnen tatsächlich lesen, was da angezeigt wird. Aber damit können wir arbeiten und nach der Programmierung prüfen, ob die Funktionalität erfüllt ist. Geben Sie den folgenden Quellcode in einen einfachen Editor (z.B. den kostenlosen notepad++) ein und speichern ihn unter dem Namen `seite01.html` im bekannten Verzeichnis Ihres Webservers.

```
<html>
<head>
<title>Die erste HTML-Webanwendung</title>
</head>
<body>
Einführung in die Programmierung, Teil 1
</body>
</html>
```

Gleich vorweg: Willkommensgrüße oder ähnliches haben auf einer Webanwendung nichts verloren. Entweder der Nutzer kann auf Anhieb erkennen,

was hier angeboten wird oder wir haben als Entwickler etwas falsch gemacht.

Wenn Sie jetzt die Webanwendung aktualisieren (oder bei gestopptem Webserver zunächst den Webserver neustarten und anschließend wieder die Webanwendung localhost aufrufen), sehen Sie, dass das neue HTML-Skript als Datei angezeigt wird. Wählen Sie doch einfach den „Link“ an und öffnen Sie damit die Webanwendung, die Sie gerade programmiert haben.

Es gibt zwei Dinge, die Ihnen auffallen, wenn Sie das Ergebnis im Browser genau prüfen (zumindest wenn Sie keinen Tippfehler im Quellcode haben):

- Zum einen steht da nicht das Wort Einführung im Text, sondern so etwas wie EinfÄ3/4rung.
- Und dann steht auf der Registerlasche der Webanwendung der Schriftzug Die erste HTML-Webanwendung.

Wenn Sie die einleitenden Kapitel aufmerksam gelesen haben, dann haben Sie sicher schon eine Idee, warum hier kein ü in Einführung steht. Aber dazu gleich mehr. Zuvor schauen wir uns zwei Dinge an. Da wäre einmal die Frage, wie wir möglichst effizient programmieren können und dann steht die Frage an, welche Bestandteile unser Quellcode enthält.

Tags und Container

Der Schriftzug Die erste HTML-Webanwendung, der als Titel der Seite im Webbrowser angezeigt wird, steht zwischen den „Befehlen“ `<title>` und `</title>`. Solche „Befehle“ werden in Markup Languages als Tags bezeichnet. (Aussprache wie tägs und nicht wie tags (im Sinne von mit-tags).)

Wenn wir wie bei `<title>` und `</title>` zwei Tags haben, von denen das eine den Anfang und das andere das Ende von einem Teil unserer Webanwendung definiert, dann bezeichnen wir die beiden als **öffnendes** bzw. **schließendes** Tag und beide gemeinsam mit allem, was zwischen ihnen steht als einen **Container**.

Wir haben aber auch Container, die zwischen dem öffnenden und dem schließenden Tag keinen eigentlichen Inhalt haben. Um zu verdeutlichen, dass ein Container keinen solchen Inhalt hat, können wir ein öffnendes Tag auch direkt wieder schließen und brauchen kein schließendes Tag zu programmieren. Dazu schreiben wir als letztes Zeichen in einen öffnenden Tag

das \-Symbol.

Kontrolle:

- Wie gesagt programmieren Sie in einer Markup Language ausschließlich die Struktur von Anwendungen und weder ihre Funktionalität noch ihre Gestaltung. In sofern war HTML4.01 keine reine Markup Language, sondern eine Mischung aus Markup und Design Sprache.
- Sie wissen jetzt, dass die Elemente, aus denen Sie eine Struktur in HTML programmieren Container heißen und dass jeder Container aus einem in sich abgeschlossenen Tag oder aus einem öffnenden und einem schließenden Tag besteht.

Dokumentteile auslagern

Mit HTML-Dokumenten definieren Sie, aus welchen Bestandteilen eine Webanwendung besteht. Sie besteht dagegen nicht aus den konkreten Inhalten. Texte, Bilder, Sound und Videos sind nicht Teil von HTML. Allerdings können wir Texte durchaus direkt ins HTML einfügen, so wie wir das oben getan haben. Das ist allerdings eine Vorgehensweise, die bei größeren Projekten nicht empfehlenswert ist. Vielmehr sollten Sie auch Texte in eigenständigen Dateien speichern und sie durch den Webserver ins HTML einfügen lassen. Die einfachste Möglichkeit dafür ist eine PHP-Funktion, die Sie in das HTML-Dokument einfügen.

1. Erstellen Sie dafür eine Datei mit dem Namen `test_title_001.txt`, in der Sie die folgende Zeile eintragen und die Sie dann im gleichen Verzeichnis wie `seite01.html` abspeichern:

```
echo("Die erste HTML-Webanwendung");
```

2. Erstellen Sie jetzt eine Datei mit dem Namen `test_begrueessung_001.txt` mit dem Inhalt `echo("Einführung in die Programmierung, Teil 1");`.
3. Ändern Sie die Datei `seite01.html` wie folgte ab und speichern Sie sie unter dem Namen `seite01.php`:

```
<html>
<head>
<title><?php include(test_title_001.txt); ?>
</title>
</head>
```



```
<body>
<?php include(test_begruessung_001.txt); ?>
</body>
</html>
```

4. Rufen Sie jetzt im Browser `localhost` auf und wählen Sie dort `seite01.php` an.

Wie Sie sehen sieht die Webpage genauso aus, als wenn Sie `seite01.html` geöffnet hätten. Schauen wir uns die Änderungen und Ihre Auswirkungen einmal im Detail an:

- Die Funktion `echo()` ist eine Funktion der Programmiersprache PHP.
- Alles, was Sie zwischen zwei Anführungszeichen in die Klammer von `echo()` schreiben wird ausgegeben.
- Eine Programmzeile in PHP wird durch ein Semikolon beendet.
- Die PHP-Funktion `include()` lädt den Inhalt einer anderen Datei und fügt ihn an der Stelle ein, wo die `include()`-Funktion steht.
 - Sie können also mit `include()` beliebige Teile eines PHP-Programms in eigenen Dateien speichern.
 - Das besondere dabei ist, dass Sie somit jeden Teil eines PHP-Programms, der mehrfach genutzt werden soll nur einmal programmieren müssen. (Wenn Sie die objektorientierte Programmierung kennen lernen werden Sie dafür eine andere Möglichkeit kennen lernen.)
- Um PHP-Programme oder Programmteile in ein HTML-Dokument einzufügen müssen Sie es in ein Tag eintragen, das mit `<?php` beginnt und mit `?>` endet.
- Das Großartige dabei ist, dass Sie auf diese Weise gleich zwei Fliegen mit einer Klappe schlagen:
 1. Sie können beliebigen PHP-Code ohne weitere Vorbereitung direkt in HTML einfügen.
 2. Sie können über den oben gezeigten Weg beliebigen HTML-Code aus anderen Dateien an beliebigen Stellen in verschiedene HTML-Dokumente einfügen.

Ausblick für Fortgeschrittene:

- Um Programme oder Programmteile in ein HTML-Dokument zu integrieren, die in JavaScript programmiert wurden, nutzen Sie das `script`-Tag. Wenn Sie beispielsweise die JavaScript-Datei `intro.js` an einer Stelle Ihres HTML-Dokuments einzufügen, lautet das HTML-Tag `<script src=intro.js>`. In HTML4.01 musste deutlich mehr programmiert werden, um JavaScript-Code in HTML einzufügen.
- Wollen Sie dagegen JavaScript direkt in HTML programmieren, dann sieht das so aus: `<script> ... Hier steht der JavaScript-Code ... </script>`. Auch hier gilt wieder: In HTML4.01 musste deutlich mehr programmiert werden, um JavaScript-Code in HTML einzufügen.
- **Hinweis:** Diejenigen von Ihnen, die bei mir den Leistungsnachweis „Projekt 1 (MS)“ erwerben wollen sollten sich das genau merken: Ich brauche in aller Regel nur zwei Sekunden, um zu erkennen, ob Sie HTML4.01 oder HTML5 programmieren. Für den Leistungsnachweis gilt: Die Nutzung von HTML4.01 bedeutet automatisch, dass Sie (noch) nicht bestanden haben.

6.0.3 Struktur eines HTML-Dokuments

Damit bleibt die Frage, was die HTML-Tags eigentlich bewirken oder besser gesagt, wie sie von einem Browser interpretiert werden, um daraus die Ansicht zu generieren, die wir als NutzerInnen angezeigt bekommen.

- Zu Beginn sehen Sie dort den öffnenden Container `<html>`. Dieser gibt an, dass alles nachfolgende HTML-Code ist. In der letzten Zeile finden Sie dann das schließende Tag `</html>`. Dieses besagt also schlicht, dass jetzt kein HTML-Code mehr folgt.
- Danach folgt der `head`-Container. Wie Sie sehen befindet sich innerhalb dieses Containers der `title`-Container, der definiert, unter welchem Titel die Webanwendung angezeigt wird. Hier werden Sie später auch allgemeine Formatierungen für eine einzelne Webanwendung programmieren. Grundsätzlich gilt, dass wir nicht unbedingt einen `<head>`-Container programmieren müssen. Aber es macht nur selten Sinn, ihn vollständig wegzulassen.
- Danach folgt der `body`-Container. Dieser beinhaltet all das, was einer Ansicht der Webanwendung angezeigt wird.

Anmerkung:

Bei HTML wird häufig der Begriff HTML-Skript verwendet, weil es sich ja um kein Programm im dem Sinne handelt, dass hier ein Algorithmus umgesetzt wird. Das ist aber keine allgemeingültige Definition; Sie können also ruhig von einem HTML-Programm oder einer HTML-Seite sprechen, wenn Sie sich damit wohler fühlen. Der von mir angesprochene Unterschied zwischen Skript und Programm wird Ihnen spätestens dann deutlich, wenn Sie mit der imperativen Programmierung in PHP beginnen.

Kontrolle:

Sie wissen jetzt, dass jedes HTML-Skript drei Container beinhalten sollte, den html-, den head- und den body-Container. Sie haben richtig gelesen: Es sollte diese Container beinhalten, muss es aber nicht. Das HTML-Skript ließe sich also auch mit den folgenden Zeilen programmieren:

```
<title>Die erste HTML-Webanwendung</title>
Willkommen zur Einführung in die Programmierung.
```

Aus Gründen der Lesbarkeit sollten Sie dennoch immer die etwas umfangreichere Struktur mit html-, head- und body-Container verwenden.

HTML-Referenz: W3Schools

Bis Anfang 2015 galt **SelfHTML** als die Standardreferenz für HTML. Leider kann ich hiervon mittlerweile nur noch abraten: SelfHTML ignoriert wie die meisten HTML 4.01-Veteranen alles, was HTML 5 zu einer echten nächsten Version von HTML macht. Wenn Sie sich an SelfHTML orientieren, dann können Sie auch gleich bei HTML 4.01 bleiben.

Die einzige mir bekannte Webpage, bei der eine klare Trennung zwischen HTML 4.01 und 5 durchgeführt wird ist zum Jahreswechsel 2015/16 (neben dem W3 Consortium, das die Standards festlegt) die Seite **W3Schools** (<http://www.w3schools.com/default.asp>).

Übersicht über alle Tags Wir werden in diesem Kurs nur einen kleinen Teil aller Tags besprechen, die es gibt. Und wir werden hier auch jeweils nur einige wenige Anwendungen besprechen können. Damit Sie aber sinnvolle Webanwendung und Webanwendungen entwickeln können, müssen Sie wissen, wo Sie nachschlagen können, wenn Sie mehr Informationen zu einzelnen Tags suchen. Bei den W3Schools nutzen Sie dazu die folgende

Unterseite: <http://www.w3schools.com/tags/>

Hier finden Sie eine vollständige Übersicht über die Tags in HTML. Außerdem finden Sie hier Hinweise darauf, was sich zwischen HTML4.01 und HTML5 geändert hat. Erfahrene HTML-Programmierer werden hier überrascht: Zwar sind alle Bereiche aus HTML entfallen, die mit dem Layout bzw. der Gestaltung zusammen hängen, aber dafür gibt es zum Teil Elemente aus HTML3, die in HTML4 entfernt und in HTML5 wieder hinzugefügt wurden.

Wichtige Hinweise:

- Auf www.w3schools.com finden Sie zu jedem Tag Hinweise darauf, welcher Browser das jeweilige Tag unterstützt. Die Angaben dort werden jedoch nicht aktualisiert, sodass der Eindruck entstehen könnte, dass Sie HTML5 praktisch kaum einsetzen können. Das trifft aber nicht zu. Welche Tags tatsächlich von welchem Browser nicht automatisch unterstützt werden und wie Sie dafür sorgen können, dass ein Tag dennoch richtig angezeigt wird, erfahren Sie im Abschnitt zu den sogenannten Polyfills.
- Für diejenigen, die bereits in HTML4.01 programmiert haben hier noch ein essentieller Hinweis: Einige Container wie z.B. `<div>`, die bislang sehr oft verwendet wurden, gibt es weiterhin, aber sie sind nur noch in den seltenen Spezialfällen einzusetzen, die mit den neuen Tags in HTML5 nicht abgedeckt sind.
- Leider wird an dieser Stelle häufig auf [selfhtml](#) verwiesen. Das ist eine Webpage, die sehr detailliert ist, wenn es um HTML4.01 geht. Bezüglich HTML5 ist die Seite leider zurzeit nicht nutzbar. Zu oft werden Sie dort nach dem Lesen eines Abschnitts vor der Frage stehen: Gilt das jetzt immer noch? Deshalb lautet die klare Empfehlung: Nutzen Sie die Seite der W3Schools.

Was haben XML und XHTML mit Webanwendungen zu tun?

Einer der ersten Versuche, um semantische Webanwendungen zu entwickeln wurde unter der Bezeichnung **XML** (kurz für extended markup language) veröffentlicht. Eine Kombination aus XML und HTML wird als **XHTML** bezeichnet.

Validierung - Prüfung auf Fehler

Bei der Eingabe von Programmcode machen wir unvermeidlich Fehler. Sie haben im vorigen Kapitel den Unterschied zwischen Syntax und Semantik

kennen gelernt und wissen deshalb, dass es für einen Computer möglich ist, syntaktische Fehler zu erkennen, aber unmöglich semantische Fehler zu erkennen. Eine weitere Art von Fehlern, die ein Computer nicht erkennen kann hat etwas mit der Logik eines Programms zu tun: Wenn wir in einem Programm etwas einprogrammiert haben, dass der Computer ausführen kann, das aber nicht zu dem Ziel führt, zu dem wir gelangen wollen, dann kann der Computer das mit den bekannten Sprachen nicht erkennen.

Vor diesem Hintergrund ist die Behauptung, dass nur **streng typisierte Sprachen** sicher seien kompletter Humbug: Sie bieten eine minimale Art von Sicherheit, die aber häufig irrelevant ist. Wer das nicht glauben mag, dem möchte ich einmal empfehlen, sich anzusehen, warum der erste Start der *Ariane V* ein Totalausfall war. Was hat all die ach so großartige Typsicherheit beim Start dieser Rakete gebracht? Gar nichts! Ist Typsicherheit deshalb irrelevant? Nein. Sorgt sie dafür, dass Entwickler ein ungerechtes Gefühl von Sicherheit haben? Definitiv. Ist sie somit selbst ein Sicherheitsrisiko? Das können Sie sich selbst beantworten.

Kommen wir nun wieder zurück zur Programmierung von HTML-Dokumenten. Bislang kennen Sie keine Möglichkeit, um Ihr HTML-Dokument auf Fehlerfreiheit zu prüfen. Der Begriff der Fehlerfreiheit ist aber ungenau. Syntaktische Fehler können Sie ja recht leicht durch die Syntaxhervorhebung von `notepad++` erkennen. Bei Markup Languages geht es uns aber um die Struktur einer Anwendung, also wäre es schön, eine Möglichkeit zu haben, um zu prüfen, ob die Struktur zumindest grundsätzlich den Vorgaben für HTML5 entspricht. Und das wird als **Validierung** bezeichnet.

Damit aber ein Browser oder Validator (Programm, das die Validität eines Dokuments prüft) erkennen kann, dass wir ein HTML5-Dokument erstellt haben, müssen wir noch eine Zeile an den Anfang des Dokuments einfügen. Diese Zeile gibt den sogenannten **Doctype** an. Neben der Arbeitserleichterung folgt hieraus auch, dass Sie sich rechtlich etwas absichern: Indem Sie eine validierbare Webanwendung programmieren zeigen Sie, dass Sie nach den aktuellen technischen Standards arbeiten. Und das kann sich im Falle eines Gerichtsverfahrens zu Ihren Gunsten auswirken.

Den **W3C-Validator** finden Sie unter <http://validator.w3.org>. Hier geben Sie den Link auf die zu prüfende Seite ein und erhalten dann eine Ausgabe über die Validität Ihrer Seite. Alles was Sie tun müssen, um ein HTML-Skript als HTML5-Skript zu markieren ist das Hinzufügen einer kurzen Zeile am Anfang des Skripts (noch vor dem `html`-Container):

```
<!doctype html>
```

Und das ist alles. Wenn Sie sich nicht sicher sind, ob Sie wirklich HTML5 programmieren wollen, brauchen Sie sich keine Sorgen zu machen: HTML5-Skripte können Passagen enthalten, die wie bei einer früheren HTML-Version programmiert sind. Diese werden dann wie bei den früheren Versionen ausgeführt.

Wenn wir also unseren bisherigen Code (als HTML5-Code) validierbar machen wollen, dann sieht er so aus:

```
<!doctype html>
<html>
<head>
<title><?php include(test_title_001.txt); ?></title>
</head>
<body>
<?php include(test_begrueessung_001.txt); ?>
</body>
</html>
```

Die erste Programmzeile innerhalb eines HTML-Dokuments gibt somit zwei Dinge an, die gemeinsam als **Doctype Definition** (kurz DTD) bezeichnet werden:

- Dies ist ein HTML-Dokument.
- Die HTML-Version, in der das Dokument erstellt wurde.

Bei HTML 4.01 war diese Zeile recht lang und es gab mehr als nur eine Variante. Hier ein Beispiel für eine DTD in HTML4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN">
```

Da ist die Variante von HTML5 doch deutlich angenehmer, nicht zuletzt weil es nur genau eine Variante gibt.

Wenn Sie Ihr HTML-Dokument mit der DTD gespeichert und den Browser aktualisiert haben, werden Sie feststellen, dass sich die Anzeige nicht geändert hat. Deshalb hier nochmals der Hinweis: Diese Programmzeile teilt dem Webbrowser lediglich mit, dass es sich hier um eine HTML5-Seite handelt. Da es für NutzerInnen einer Webanwendung üblicherweise belanglos ist, wie diese Seite programmiert wurde, wird diese Information

auch nicht angezeigt. Der Webbrowser hat damit aber eine wichtige Information erhalten, denn bei der Darstellung einer Seite richten sich Webbrowser unter anderem nach den Sprachen und Versionen, in denen Dokumente verfasst sind.

Absätze

Wenn Sie in Ihrer Webanwendung mehrere Absätze Text einfügen wollen, dann machen Sie das mit dem `p`-Container. Nutzen Sie bitte keinesfalls leere Container oder den `<br \>`-Container, um leere Zeilen einzufügen. Wenn Sie solche Kniffe anwenden, dann programmieren Sie die Ansicht der Webanwendung. Das ist aber Mediendesign und hat in der Programmierung von HTML5 nichts zu suchen.

```
<p> ... Erster Absatz ... </p>
<p> ... Zweiter Absatz ... </p>
usw.
```

Pflegen wir das doch gleich in unsere `.txt`-Dateien ein: Erweitern Sie den Inhalt der `echo()`-Funktion um einige Absätze, die Sie genauso programmieren, als wenn sie direkt im HTML-Dokument stehen würden. (Wenn die Ausgabe nicht ganz so aussieht, wie Sie es erwarten, dann gedulden Sie sich noch ein wenig: Neben einem einfachen Syntaxfehler gibt es noch die Möglichkeit, dass Sie das HTML-Dokument nicht lokalisiert haben, oder dass Sie Sonderzeichen wie `"` verwendet haben, die von der `echo()`-Funktion nicht übernommen werden. Aber keine Sorge, um beide Fälle kümmern wir uns bald.

Kommentare

Bei längeren HTML-Dokumenten oder Containern, die nicht selbstredend sind, sollten Sie außerdem Kommentare einpflegen. Kommentare sind Codezeilen, die nicht ausgeführt werden, sondern ausschließlich dazu dienen, um anzuzeigen, was in einem Teil einer Programms oder Skripts passiert. Im Gegensatz zu anderen Containern sieht ein Kommentar wie folgt aus:

```
<!-- Kommentar, auch über mehrere Zeilen -->
```

Bei Kommentar-Containern gibt es also keinen Abschluss durch den Slash, wie das bei anderen Tags der Fall ist.

In der Vergangenheit wurden spezielle Befehle, die nur für den Internet Explorer zugeschnitten waren in Form von Kommentaren einprogrammiert; sollten Sie also in fremdem HTML-Skripten Kommentare der Form `if IE ...` treffen, dann wissen Sie jetzt, worum es dabei geht.

Attribute

Später werden Sie in öffnende Tags noch Attribute einfügen. Attribute legen Einschränkungen oder Erweiterungen fest, die für einen Container gilt und für alles, was sich darin befindet. Diese programmieren Sie ausschließlich ins öffnende Tag. Das schließende Tag beinhaltet weiterhin nur die Bezeichnung eines Containers, um dem Browser anzuzeigen, dass der Container hier „zuende“ ist.

Wenn Sie zum Beispiel für den `html`-Container: Wenn Sie hier das `lang`-Attribut festlegen wollen, über das wir noch nicht gesprochen haben und diesem Attribut den Wert `de` zuordnen, sieht das öffnende `html`-Tag so aus: `<html lang=de>`. Das schließende `html`-Tag wäre dann aber immer noch `</html>`. Also wäre es unsinnig, hier die Attribute zu programmieren: Attribute gelten immer für einen Container und gelten damit nicht mehr, sobald der Container geschlossen ist.

Übrigens legen Sie mit dem `lang`-Attribut die sogenannte Internationalisierung fest. Was das im Detail bedeutet besprechen wir später.

Für diejenigen, die bereits mit HTML4.01 programmiert haben: Dort war es üblich, über Attribute die Gestaltung von Containern direkt über das `style`-Attribut zu programmieren. Das passiert unter HTML5 nur noch indirekt über die beiden Attribute `class` und `id`. Bei diesen funktioniert es aber noch genauso wie unter HTML4.01.

Aufgaben

1. Erstellen Sie jetzt die folgenden sechs HTML-Dokumente, programmieren Sie jeweils DTD und die drei Container:
 - `arbeitsweg.php`
 - `meinCampus.php`
 - `meineHobbies.php`
 - `meineZiele.php`
 - `meinBlog.php`
 - `index.php`
2. Erklären Sie in eigenen Worten, warum es für die Ansicht im Browser keinen Unterschied macht, ob die Dateiendung `.html` oder `.php` lautet.
3. (Für Fortgeschrittene, benötigt Recherche im Netz) Erklären Sie in eigenen Worten, warum es für die Ansicht im Browser einen großen

Unterschied macht, ob Sie HTML-Dokumente, in denen PHP-Code enthalten ist direkt vom Browser geöffnet oder von einem Webserver (also per localhost) an den Browser übertragen werden.

Zusammenfassung

Sie kennen jetzt alles, was Sie benötigen, um eine einzelne Seite mit Text ins Netz zu stellen.

6.1 Barrierefreiheit, Internationalisierung und Lokalisierung

Die meisten Kurse zu HTML4.01 kommen (wenn überhaupt) erst ganz zum Schluss zu diesem Thema, dabei ist es für Webanwendungen essentiell: Dieser Abschnitt vermittelt Ihnen ein paar grundlegende Informationen dazu, wie Sie sicherstellen können, dass Ihre Webanwendung auf den verschiedensten Endgeräten und von den verschiedensten Nutzern genutzt werden kann. Dazu gehört aber vor allem etwas, das bereits mehrfach betont wurde: Programmieren Sie in HTML5 nur dann Design, wenn es anders unmöglich ist.

Der Grund für die Bedeutung dieses Kapitels ist so simpel wie ärgerlich:

Viele Entwickler vergessen schlicht, dass eine Webanwendung auf einem Smartphone anders dargestellt wird als auf einem Rechner mit einem 2k-Display, das horizontal ausgerichtet ist. Außerdem nutzen nicht nur hörbehinderte Personen Software, die Ihnen Webanwendung vorliest. Und zu guter Letzt bieten mehr und mehr Browser eine automatische Übersetzung von Texten an.

Damit Ihre Webanwendung unter all diesen Bedingungen immer noch gut aussieht, mussten Sie unter HTML4.01 eine ganze Menge Arbeit leisten, die bei HTML5 mit wenigen Befehlen erledigt werden kann. Wobei all die guten Vorsätze nichts Wert sind, wenn Sie im Team jemanden haben, der in HTML Design programmiert.

6.1.1 Barrierefreiheit

Hier geht es darum, eine Webanwendung so zu programmieren, dass Sie auch dann noch gut aussieht, wenn der Nutzer einen starken Zoom-Faktor nutzt oder sich die Webanwendung durch eine Software vorlesen lässt. Darüber hinaus geht es aber auch um Personen, die beispielsweise die Sprache der Webanwendung nicht gut verstehen.

Aufgabe

Öffnen Sie die Webpage der HAW mit Ihrem Smartphone und versuchen Sie nun, über die Links auf der Startseite auf die Seite des Departments Medientechnik zu gelangen.

Selbst wenn Sie gute Augen haben, werden Sie merken, dass diese Aufgabe sehr anstrengend ist. Und das liegt eben nicht daran, dass es allzu schwer ist, die passenden Links zu finden; vielmehr wurde bei der Webpage der HAW die Barrierefreiheit zum Teil ignoriert. Damit sollte Ihnen klar sein, dass Barrierefreiheit kein belangloses Thema für Randgruppen ist, sondern dass die Missachtung der Barrierefreiheit ein Problem ist, das Nutzer davon abschreckt eine Webanwendung zu nutzen.

Es gibt drei einfache erste Prüfungen, um die grundsätzliche Einhaltung der Barrierefreiheit zu prüfen:

1. Es gibt einen Seitentitel, der kurz und zutreffend ist und die Seite von anderen Seiten unterscheidet.
2. Jeder Inhalt in einem Container bekommt eine Überschrift. (Ausnahmen sind z.B. `<p>`-Container.)
3. Für jedes multimediale Element gibt es eine alternative Bezeichnung, die Sie mit Hilfe des `alt`-Attributs festlegen können.

Den ersten Punkt können Sie jetzt schon umsetzen, zum zweiten und dritten Punkt kommen wir, sobald wir die entsprechenden Container behandeln.

Kontrolle:

- Sie verstehen, dass es bei Barrierefreiheit nicht nur um die Unterstützung von Menschen geht, die körperlich oder geistig beeinträchtigt sind.
- Sie wissen vielmehr, dass die Beachtung der Barrierefreiheit wichtig ist, um Nutzer nicht von der eigenen Webanwendung zu vertreiben.
- Ihnen ist klar, dass dazu wesentlich mehr nötig ist, als die drei Prüfungen, die Sie gerade kennen gelernt haben. (Für diesen Kurs soll das aber als kleine Einführung genügen.)

6.1.2 Internationalisierung (kurz i18n) und Lokalisierung (kurz l10n)

Wenn Sie sich fragen, warum die Abkürzung i18n lautet, hier ist die Antwort: Das englische Wort internationalization beginnt mit einem i, hat 18 Buchstaben und endet mit einem n. Auf ganz ähnliche Weise kommen sie von localization zu l10n.

Nach der Barrierefreiheit kümmern wir uns nun darum, dass der Browser mit den deutschen Umlauten zurechtkommt. Dazu müssen wir zwei Dinge erledigen:

1. Zum einen müssen wir unsere Seite **internationalisieren**. Das bedeutet schlicht, dass wir dem Browser mitteilen, dass unsere Seite auf Deutsch (bzw. in einer anderen Sprache) verfasst wurde. Dieser Schritt dient Browsern mit Übersetzungsalgorithmen, um unsere Seite automatisch „richtig“ zu übersetzen. Sie werden später erfahren, wie wir einzelne Passagen unserer Webanwendung zu programmieren können, dass sie von der automatischen Übersetzung ausgenommen werden.
2. Danach müssen wir noch die **Codierung** festlegen, damit der Browser weiß, dass wir z.B. deutsche Umlaute verwenden. Dieser Schritt wird als **Lokalisierung** bezeichnet. Im Gegensatz zur Internationalisierung werden Sie die Auswirkung der Lokalisierung direkt auf der Webanwendung sehen.

Wie Sie Ihre Webanwendung internationalisieren und lokalisieren erfahren Sie direkt im nächsten Abschnitt.

6.2 Der head-Container, Meta-Daten und Attribute

Sie wissen, dass unser html-Container (der auch als **Wurzelement** eines HTML-Dokuments bezeichnet wird) die beiden Container head und body enthält. Wenn Sie sich nun die Webanwendung ansehen, dann können Sie sich schon denken, was die beiden unterscheidet: Im head steht alles, was nicht im Fenster des Browsers angezeigt wird. Hier finden sich zusätzliche Informationen und allgemeine Definitionen, die für die gesamte Webanwendung gelten. Wenn es also eine Einstellung gibt, die für alle Container im <body> gelten soll, dann werden Sie sie in aller Regel im <head> programmieren. Beispielsweise wird hier die Lokalisierung festgelegt. Solche allgemeinen Definitionen, die für eine ganze Webanwendung gelten, werden auch als **Meta-Daten** bezeichnet. Hier der entsprechende Container:

```
<meta charset=utf-8 />
```

Dieser meta-Container legt fest, dass als Codierung für unsere Webanwendung UTF-8 verwendet werden soll. Codierung ist Teil der Nachrichten- und Kommunikationstechnik, wird aber auch in einführenden Veranstaltungen der Technischen Informatik besprochen.

Außerdem sehen Sie, dass der Container am Ende einen `/` enthält. Wie Sie bereits wissen, gibt es dafür einen einfachen Grund: Wenn ein Container keinen Inhalt hat, dann können Sie ihn so abschließen und brauchen kein eigenständiges schließendes Tag programmieren. In HTML5 ist das nur dann nötig, wenn ein Container im Regelfall einen Inhalt hat. Hier können Sie also genauso gut die folgende Zeile programmieren:

```
<meta charset=utf-8>
```

Damit haben Sie auch Ihr erstes Attribut mit einer Wertzuweisung kennen gelernt: `charset` ist das Attribut, `utf-8` der Wert, der diesem Attribut zugeordnet wird.

Nun wollen wir unsere Webanwendung internationalisieren. Das Attribut für Sprache lautet `lang`. Da wir aber nicht nur allgemein festlegen wollen, dass eine Sprache verwendet wird (das wäre ja sinnlos), müssen wir noch festlegen, dass die Sprache Deutsch sein soll. Und das tun wir, indem wir mit `lang=de` als Wert des Attributs festlegen. Im Gegensatz zur Lokalisierung ist `lang` ein Attribut des `html`-Containers.

Wenn Sie die nötigen Änderungen an Ihren HTML-Dokumenten durchgeführt haben, sieht das also so aus:

```
<!doctype html>
<html lang=de>
<head>
<meta charset=utf-8>
<title><?php include(test_title_001.txt); ?></title>
</head>
<body>
<?php include(test_begrueessung_001.txt); ?>
</body>
</html>
```

Sie wollen wissen, welche Codierung Sie für Sprachen wie Farsi (Afghanistan) oder Chinesisch brauchen? Genau dieselbe wie für Deutsch. Eine weitere Anpassung im Rahmen der Lokalisierung ist also nicht nötig. Aber Sie müssen die Internationalisierung in diesen Fällen über das `lang`-Attribut anpassen.

Aufgabe:

Nachdem Sie Ihr Dokument um Internationalisierung und Lokalisierung erweitert haben, laden Sie es erneut. Jetzt wird endlich das ü im Browser als ein ü angezeigt.

6.2.1 Entitys - Umlaute und andere Sonderzeichen

Es kann jedoch (aufgrund eines veralteten Browsers bei NutzerInnen) immer passieren, dass die Lokalisierung nicht funktioniert. Diese Abschnitt erklärt, wie Sie in dem Fall Sonderzeichen einfügen können und diese Methode funktioniert immer:

In diesem Fall müssen Sie beispielsweise anstelle des Buchstaben ü die Sequenz `ü` in den fließenden Text eintragen. Aber das sieht nur auf den ersten Blick unübersichtlich aus: Jede dieser Sequenzen beginnt mit dem kaufmännischen Und-Zeichen und endet mit einem Semikolon.

Die Zeichenkette dazwischen ist für alle Umlaute simpel: Zunächst der Buchstabe (a, o, u bei kleinen Umlauten und A, O, U bei großen Umlauten) und dann die Zeichenfolge `uml` für Umlaut.

Ähnliches gilt für die übrigen Sonderzeichen, die bei HTML als Entitys bezeichnet werden. Warum sie nicht entsprechend der englischen Grammatik als Entities bezeichnet werden, kann ich Ihnen nicht sagen. Zu Fragen und Nebenwirkungen wenden Sie sich bitte den Anglisten Ihres Vertrauens.

Hier eine Tabelle der für den Einstieg wichtigsten Entities :

Ä	<code>&Auml;</code>	
ä	<code>&auml;</code>	
Ö	<code>&Ouml;</code>	
ö	<code>&ouml;</code>	
Ü	<code>&Uuml;</code>	
ü	<code>&uuml;</code>	
ß	<code>&szlig;</code>	(ß mit Ligatur)
€	<code>&euro;</code>	
&	<code>&amp;</code>	(ampers and)
>	<code>&gt;</code>	(greater than)
<	<code>&lt;</code>	(less than)
Anführungszeichen unten	<code>&bdquo;</code>	
Anführungszeichen oben	<code>&rdquo;</code>	

Sollten Sie einmal ein Zeichen nutzen wollen, dass Sie in keiner HTML-Tabelle finden, dann gibt es noch eine Lösung: Zeichen, die in der Unicode-Tabelle bzw. der ISO 10646 aufgeführt werden, können wie folgt eingefügt werden. Wählen Sie dazu die Nummer aus der Codetabelle, die Ihrem Zeichen entspricht und fügen es zwischen `&#` sowie dem Semikolon ein. Sollte der Wert ein Hexadezimalwert sein, müssen Sie noch ein `x` einfügen.

Beispiel: Nehmen wir an, Sie wollen ein bestimmtes chinesisches Element in Ihrem Text darstellen, dass im Unicode die Codenummer 9FB9 hat. Es ist leicht zu erkennen, dass es sich hier um eine hexadezimale Zahl handelt. Also müssen wir noch ein `x` in `&#` ; einfügen. Es ergibt sich also folgende Sequenz, mit dem wir das gewünschte Zeichen einfügen können:

```
&#x9fb9;
```

Kontrolle

Machen Sie das einmal selbst: Laden Sie sich die Unicode-Tabellen herunter (Umfang knapp 130 MB). Fügen Sie dann Ihre Email-Adresse in das HTML-Skript, wobei Sie den Wert für das @-Symbol anhand der Codetabelle in Ihren Text einfügen. Sie werden hier nicht lange suchen müssen: Der Eintrag befindet sich gleich auf der ersten Seite in einem der Dokumente, die Sie heruntergeladen haben.

Hinweis: Beachten Sie bitte, dass es sich hier um eigenständige Zeichen handelt, die mit dieser Zeichenkette erzeugt werden und nicht etwa um Container oder Tags. Spitze Klammern haben an dieser Stelle also nichts verloren.

6.3 Mehr Grundlagen in HTML

6.3.1 Mehr Auslagerung von Code

Aufgabe:

Oben haben Sie gelernt, wie Sie Texte aus Dateien mit Hilfe von PHP in eine HTML-Dokument einfügen können. Stellen Sie sich vor, Sie müssten 95 HTML-Dokumente programmieren und bei allen würde in den ersten drei Zeilen der folgende Code stehen:

```
<html lang=de><head><meta charset=utf-8>.
```

Programmieren Sie die Auslagerung dieses Code-Fragments in eine Datei,

die per PHP ausgelesen und in ein HTML-Dokument eingefügt wird.

Hinweis:

Eine Auslagerung von so wenig Code ist meist nicht sinnvoll, weil es Ihnen den Überblick über den Code erschwert. Aber wenn wir zur Programmierung von PHP kommen werden Sie feststellen, dass Sie sich mit dieser Methode viel Tipparbeit ersparen können. Und das bedeutet sehr oft eine deutlich reduzierte Fehleranfälligkeit. Außerdem stellen Sie damit sicher, dass Code-Fragmente, die in mehreren HTML-Dokumenten identisch sein sollen auch dauerhaft identisch sind. Stellen Sie sich umgekehrt den Aufwand und die Fehlerwahrscheinlichkeit vor, wenn Sie in 95 HTML-Dokumenten eine kleine Änderung durchführen müssen.

Kontrolle

- Sie wissen, dass jedes HTML-Dokument aus einer Doctype Declaration sowie den drei Containern `html`, `head` und `body` besteht.
- Sie haben eine erste Vorstellung davon, was Sie im `<head>` und was Sie im `<body>` programmieren.
- Sie wissen, dass Sie Text im Webbrowser anzeigen können, indem Sie ihn innerhalb des `<body>`-Containers eingeben.
- Sie verstehen noch nicht genau, wie Sie mit Hilfe von Attributen einzelne Container ändern können.

6.3.2 Verwendung von Escape-Sequenzen

Vor HTML5 galt die Aussage, dass Sie grundsätzlich davon ausgehen mussten, dass Sie Sonderzeichen einer Sprache mit sogenannten Entitys programmieren mussten. Diese werden Ihnen auch weiter in HTML-Quellcode begegnen, weil Sie ja beispielsweise eine spitze Klammer nicht als Text in HTML einfügen dürfen.

Standardmäßig beherrschen Webbrowser ausschließlich die Zeichen, die in der sogenannten ASCII-Tabelle aufgeführt sind. Für HTML4.01 bedeutet das: Nur Zeichen, die im englischen Alphabet vorkommen können direkt auf einer Webanwendung angezeigt werden. Alle anderen müssen mit einer sogenannten Escape-Sequenz ausgedrückt werden. Diese beginnt mit einem `&`-Zeichen und endet mit einem Semikolon. Das deutsche ß mussten Sie unter HTML4.01 mit der Escape-Sequenz `ß` programmieren.

Weiterhin gab es noch die Möglichkeiten, Escape-Sequenzen mit der Nummer der Unicode-Tabelle zu programmieren. Beispielsweise steht `龹`

für ein chinesisches Schriftzeichen. Stellen Sie sich einmal vor, Sie müssten auf diese Weise eine Webanwendung Zeichen für Zeichen programmieren...

Auch hier zeigt sich, dass die Entwickler beim W3C mit HTML5 einen Standard veröffentlicht haben, der wirklich sinnvolle Änderungen einführt, die auch das Programmieren von Webanwendung deutlich vereinfacht, ohne dabei die Funktionalität oder das Layout von Seiten zu beschränken. Vielmehr ist das Gegenteil der Fall.

Es folgen einige Escape-Sequenzen, die Sie auch weiterhin benötigen werden:

```
&amp;   &   (kaufmännisches Und)
&lt;     <   (less than)
&gt;     >   (greater than)
```

Nehmen wir dazu an, dass Sie innerhalb eines HTML-Dokuments den folgenden Satz angeben wollen:

Die Zeichenfolge `<p>` öffnet einen Absatz-Container in HTML,

dann müssen Sie das so einprogrammieren:

Die Zeichenfolge `<p>` öffnet einen Absatz-Container in HTML.

Unter HTML4.01 hätten Sie zusätzlich für das ö von öffnet die Entity `ö` eingeben müssen:

Die Zeichenfolge `<p> öffnet` einen Absatz-Container in HTML.

6.3.3 HTML5: Anführungszeichen sind weitestgehend optional

Wenn Sie sich ältere HTML-Kurse ansehen, werden Sie feststellen, dass bei Wertzuordnungen wie `lang=de` der zugeordnete Wert immer in Anführungszeichen steht: `<html lang="de">` Das ist bei HTML5 nicht mehr zwingend vorgeschrieben.

Wenn Sie allerdings Werte zuordnen, die Leerzeichen beinhalten oder eine Webanwendung für veraltete Browser entwickeln wollen, dann sollten Sie in jedem Fall Anführungszeichen verwenden.

6.3.4 Zusammenfassung

Für den Rest dieses Kapitels werden wir nicht wieder über die Container `<html>` und `<head>` sprechen.

6.4 Strukturen von HTML5-Dokumenten

Auch wenn das offensichtlich sein sollte, sei hier betont: Alle Container, die in diesem Abschnitt besprochen werden können ausschließlich im body-Container eingesetzt werden.

All diese Elemente haben in HTML5 eine feste Bedeutung, die insbesondere im Rahmen der Barrierefreiheit von Belang ist. Bei HTML4.01 mussten Webentwickler sich noch mit div-Containern behelfen, denen Sie nicht-standardisierte Attribute zuordneten. Die Folge bestand darin, dass Browser die in HTML-Dokumenten vorgegebenen Strukturen nicht erkennen konnten und somit weitgehend willkürlich die Ansicht generierten.

Das wiederum brachte viele „professionelle“ EntwicklerInnen dazu, Unmengen an Zeit damit zu verschwenden, HTML-Dokumente mit tausenden Zeilen Code zu erweitern, die letztlich nur dafür sorgen sollten, dass die Ansicht wie gewünscht in jedem denkbaren Browser angezeigt wurde.

Deshalb werden Sie heute eine Vielzahl an Frameworks finden, die Ihnen diese Arbeit abnehmen, die aber gleichzeitig garantieren, dass Ihre Webanwendung auf bestimmten Endgeräten oder für bestimmte NutzerInnen grausig aussehen werden, weil z.B. die Barrierefreiheit ignoriert wird. Die Seite der HAW ist da nur ein Beispiel. Schauen Sie sich beispielsweise Seiten auf einem Rechner an, deren Design für Smartphones entwickelt wurde: Zum Teil füllen deren Überschriften ein Viertel des Bildschirms: Das ist mangelhaftes Webdesign! Nehmen Sie bento (einen Ableger von Spiegel online), um einen Eindruck zu bekommen, was hier gemeint ist: <http://www.bento.de/>

Also nochmal: Ignorieren Sie bitte das Design Ihrer Anwendung, denn so lange Sie nicht über jahrelange Erfahrung in diesem Bereich verfügen werden Sie sonst nur Anwendungen entwickeln, die auf einer Art von Endgerät gut aussehen, auf allen anderen dagegen grausig. Und das ist das Vorgehen von Amateuren.

6.4.1 header, footer, main und aside

Mit diesen vier Containern können Sie eine grobe Struktur Ihrer Webanwendung vorgeben.

- Im `<main>` sollen sich alle Inhalte befinden, die auf der Webanwendung zentral angezeigt werden sollen.
- Der `<aside>`-Container ist dann für ergänzende Inhalte zu den Inhalten im `<main>` gedacht.
- Im `<header>` (nicht zu verwechseln mit dem `<head>`) können Sie beispielsweise ein Unternehmenslogo oder ähnliches unterbringen.
- Der `<footer>` dient dann dazu, um beispielsweise einen Verweis auf das Kontaktformular, das Impressum und ähnliches aufzunehmen.
 - Der `<nav>`-Container kann in jedem der anderen vier Container untergebracht werden. Er ist vorrangig dafür gedacht, um eine Navigationsleiste zu realisieren. (Wir haben noch nicht darüber gesprochen, wie Sie einen Link auf eine andere Seite programmieren, aber auch dazu kommen wir bald.)

Wenn also ein Browser diese Container unterstützt, dann könnte das bedeuten, dass Header und Footer automatisch am oberen bzw. unteren Rand des Bildschirms eingeblendet werden, während NutzerInnen durch die Seite scrollen. Der Main-Bereich würde dann rund zwei Drittel der Breite und der Aside-Bereich ein Drittel des Bildschirms einnehmen.

Bei Smartphones mit kleinen Displays würden dagegen Header und Footer wahrscheinlich automatisch minimiert werden und Main sowie Aside untereinander angezeigt werden, damit die Ansicht auf dem Gerät im Sinne der NutzerInnen aufgebaut wäre.

Es wären auch noch viele weitere Möglichkeiten denkbar, aber letztlich würden gut programmierte Browser diese Entscheidung anhand der Bauweise des Gerätes selbst durchführen. Als HTML-EntwicklerInnen könnten uns dann vollständig darauf konzentrieren, gut strukturierte semantische Webpages zu entwerfen, anstatt den Großteil unserer Zeit damit zu verschwenden jeden denkbaren Fall manuell zu programmieren.

6.4.2 article und section

Diese beiden Container kommen vorrangig im `<main>` und `<aside>` zum Einsatz. Das Konzept sieht wie folgt aus: Wenn Sie wie bei einem Buch die Inhalte Ihrer Webanwendung in Kapitel und Unterkapitel unterteilen wollen, dann beginnen Sie mit einem `<article>`. Sobald ein Unterkapitel beginnt, fügen Sie innerhalb dieses Containers einen `<section>`-Container ein. Wenn Sie dann noch weiter unterteilen wollen, fügen Sie an der entsprechenden Stelle des `<section>` einen `<article>` ein usw. usf. Wichtig ist nur, dass Sie zum einen mit `<article>` beginnen, und dass Sie

die beiden Container-Typen im Wechsel verwenden, wenn Sie jeweils eine weitere unter-Struktur (Unter-Unter-Kapitel zum Unter-Kapitel) beginnen wollen.

6.4.3 h1 bis h5

Alle Container, die Sie in den beiden vorigen Abschnitten kennen gelernt haben, sollen in HTML5 mit einer Überschrift (engl. heading) beginnen. Leider gibt es kein allgemeines `<h>`-Element, mit dem Sie eine Überschrift definieren können.

Vielmehr müssen Sie entsprechend der Struktur, die Sie z.B. durch `article` und `section` konstruiert haben den passenden Container (`<h1>` bis `<h5>`) auswählen. Durch diese Auswahl legen Sie fest, wie die jeweilige Überschrift angezeigt wird. `<h1>` ist eine Kapitelüberschrift, `<h2>` die Überschrift eines Unterkapitels, usw.

6.4.4 p

Wenn Sie einen `article` oder eine `section` in mehrere Absätze unterteilen wollen, dann nutzen Sie dafür den `<p>`-Container. Es gibt zwar auch noch den `<div>`, der bei HTML4.01 sehr oft verwendet wurde, aber fast alles, wofür der dort verwendet wurde wird in aller Regel durch die Container erfüllt, die in den beiden ersten Abschnitten aufgezählt wurden.

Der `<div>` ist der einzige Container, für den keine semantischen Merkmale festgelegt werden können. Alleine deshalb sollten Sie im Regelfall `<p>` verwenden. Bitte missverstehen Sie das nicht: Der Inhalt eines `div`-Containers kann selbstverständlich auch semantische Informationen enthalten, der Container selbst dagegen nicht.

Für diejenigen, die bereits HTML4.01 programmiert haben:

Wenn Sie Container wie `<div class="main">`
`<div class="section">` usw. verwenden, ist klar, dass Sie nichts von dem verstanden haben, was hier wir bislang über HTML5 besprochen haben.

`<h1>` bis `<h5>`, `<div>` und `<p>` gab es bereits in HTML4.01. In HTML5 haben Sie aber zusätzlich die eine Vielzahl an Containern, die für die standardisierte Strukturierung einer Webanwendung genutzt werden sollen.

6.4.5 Aufgabe

Integrieren Sie die neuen Container in die bisherigen Dateien, auch wenn sie damit leer sind. Vergessen Sie dabei nicht, ggf. über PHP identische Teile unterschiedlicher HTML-Dokumente auszulagern.

6.5 Polyfills

Das englische Wort **Polyfill** bedeutet schlicht Spachtelmasse. Es geht hier also um Programm-fragmente, die dazu dienen, um Lücken aufzufüllen. Sie werden immer wieder auf Möglichkeiten von HTML5 treffen, die von einzelnen Webbrowsern nicht oder nicht vollständig unterstützt werden. Wenn Sie dann einen Container programmiert haben, wird der im betreffenden Browser nicht wie gewünscht angezeigt und seine Elemente werden wie bei HTML4.01 relativ willkürlich platziert.

Aber das ist kein echtes Problem, denn für die meisten dieser Fälle gibt es eine Lösung, die Sie per Copy-Paste in Ihre Webanwendung kopieren können. Und diese Lösungen werden als Polyfill bezeichnet.

Wichtig ist hier nicht, dass Sie sich merken, für welche Fälle Sie ein Polyfill benötigen, denn das ändert sich ja kontinuierlich mit jeder neuen Version der verschiedenen Browser. Wichtig ist vielmehr, dass Sie sich merken, auf welcher Seite Sie prüfen können, ob Sie ein Polyfill benötigen und wo Sie es bekommen:

- Auf www.caniuse.com können Sie für jeden HTML5-Container prüfen, in welchem Browser er unterstützt wird. Häufig finden Sie bei den einzelnen Containern auch Verweise auf Seiten, auf denen Sie ein passendes Polyfill finden.
- Auf www.html5please.com finden Sie eine Vielzahl an Polyfills. Merken Sie sich dazu den folgenden kurzen Dialog:

- What **can I use**?
- **HTML5, please**.

Denn damit haben Sie sich schon fast die URLs der beiden Seiten gemerkt.

Aktualisierung am 15. März 2016:

Es gibt zwar noch Fälle, in denen einzelne HTML5-Container in einzelnen Browsern nicht oder nicht vollständig unterstützt werden. Vollständigkeit halber werde ich diesen Abschnitt deshalb noch hier belassen, langfristig dürfte die Arbeit mit Polyfills aber überflüssig werden.

6.5.1 Einbindung von Polyfills - script-Container

Polyfills werden in der Sprache **JavaScript** programmiert. Sie brauchen sich jedoch keine Gedanken zu machen, wenn Sie diese Sprache nicht kennen. Denn Sie müssen lediglich einen `<script>`-Container in Ihren `<html>`-Container einfügen. Sämtlichen JavaScript-Code, den Sie verwenden wollen, müssen Sie lediglich in diesen `<script>`-Container kopieren und schon wird er auf Ihrer Webanwendung angewendet.

Wichtig ist dabei nur, dass Sie jeweils genau die Bezeichnungen für Container verwenden, die bei HTML5 gelten. Denn genau wie CSS ändern Polyfills die Art, wie ein bestimmter HTML-Container dargestellt wird. Das geht aber nur, wenn die Container die richtige Bezeichnung haben. Sollten Sie sich also vertippen und anstelle des `<aside>`-Containers einen `<aseid>`-Container programmieren, dann wird Ihnen ein Polyfill, das die Darstellung des `<aside>`-Containers sicherstellt nichts nützen.

Auch hier wieder ein Hinweis für fortgeschrittene HTML-Programmierer: In HTML4.01 mussten Sie für die Verwendung von JavaScript noch eine Zeile im `<head>`-Container erstellen, die dem Browser mitteilt, dass „Scripte“ in JavaScript (oder einer anderen Programmiersprache) erstellt werden. Das ist bei HTML5 nicht mehr nötig, weil hier JavaScript die Standardprogrammiersprache ist.

6.6 Zusammenfassung

Die folgenden beiden HTML-Dokumente zeigen, wie Sie mit den bisher vorgestellten HTML5-Elementen eine einfache Webanwendung entwickeln können. Wichtig: Bislang haben Sie noch keine Möglichkeit kennen gelernt, um

- Links auf andere Seiten zu erstellen,
- Bilder und andere multimediale Inhalte einzufügen oder
- die Webanwendung semantisch zu machen.

Aber keine Sorge, all diese Punkte und noch viel mehr werden wir in Kürze behandeln.

Hier ein Beispiel, in dem der Anfang dieses Skripts als zwei HTML-Dokumente zusammengefasst ist. Die Absätze wurden deutlich gekürzt, damit Sie jeweils sehen können, wie die Programmierung aussehen kann.

```
<!doctype html>
<html lang=de>
<head>
<meta charset=utf-8>
<title>PRG - Vorwort</title>
</head>

<body>
<header>
<!-- Hier später das Logo der Webanwendung einfügen
-->
</header>

<main>
<h1>Vorwort</h1>
<p>Ein häufiges Missverständnis besteht darin,
dass Programmierung und Informatik miteinander
verwechselt werden. ....</p>
</main>

<footer>
<!-- Hier später den Link auf das Impressum einfügen.
-->
</footer>

<aside>
<article>
<h1>Zusammenfassung</h1>
<p>Informatiker entwickeln Konzepte und Modelle,
um Ideen möglichst effizient umzusetzen. Dabei
spielt es keine Rolle, ob diese Konzepte mit
einem Computer umsetzbar sind.
</p>
<p>Programmieren sind Menschen, die Konzepte
und Modelle in eine Programmiersprache umsetzen.
</p>
</article>
</aside>
</body>
</html>

<!doctype html>
<html lang=de>
<head>
```

```
<meta charset=utf-8 />
<title>PRG - WWW und semantic Web</title>
</head>

<body>
<header>
<!-- Hier später das Logo der Webanwendung einfügen
-->
</header>

<main>
<article>
<h1>Vorwort</h1>
<p>Programmierveranstaltungen bereiten Sie in aller
Regel darauf vor, Programme für einen Computer zu
....</p>
<section>
<h2>Programmierung von Webanwendung und
Webapplicationen</h2>
<p>Die bekannteste Markup Language ist HTML, die
HyperText Markup Language. ...</p>
<p>Wie gesagt definieren Sie in einer Markup Language
lediglich ...</p>
</section>
<section>
<h2>Das semantische Web</h2>
<p>Bis hierher haben Sie nur über Dinge gelesen, die
Sie unter ...</p>
<p>Sehen wir uns das mal im Detail an:</p>
<article>
<h3>Syntax und Semantik </h2>
<p>Den einen dieser Begriffe haben Sie wahrscheinlich
in der Schule kennen und ...</p>
<p>Damit kommen wir zur Semantik. Mit Semantik
bezeichnen wir ...</p>
<p>Aber es gibt einen sehr großen Unterschied
zwischen ...</p>
<p>Stellen Sie sich nun die folgende Situation
vor, ...</p>
</article>
</section>
</article>
</main>
```

```
<footer>
<!-- Hier später den Link auf das Impressum einfügen.
-->
</footer>

</body>
</html>
```

6.6.1 Hausaufgabe

Sie haben vorhin sechs Dateien (arbeitsweg.html usw.) erstellt. Erweitern Sie diese Seiten zum nächsten Mal wie folgt:

- Internationalisieren und Lokalisieren Sie jede Seite.
- Erstellen Sie für jede Seite eine grundlegende Struktur, bei der Sie insbesondere sämtliche hier vorgestellten HTML5-Container sinnvoll verwenden.
- Füllen Sie Ihre Seiten mit Inhalten, die zum Seitentitel passen.
- Wenn Sie irgendwelche Inhalte wie Spiele, Videos, Bilder usw. einfügen wollen, dann tragen Sie an der entsprechenden Stelle einen Platzhalter ein. (Z.B. „Einzufügen: Bild von Horst“, „Hier mein geniales Gitarrensolo einbauen.“ usw. usf.) Wichtig: Es spielt keine Rolle, ob Sie sich zutrauen, die entsprechenden Inhalte selbst zu entwickeln. Lassen Sie einfach Ihrer Phantasie freien Lauf.
- Lassen Sie die Datei `index.html` vorerst so, wie Sie ist.
- Suchen Sie nach einem Polyfill und programmieren Sie es ein, damit der `<aside>`-Container in den folgenden Browsern „richtig“ angezeigt wird: Firefox, Internet Explorer, Safari, Edge

6.7 Hyperlinks

Vorhin haben Sie gelernt, dass das Protokoll zur Übertragung von Webanwendung HyperText Transfer Protokoll heißt. Sie wissen bereit, dass das Wort Protokoll nur eine Bezeichnung für eine Vereinbarung darüber, wie etwas zu tun ist. In diesem Fall geht es also um eine Vereinbarung darüber, wie Hypertexte übertragen (transferiert) werden sollen. Also kommen wir kurz dazu, was denn nun wiederum **Hypertexte** sind und was die mit Webanwendung zu tun haben.

Die Antwort ist ganz einfach: Im Gegensatz zu einem Buch beinhaltet eine Webanwendung Kreuzverweise, denen Sie folgen können. Diese Verweise kennen Sie umgangssprachlich als Links. Eine Webanwendung ist also mehr als nur eine Ansammlung von Texten wie bei einem Buch. Und aus diesem Grund wurde das, was wir heute Webpage nennen, in der Zeit als **Hypertext** bezeichnet, als das WWW gerade erst entwickelt wurde. Wenn Sie also in **HTML** programmieren, dann programmieren Sie Hypertext. Nur nennt das heute kaum noch jemand so. Im Namen des Protokolls HTTP lebt dieser Begriff wahrscheinlich noch sehr lange weiter.

Damit kommen wir zu den Links, die ursprünglich als Hyperlinks bezeichnet wurden. Das englische Wort Link bezeichnet ja allgemein eine Verbindung. Dementsprechend bezeichnet ein **Hyperlink** eine Verbindung zwischen zwei Hypertexten. Jetzt aber genug über Begriffe, schließlich wollen Sie wissen, wie Sie einen Link programmieren können. Warum der so heißt, wie er heißt, dürfte da für Sie nebensächlich sein.

6.7.1 Anker

Auch wenn wir hier in Hamburg sind, wo Sie am Hafen Anker in Hülle und Fülle finden, reden wir an dieser Stelle über Teile einer Webanwendung, wenn wir über einen Anker reden. **Anker** sind bei HTML Container, auf die ein Link verweisen kann. In anderen Worten: Wenn Sie in Ihrem HTML-Dokument einen Anker definieren, dann können Sie von einer beliebigen anderen Stelle aus auf diese Stelle verweisen. Sie können dann also an einer beliebigen Stelle einen Link einprogrammieren, mit dem ein Nutzer genau bei einem bestimmten Anker landet.

Wie alles andere in HTML waren Anker ursprünglich vollwertige Container. Deshalb können sie auch als eigenständige Container programmiert werden. Sinnvoller ist es allerdings, Container mittels des `id`-Attributs als einen Anker zu programmieren. Hier ein Beispiel für einen Hypertext, in dem ein Überschrift-Container als Anker programmiert wurde:

```
...
<main>
<article>
<h1>Mein leckerstes Fleischgericht</h1>
<p>Dieses Rezept habe ich von meiner Oma, die ...</p>
<p>Als sie dann 1972 in ...</p>
<section>
<h2 id=blanchieren>Blanchieren und andere
Zubereitungsarten</h2>
<p>Und dann sagte sie ...</p>
```

```
</section>
...
```

Auch wenn es naheliegend ist: In einem HTML-Dokument darf jedes **id**-Attribut nur einmal verwendet werden. Das heißt nicht, dass Sie jeweils nur einen Anker programmieren dürfen, sondern dass Sie z.B. innerhalb eines HTML-Dokuments nur einmal `id=blanchieren` einprogrammieren dürfen.

Aufgabe:

- Programmieren Sie einige Anker in Ihre HTML-Dokumente.

6.7.2 Links

Es gibt drei wichtige Varianten von Links, die aber im Großen und Ganzen gleich programmiert werden.

Wenn Sie auf eine andere Seite verlinken wollen, dann nutzen Sie dazu den Container `<a href ...>`. Das `a` am Anfang ist noch ein Überbleibsel aus der Zeit, als Anker in Form des `<a>`-Containers programmiert wurden. Wie gesagt wird dafür heute das `id`-Attribut verwendet.

Hier die drei genannten Varianten:

- Sie wollen auf einen Anker verlinken, der sich im selben HTML-Dokument befindet, aus dem heraus Sie ihn verlinken wollen. Bsp.: Sie haben ein kleines Glossar auf Ihrer Seite, in dem Sie den Anker „blanchieren“ einprogrammiert haben. Ein Nutzer soll innerhalb dieses Glossars zum Anker springen können. Dann sieht der Link-Container so aus:

```
<a href=blanchieren> Beliebiger Text, der auf der
Webanwendung unterstrichen angezeigt wird und damit
anzeigt, dass hier ein Link vorhanden ist. </a>
```

- Sie wollen auf eine andere Webanwendung verlinken. Dann tragen Sie nach dem Gleichzeichen die vollständige URL ein. (Was eine URL ist, klären wir gleich.) Im folgenden Beispiel programmieren wir einen Link auf die Webanwendung des Departments Medientechnik:

```
<a href=http://www.mt.haw-hamburg.de>
Zum Department Medientechnik</a>|
```

- Nehmen wir an, Sie wollen dagegen auf einen Anker auf einer anderen Seite verweisen. Dann geben Sie zunächst die URL der Seite an, gefolgt von einem Hash (das ist dieses Zeichen: #) und gefolgt vom Namen des Ankers. Nehmen wir an, die Seite heißt `meineRezepte.html` und der Anker heißt `kohlroulade`. Dann könnte der Link so aussehen:

```
Zu meinem Rezept für  
<a href=meineRezepte.html\#kohlroulade>  
Kohlrouladen</a>.
```

Aufgabe:

- Programmieren Sie jetzt einige Links auf die verschiedenen Anker Ihrer Webanwendung.

(Sie haben doch die letzte Aufgabe erfüllt, in der Sie Anker programmieren sollten, nicht wahr?)

6.7.3 Verlinkungen als expliziter Download

Bei HTML4.01 bewirkt das Anwählen eines Hyperlinks, dass der Browser versuchen wird, die Datei zu öffnen. Erst wenn er feststellt, dass es sich um ein Format handelt, dass er nicht öffnen kann, wird er einen Download anbieten. Mit HTML5 wurde für `<a href>`-Container das `download`-Attribut eingeführt. Darüber können Sie explizit angeben, dass ein Link heruntergeladen werden soll.

Wenn Sie diesem Attribut einen Namen als Wert übergeben, dann geben Sie dem Browser vor, unter welchem Namen die Datei gespeichert werden soll. Das ist vor allem dann von Vorteil, wenn der Dateiname eher kryptisch ist.

```
<a href=DC9287349723.jpg  
download=FenderAmericanVintage.jpg>  
Foto meiner Gitarre</a>
```

6.7.4 Hausaufgabe:

- Erstellen Sie einige Bilddateien, damit Sie diese nutzen können, um explizite Links in Ihrem HTML-Dokumenten einprogrammieren können.

Wichtig:

Sie müssen diese Bilddateien selbst erstellt haben und dürfen keine rechtlich geschützten Gegenstände aufnehmen. Sie sollten ebenfalls darauf achten, dass keine Personen auf den Bildern zu sehen sind, außer wenn Sie das schriftliche Einverständnis dieser Personen haben. Denn auch wenn die Bilddateien vorerst nicht veröffentlicht werden sollen, könnten Sie ansonsten rechtliche Probleme bekommen, bei denen durchaus Bußgelder im vierstelligen Bereich drohen.

Diesen Hinweis können Sie auf alle Daten und Dateien beziehen, mit denen Sie arbeiten. Es spielt hier zunächst keine Rolle, ob sie Dateien selbst erstellen oder „nur“ weiterverwenden. Auch wenn Sie sie gar nicht veröffentlichen kann es teuer werden: Die Rechtslage ist hier sehr schnell gegen sie. Mehr darüber lernen Sie in der Veranstaltung Medienrecht. Wenn Sie hier wie unsere Studierenden in Media Systems von einem Anwalt unterrichtet werden, dann können Sie sich freuen, denn der kann Ihnen nicht nur erklären, wie die Rechtslage auf dem Papier ist, sondern auch wie tatsächlich im Gerichtssaal entschieden wird und was all diese Gesetzestexte bedeuten.

6.7.5 URLs – absolute und relative Adressen

Wenn Sie im WWW unterwegs sind, rufen Sie Seiten wie `www.haw-hamburg.de` auf. Eine solche Adresse wird als Uniform Resource Locator (kurz **URL**) bezeichnet. Aber auch wenn das Protokoll angegeben wird (wie bei `http://www.haw-hamburg.de`) und in einer Reihe weitere Fälle spricht man von einer URL. Auch „Adressangaben“, die sich auf Dateien auf Ihrem Computer beziehen werden als URL bezeichnet. Kurz gesagt ist eine URL eine standardisierte Angabe darüber, wo eine Datei zu finden ist.

Eine **absolute URL** ist nun eine URL, die den vollständigen Pfad zu einer Datei angibt. (Zur Erinnerung: Das WWW ist nichts als eine Ansammlung von Dateien auf Rechnern, die weltweit vernetzt sind.)

Im Gegensatz dazu ist eine **relative URL** eine Adressangabe, die den Pfad von dem Standort aus beschreibt, an dem die Datei gespeichert ist, in der die URL einprogrammiert wurde. Meist werden Sie aus einem einfachen Grund mit relativen URLs programmieren: Da die Dateien bei der Programmierung nicht an derselben Stelle gespeichert sind wie später, wenn sie online abrufbar sind, müssten Sie bei absoluten URLs später alle Adressen einmal ändern und würden mit Sicherheit Fehler erzeugen.

Ein „exzellentes“ Beispiel finden Sie unter den Tutorials zu Java. Dort wurden viele Links absolut programmiert. Als dann Java von Sun Systems an Oracle verkauft wurde, wurden nicht alle Links überarbeitet. Heute können Sie einzig aus diesem Grund viele Tutorials nicht aufrufen: Da der Link mit `www.sun.com` beginnt, die entsprechende Seite aber bei `www.oracle.com` liegt, führt der Link ins Leere. Häufig wurden die entsprechenden Tutorials dann auf `www.oracle.com` in einem anderen Verzeichnis als bei `www.sun.com` gespeichert, sodass auch eine manuelle Änderung von `sun` in `oracle` nicht hilft, um das Tutorial zu finden.

Bei URLs, die nicht auf einen Dateinamen enden, suchen Webbrowser automatisch nach einer Datei namens `index.html`. Deshalb ist es wichtig, dass Sie bei einer Webanwendung immer eine Datei `index.html` einprogrammieren. (Ausnahmen sind Webpages und Webanwendungen, die Sie mithilfe eines Frameworks, eines CMS oder anderer „Hilfen“ erstellen.

Aber wenn Sie beispielsweise eine Webanwendung programmieren, die aus mehreren HTML-Dokumenten besteht, dann brauchen Sie für einen Link von einem dieser Dokumente zum anderen nicht die absolute URL angeben. Nehmen wir an, alle HTML-Dokumente Ihrer Seite würden innerhalb eines Verzeichnisses liegen. Dann brauchen Sie nur den Dateinamen des HTML-Dokuments als URL angeben, auf das Sie verlinken wollen. Eine solche URL wird dementsprechend als **relative URL** bezeichnet.

Wichtig ist aber vor allem, dass Sie grundsätzlich verstehen, was eine URL ist und wie sie syntaktisch richtig geschrieben wird.

6.8 Formulare

Formulare sind Bereiche einer Webanwendung, über die Nutzer Daten per Tastatur oder Maus eingeben können. Interaktive Elemente wie Spiele können auch dazu gehören.

Wie Sie wissen können Sie mit HTML lediglich Container definieren, deren Darstellung über CSS festgelegt wird. Wenn Sie dann noch Nutzereingaben verarbeiten wollen, benötigen Sie zusätzlich eine Programmiersprache wie PHP oder JavaScript.

Da Sie also mit HTML alleine eine Nutzereingabe nicht verarbeiten können macht es scheinbar wenig Sinn, in HTML Formulare zu erstellen. Auf der anderen Seite wird ja in HTML definiert, aus welchen Elementen eine Webanwendung zusammengestellt wird. Und da auch Formulare solche Ele-

mente sind bzw. aus solchen Elementen zusammengestellt werden, müssen wir uns bei der Programmierung in HTML mit Formularen beschäftigen.

Hier sind wir dann auch an einem Punkt angelangt, wo die sonst sehr klare Trennung zwischen HTML und PHP verschwimmt: Alles, was mit Formularen zu tun hat müssen sowohl PHP-EntwicklerInnen als auch HTML-EntwicklerInnen beherrschen.

6.8.1 Elemente eines Formulars

Wie alles andere in HTML programmieren wir auch Formulare als Container. Hier ist es der `<form>`-Container.

Wie gewohnt können Sie über das `id`-Attribut ein Formular zu einem Objekt im Sinne des DOM machen, das einen Namen hat und auf das verlinkt werden kann.

```
<form id=registrierung>
<!-- - Hier werden die einzelnen Eingabemöglichkeiten von Nutzern einprogrammiert -->
</form>
```

Der `<form>`-Container ist das Wurzelement für Formulare, so wie der `<html>`-Container das Wurzelement für HTML-Dokumente ist: Alles, was ein Nutzer eingeben darf oder soll wird einfach in Form verschiedener Container in den `<form>`-Container einprogrammiert. Sie können in jedem HTML-Dokument beliebig viele `<form>`-Container programmieren.

Die Grundidee eines Formulars ist, dass NutzerInnen hier verschiedene Angaben machen und Optionen anwählen können, die dann gewissermaßen als ein Paket verarbeitet werden. Deshalb brauchen Sie in jedem Formular ein Element, das einzig dafür da ist, dass der Nutzer bestätigt, dass alle Daten des jeweiligen Formulars abgeschickt werden sollen. Und auch wenn Sie beliebig viele Formulare auf einer Seite programmieren können, sollten Sie möglichst nicht zu viele individuelle Formulare programmieren, da Nutzer sonst mehr damit beschäftigt sind, die vielen Formulare einzeln abzusenden, als damit, das nötige Formular auszufüllen.

Später werden die Eingaben von Nutzern wie beschrieben an Programme weiter gegeben, die z.B. in PHP oder JavaScript programmiert wurden. Diese Datenübergabe steuern Sie dann durch zwei Attribute des `<form>`-Containers: Das `action`-Attribut gibt die URL des Programms an, das steuert, wie mit den Eingaben des Nutzers umgegangen werden soll. Das `method`-Attribut ist für die Übertragung per HTTP wichtig und steuert, wie die Daten an das Programm übertragen werden. Da wir uns momentan auf die Programmierung eines Formulars in HTML kümmern, dessen

Eingaben noch nicht von einem Programm verwendet werden sollen, lassen wir diese Attribute vorerst außen vor.

Wie gewohnt wird die genaue Darstellung bzw. die Anordnung der Elemente im Webbrowser später über CSS programmiert.

Wichtig: Im Gegensatz zu HTML4.01 bietet Ihnen HTML5 eine Vielzahl an Möglichkeiten, damit Sie prüfen können, ob die Eingabe eines Nutzers valide ist. Bei einem Datum ist es dann beispielsweise unmöglich, den 32. Dezember einzugeben. Die entsprechenden Kontrollfunktionen mussten Sie vor HTML5 mithilfe einer Programmiersprache wie PHP oder JavaScript selbst programmieren.

6.8.2 Das name-Attribut und das id-Attribut – Sonderfälle in Formularen

Innerhalb eines `<form>`-Containers erstellen Sie für jede Eingabemöglichkeit einen weiteren Container. Damit Sie die Eingaben später weiter verwenden können, müssen Sie bei vielen Containern ein `name`-Attribut programmieren, dessen Wert innerhalb des `<form>`-Containers nur einmal vorkommen darf.

Wenn Sie sich jetzt wundern, warum hier nicht mehr das `id`- sondern das `name`-Attribut verwendet werden muss: Das `id`-Attribut wurde mit HTML5 so erweitert, dass Elemente einer Webanwendung als Objekte in einer objektorientierten Sprache verwendet werden können. Das `name`-Attribut wird für die Übergabe von Nutzereingaben eines Formulars an ein Programm verwendet. Deshalb gibt es bei Formularen beide Attribute (`name` und `id`).

Bei allen imperativen Programmiersprachen werden Werte gespeichert, indem sie jeweils einer Variablen zugeordnet werden. Eine Variable hat einen Bezeichner und einen Datentyp. In diesem Buch verwende ich den Begriff des Bezeichners auch um Verwechslungen mit dem `name`-Attribut zu vermeiden. Bei den meisten imperativen Programmiersprachen müssen Sie den Bezeichner festlegen, bevor Sie ihn nutzen dürfen. Danach können Sie mit dem Wert der Variable an mehreren Stellen etwas tun. Das macht vor allem dann Sinn, wenn ein Wert sich immer wieder ändern kann oder er an vielen Stellen innerhalb eines Programms geändert werden kann bzw. soll.

Der Typ einer Variablen sagt etwas darüber aus, um was für eine Art von Variable es sich handelt. Für Sie und mich ist es beispielsweise klar erkennbar, ob eine Zeichenfolge nun ein Text ist oder ein Datum, eine Rechenaufgabe oder etwas anderes. Für einen Computer ist das nicht klar. Ein

Computer kann z.B. nicht unterscheiden, ob die Zeichenfolge 10 die Zahl 10 oder die Zahl 2 (binär 10) oder der Text 10 sein soll; all diese Interpretationen werden innerhalb des Computers unterschiedlich gespeichert und verarbeitet. Deshalb gibt es bei der Nutzung von Variablen innerhalb eines Computerprogramms immer einen Typ für jede einzelne Variable.

Wichtig: In Sprachen wie Java wird der Typ für jede Variable vom Programmierer festgelegt und kann sich dann nicht mehr ändern. Diese nicht-Änderbarkeit des Datentyps einer Variablen wird als **statische Typisierung** bezeichnet. Daneben gibt es noch die **dynamische Typisierung**. Hier wird der Typ einer Variablen von der Programmiersprache verwaltet und bei Bedarf geändert. Schon vorweg sei gesagt, dass PHP eine dynamisch typisierte Sprache ist. Beide Verfahren haben unterschiedliche Vor- und Nachteile und keines (!) ist schlecht. Insbesondere ist die Aussage, dass die dynamische Typisierung unsicher sei kompletter Humbug; es gibt eine Art von Sicherheit, die durch die statische Typisierung sicher gestellt wird, aber gleichzeitig ist sie der Grund für eine Vielzahl überflüssiger Fehlermeldungen.

6.8.3 Container für Formulare

Die restlichen Inhalte des Kapitels sollten Sie überfliegen, um sich zunächst einen groben Überblick darüber zu verschaffen, welche Arten von Eingaben HTML5 direkt unterstützt. Normalerweise hätte ich Sie hierfür auf die Webanwendung der W3Schools verwiesen, aber leider sind dort die Tags nach Ihrem Namen und nicht nach der Funktion sortiert. Das ist für Einsteiger eher verwirrend, denn so brauchen Sie eine ganze Weile, um z.B. das passende Tag zu finden, wenn Sie wollen, dass der Browser prüft, ob eine Eingabe ein reales Datum sein kann oder nicht.

Wichtig: Während viele dieser Eingabemöglichkeiten auf einem Rechner nur die Kontrolle durchführen, ob eine Eingabe des Nutzers zum jeweiligen Typ passt, öffnen sich bei Smartphones häufig kleine Fenster, über die Nutzer z.B. ein Datum anwählen können. Auch hier gilt wieder: Vor HTML5 hätten Sie solche Komfortfunktionen noch selbst programmieren müssen, mit HTML5 brauchen Sie sich um eine solche Programmierung nicht zu kümmern. Alles, was Sie hier tun müssen, ist das passende Tag auszuwählen.

Formularfelder für Texteingaben

In diesem Unterabschnitt finden Sie die meisten Typen, die Sie nutzen können, damit Nutzer einen Text oder eine Zahl eingeben können. Wenn ein Tag hier nicht aufgeführt ist, dann liegt das daran, dass Sie wie bei

Formular-Tags in HTML 4.01 etwas programmieren müssten, damit die betroffenen Tags ihre Aufgabe erfüllen. Ein Beispiel ist das Tag, mit dem Sie es einem Nutzer ermöglichen können, nach einem Begriff zu suchen. Denn die Suche müssen Sie dann doch wieder selbst programmieren. Also taucht es hier nicht auf.

Die folgenden beiden Attribute können Sie bei allen Formularfeldern verwenden: (Ausnahmen sind jeweils angegeben, zum Teil können Sie sich das aber auch logisch erschließen.)

- **Das `value`-Attribut**
Bei allen Formular-Containern (außer dem `file`-Container) können Sie mit dem `value`-Attribut eine Antwort eintragen, die der Nutzer aber jederzeit überschreiben kann. Wenn ein Nutzer das nicht tut, wird dieser Wert beim Absenden des Formulars so übertragen, als wenn der Nutzer ihn eingetragen hätte.
- **Das `required`-Attribut**
Sie können festlegen, dass Nutzer einzelne Felder ausfüllen müssen, bevor sie ein Formular absenden können. Dazu programmieren Sie schlicht das Attribut `required`.

Jetzt folgen die meisten der Container, die Sie verwenden können, damit NutzerInnen alphabetische oder alphanumerische Eingaben durchführen können:

- **Kurzer Text**
Zweck: Damit können Nutzer einen kurzen Text von bis zu 20 Zeichen eingeben.
Quellcode: `<input>` (alternativ: `<input type=text>`)
Wenn Sie hier das Attribut `type=password` vergeben, dann wird die Eingabe maskiert; niemand kann also am Monitor sehen, was der Nutzer eingibt. Das ist aber nur ein Schutz gegen neugierige Kollegen, die nicht verfolgen können, welche Tasten der Nutzer drückt. Gegen die meisten Angriffsarten ist es dagegen vollkommen nutzlos. Um Nutzereingaben gegen diese zu schützen müssen Sie im Programm kryptographische Protokolle integrieren und sollten am besten keine Tastatureingaben als Passwort verwenden. Aber das ist ein Thema für Veranstaltungen in höheren Semestern.
- **Langer Text**
Zweck: Mit diesem Container ermöglichen Sie es Nutzern, Texte beliebiger Länge einzugeben. Im Gegensatz zu anderen Containern können Sie hier mit den Attributen `rows` und `cols` die Größe festlegen. Tipp: Auch wenn es im Moment nicht so gut aussieht, nutzen

Sie dazu besser CSS.

Quellcode: `<textarea>`

- URL eingeben

Zweck: Nutzer können eine URL eingeben. Einziger Vorteil gegenüber `type=text` ist die größere Länge.

Quellcode: `<input type=url>`

- Emails:

Zweck: Hier prüft der Browser, ob die Eingabe eine valide Email-Adresse sein kann: Ist das @-Symbol enthalten? Gibt es eine valide Endung? usw.

Quellcode: `<input type=email>`

Beispiel für ein einfaches Formular:

```
<form>
Nutzername: <input name=username required>
E-Mail: <input type=email name=email required>
Webanwendung: <input type=url name=webpage>
Ihr Anliegen: <textfield name=userrequest required>
</form>
```

Formularfelder für Zahleneingaben

Jetzt die Formularfelder, die für verschiedene numerische Eingaben gedacht sind:

- Zahlen eingeben:

Zweck: Hier können Nutzer ganze Zahlen eingeben. Eine Eingabe ist auch per Maus möglich, da zusammen mit dem Eingabefeld noch zwei kleine Schaltflächen eingeblendet werden, über die der Wert erhöht oder gesenkt werden kann.

Für diese diesen input-Typ können Sie die selben Attribute verwenden, die auch beim nachfolgenden input-Typ `range` gelten.

Quellcode: `<input type=number>`

Tipp: Verwenden Sie `number`, wenn Nutzer eine genau Zahl eingeben sollen und `range`, wenn ein grober Wert als Eingabe genügt.

- Zahlen mit einem Schieber auswählen:

Quellcode: `<input type=range min=... max=...>`

Im Gegensatz zum Typ `number` müssen Sie hier die Attribute `min`

und `max` vorgeben, weil Nutzer kein Feld für eine Eingabe erhalten, sondern einen Slider (zu Deutsch Schieberegler), mit dem sie einen Wert anwählen können. Über das Attribut `step` können Sie zusätzlich programmieren, wie groß der Abstand zwischen zwei wählbaren Zahlen sein darf.

- Telefonnummern:

Quellcode: `<input type=tel>`

Zweck: Der Name sagt schon: Damit können Nutzer eine Telefonnummer eingeben. Der Vorteil gegenüber `type=number` besteht darin, dass so auch eine internationale Vorwahl eingegeben werden kann. Wegen des `+` ist das bei `number` nicht möglich. (Streng genommen handelt es sich hier also nicht um einen Typ für Zahleneingaben, aber da die meisten Menschen Telefonnummern als Zahlen betrachten, habe ich es in diesem Abschnitt eingruppiert.)

Hier ein weiteres Beispiel für ein einfaches Formular:

```
<form>
Nutzername: <input name=username required>
E-Mail: <input type=email name=email required>
Telefon: <input type=tel name=phonenumber>
Webanwendung: <input type=url name=webpage>
Ihr Anliegen: <textfield name=userrequest required>
Ihre Dringlichkeit: <input type=range name=importance min=1 max=9>
</form>
```

Formularfelder für Datumsangaben und Zeitpunkte

Zwar setzen sich Datums- und Zeitangaben größtenteils aus Zahlen zusammen, aber bei den folgenden Typen geht es darum, sicher zu stellen, dass die Eingabe(n) von Nutzerinnen reale Zeitpunkte sind.

Wichtig:

Es gibt kein Formularfeld, das Zeiträume aufnehmen kann: Sie können zwar Einen Anfangszeitpunkt und einen Endzeitpunkt als Formularfeld programmieren, aber beide werden auch in HTML5 selbst nicht als Zeitraum, sondern als zwei individuelle Zeitpunkt betrachtet, die keinen Zusammenhang haben. Bei der Auswertung z.B. in PHP müssen sie das ggf. „korrigieren“.

- Datum

Zweck: Bei Smartphones öffnet sich in diesem Fall ein Feld, über das Nutzer ein Datum, wie z.B. ihr Geburtsdatum anwählen können.

Quellcode: `<input type=date>`

- Monat und Jahr
Zweck: Hier können Nutzer Monat und Jahr eingeben.
Quellcode: `<input type=month>`
- Monat und Jahr
Zweck: Hier können Nutzer Woche und Jahr eingeben.
Quellcode: `<input type=week>`
- Uhrzeit
Zweck: Hier kann eine Uhrzeit eingegeben werden.
Quellcode: `<input type=time>`
- Datum und Uhrzeit
Zweck: Zusätzlich zu `type=date` bietet dieser Typ noch die Angabe einer Uhrzeit an. Der Typ `datetime` (ohne `-local`) ist nicht gültig.
Quellcode: `<input type=datetime-local>`

Weiteres Beispiel für ein einfaches Formular:

```
<form>
Nutzername: <input name=username required>
E-Mail: <input type=email name=email required>
Telefon: <input type=tel name=phonenumber>
Webanwendung: <input type=url name=webpage>
Ihr Anliegen: <textfield name=userrequest required>
Ihre Dringlichkeit: <input type=range name=importancy
min=1 max=9>
Für Anrufe teilen Sie uns bitte noch mit, wann wir
Sie am besten erreichen. Von <input type=time
name=callNoEarlierThan> bis <input type=time
name=callNoLaterThan>.
</form>
```

Auswahlmöglichkeiten

Die input-Typen, die Sie bis jetzt kennen gelernt haben, lassen Nutzern eine große Freiheit bei der Eingabe. Einzig das Format (z.B. bei einer Telefonnummer) muss stimmen. Sie als Entwickler können dabei keine Antwortmöglichkeiten fest vorgeben. Für ein Bestellformular bei einem Lieferservice sind diese Formularfelder deshalb nicht ausreichend.

Es folgen Eingabefelder, die dafür gedacht sind, dass Sie Nutzern eine Reihe an Wahlmöglichkeiten anbieten.

Im Gegensatz zu den bisherigen `input`-Containern können Sie hier die Beschriftung einfach dadurch vornehmen, dass Sie sie innerhalb des Containers programmieren. Besser ist die Nutzung des `<label>`-Containers, zu dem wir im Anschluss kommen.

- Checkboxes:

Quellcode:

```
<input type=checkbox name=bezeichner value=wert>
```

Damit programmieren Sie ein Kästchen, das von Nutzern an- oder abgewählt werden kann.

Die Attribute `name` und `value` werden erst dann von Belang, wenn Sie ein Programm entwickeln, mit dem Sie die Nutzereingaben verwenden wollen (für den Moment können Sie das folgende also überspringen):

- Das Attribut `name` kennen Sie bereits.
- Wenn Sie bei einer Checkbox kein `value`-Attribut programmiert haben, dann wird dem Programm, das über das `action`-Attribut des `<form>` festgelegt wurde die Nachricht `bezeichner=on` übertragen. (Respektive der Name, den Sie programmiert haben.)
- Haben Sie dagegen ein Attribut `value` wie oben programmiert, dann wird dem Programm die Nachricht `bezeichner=wert` übermittelt.

Wichtig (ebenfalls erst in Bezug auf Programme, die Nutzereingaben verwalten):

Wenn eine Checkbox nicht angewählt wird, dann wird keine Nachricht an das Programm weitergemeldet. Das mag jetzt überflüssig klingen, aber nehmen wir an, Sie programmieren eine Liste mit solchen Checkboxes, in denen ein Nutzer angeben soll, welche Zutaten er für ein Rezept bereits zu Hause hat. In diesem Fall würden Sie wahrscheinlich das folgende im Programm festlegen: Computer, erstelle eine Einkaufsliste all der Zutaten, die der Nutzer noch nicht hat. Da das Programm aber nur diejenigen Zutaten übermittelt bekommt, die der Nutzer schon hat, funktioniert das so nicht: Es weiß ja gar nicht, welche Zutaten der Nutzer noch nicht hat.

Deshalb sollten Sie keinesfalls Checkboxes programmieren, bei denen es für die weitere Nutzung wichtig ist, dass eine Meldung an das Programm übertragen wird, wonach sie deaktiviert wurde.

- Radio Buttons:

Quellcode:

```
<input type=radio name=bezeichner value=wert>
```

Checkboxen und Radio Buttons verwirren Einsteiger häufig, weil der Unterschied zunächst nicht klar ist. Dabei ist er recht simpel:

- Checkboxen sind für die Fälle gedacht, in denen Nutzer beliebig viele Optionen anwählen dürfen.
(Denken Sie an einen Bestellservice, bei dem Nutzer beliebig viele Beilagen zu einem Gericht auswählen können.)
- Radio Buttons sind dafür gedacht, dass Nutzer sich für eine von vielen Optionen entscheiden müssen.
(Denken Sie hier an ein Reisebüro, bei dem Nutzer sich zwischen erster und zweiter Klasse entscheiden müssen.)

Beispiel für einfaches Formular mit Auswahlmöglichkeiten zum An- oder Abwählen:

```
<form>
Wählen Sie bitte Ihre Beilage:
<input type=radio name=reis value=reis>Reis
<input type=radio name=fries value=fries>Pommes Frites
<input type=radio name=potatoes value=potatoes>
Kartoffeln
</form>
```

Die folgenden Eingabetypen sind Alternativen zu Radio Buttons und Checkboxen: Im Gegensatz zu diesen beiden wird bei den folgenden jeweils eine Menüleiste eingeblendet, aus der NutzerInnen Einträge auswählen können. Ein Sonderfall ist die Farbpalette, aus der NutzerInnen eine Farbe auswählen können.

- Drop-Down-Liste

Zweck: Genau wie Radio-Buttons beschränken Drop-Down-Menüs NutzerInnen darauf, eines von mehreren Angeboten auszuwählen. Der Unterschied besteht darin, dass die Auswahlmöglichkeiten mittels Radio-Buttons vollständig angezeigt werden, während die Optionen (deshalb der Name) einer Drop-Down-Liste nur dann angezeigt werden, wenn Nutzer die Liste angewählt haben.

Quellcode:`<select>`

```
<option>Beschriftung</option>
```

```
<option>...</select>
```

Über den `<select>`-Container legen Sie fest, dass eine Drop-Down-Liste angezeigt werden sollen. Für jeden Eintrag müssen Sie einen `<option>`-Container innerhalb des `<select>`-Containers programmieren. Damit Nutzer einen Eintrag angezeigt bekommen, müssen Sie bei jedem `<option>`-Container einen Text als Inhalt programmieren.

Wenn Sie innerhalb eines Drop-Down-Menüs einzelne `<option>`-Container in einem unter-Drop-Down-Menü versammeln wollen, können Sie dafür den `<optgroup>`-Container verwenden. Einziger Unterschied gegenüber `<option>`-Containern ist, dass Nutzer durch das Anwählen des `<optgroup>`-Containers noch keine Option anwählen. Dieser hat also keinen `value`, sondern sein Name wird über das `label`-Attribut definiert.

Wenn Sie eine `<optgroup>` als nicht anwählbar markieren wollen (z.B. weil der Inhalt noch nicht programmiert ist, dann benutzen Sie dafür das Attribut `disabled`.

- Datalist

Quellcode: `<datalist><option><option>...</datalist>`

Eine Datalist sieht zunächst wie ein Textfeld aus, aber über die `option`-Container erhalten Nutzer gültige Werte angezeigt. Im Gegensatz zur Drop-Down-Liste sind die `<option>`-Container mit dem `value`-Attribut vollständig, es gibt hier also keinen zusätzlichen Container-Inhalt.

- Farbauswahl:

Quellcode: `<input type=color>`

Damit ermöglichen Sie es Nutzern, eine Farbe aus einer Palette auszuwählen. Das könnte beispielsweise nützlich sein, wenn Spieler eine Farbe für Ihre Spielfiguren auswählen sollen. Wie gewohnt nutzen Sie hier das `name`-Attribut, um die Eingabe an ein Programm zu übergeben.

Schalter

Neben den folgenden input types gibt es noch die `<button>`-Container, mit denen Sie dieselben Funktionen realisieren können. Hier lautet meine Empfehlung allerdings, keine `<button>`-Container zu nutzen: Wenn alle Eingabemöglichkeiten als `<input>`-Container programmiert werden, ist es leichter, alle Eingabemöglichkeiten im Code zu finden. (Für den Leis-

tungsnachweis „Projekt 1“ ist die Nutzung von `<button>` deshalb untersagt.

- **Schalter mit Beschriftung**

Zweck: Damit programmieren Sie einen Schalter, über den eine Funktion des Programms aufgerufen wird, das über das `action`-Attribut des `<form>`-Containers festgelegt wurde. Der Name dieser Funktion wird dem Attribut `onclick` zugeordnet. Was Funktionen sind und wie Sie sie mithilfe des `onclick`-Attributs nutzen können erfahren Sie im Kapitel zur Programmierung in PHP.

Quellcode: `<input type=button value="Beschriftung auf der Schaltfläche" onclick=function()>`

- **Schalter mit Bild**

Zweck: Bei dieser Variante programmieren Sie einen Button, der keinen Text, sondern ein Bild enthält.

Quellcode: `<input type=image src="URL eines Bildes" onclick=...>`

Wichtig: Da ein Button wieder deaktiviert wird, wenn Nutzer den Mausbutton loslassen, macht die Programmierung eines `name`- und/oder eines `value`-Attributs hier keinen Sinn. Vielmehr dienen Buttons dazu, dass Nutzer damit bestätigen, dass sie die Eingaben tatsächlich abschicken wollen. Für Sie als EntwicklerIn bedeutet das, dass die Variablen an das Programm übertragen werden, das über das `action`-Attribut des `<form>`-Containers festgelegt wurde.

- **Reset-Schalter**

Zweck: Mit diesem Schalter können Nutzer alle Eingaben löschen.

Quellcode: `<input type=reset>`

- **Absende-Schalter**

Zweck: Mit diesem Schalter übergeben Nutzer die Daten zur weiteren Verwendung durch Ihre Webanwendung.

Quellcode: `<input type=submit>`

Im Gegensatz zu `type=button` und `type=image` rufen Sie hier also keine bestimmte Funktion des Programms auf, sondern durch einen Klick auf diesen Schalter werden sämtliche Eingaben des Nutzers an das Programm übergeben.

Wichtig: Sie brauchen keine zusätzliche Beschriftung zu programmieren. Das übernimmt der Browser für Sie.

Beachten Sie dabei bitte, dass es hier nur um die Eingaben innerhalb eines Formulars geht. Wenn Sie innerhalb eines HTML-Dokuments mehrere `<form>`-Container programmiert haben, benötigen Sie für jeden dieser Container einen `type=submit`. Denn durch diesen werden ausschließlich diejenigen Daten weitergeleitet, die sich im selben `<form>`-Container befinden.

6.8.4 Container für die Gruppierung und Zuordnung von Eingabelementen

Um mehrere Elemente zu gruppieren müssen diese lediglich in einem `<fieldset>`-Container zusammengefasst und durch einen `
`-Container getrennt werden. `
`-Container sind Container ohne Inhalt, die einen Zeilenumbruch bewirken. In HTML5 kann der `/` deshalb auch weggelassen werden. Vor HTML5 waren sie generell sehr wichtig, da sie für die Gestaltung von Belang waren. Aber da das jetzt in CSS geregelt wird, brauchen Sie sie nur noch in seltenen Fällen wie eben der Zeilentrennung innerhalb eines `<fieldset>`.

Eine Überschrift für ein `<fieldset>` programmieren Sie nicht mit einem `<h...>`-Container, sondern mit einem `<legend>`-Container, der im Gegensatz zu den `<h...>`-Containern keine Ziffer enthält: Er „heißt“ immer `<legend>`, nicht `legend1`, `legend2`, `legend3` usw.. Ansonsten gibt es keine Unterschiede zwischen den beiden.

Ein weiterer Container, den Sie benötigen, um Formulare zu programmieren ist der `<label>`-Container. Dieser erzeugt keinen sichtbaren Unterschied, aber er ist wichtig, damit ein Webbrowser z.B. erkennen kann, dass ein Text, der neben einem `<input>`-Container steht als Beschriftung für dieses Eingabefeld gedacht ist. Das wirkt sich ggf. auf die Anzeige aus.

An dieser Stelle ist das `id`-Attribut des Containers wichtig, auf den das Label sich beziehen soll. Denn der `<label>`-Container hat an sich noch keine Bindung zu einem anderen Container. Er bekommt die erst, indem das `for`-Attribut genutzt wird: Dieses bekommt als Wert den Wert des `id`-Attributs desjenigen Containers, auf den das Label sich beziehen soll.

Hier wäre dann ein typisches Formular, wie Sie es für Nutzerregistrierungen verwenden können:

```
<form>
<fieldset>
<legend>Bitte geben Sie Ihre persönlichen Daten ein:
</legend>
<label for=surname>Nachname:</label>
<input id=surname name=surname required >
<br>
<label for=email>E-Mail:</label>
<input type=email id=email name=email required >
<br>
<label for=age>Alter:</label>
<input type=number id=age min=0 max=140 name=age
required >
<br>
<label for=birthdate>Geburtsdatum:</label>
<input type=date id=birthdate name=birthdate
required >
</fieldset>
<input type=submit>
</form>
```

6.8.5 Zusammenfassung

Es gibt in HTML5 deutlich mehr Formularfelder als bei 4.01. Der Grund ist recht einfach: Die neuen Felder prüfen (bis auf `type=text`, `type=button` und ähnliche), ob die Nutzereingabe valide ist. Die Eingabe einer Telefonnummer im Feld für die Eingabe der Mailadresse ist damit ausgeschlossen. Auch unsinnige Eingaben wie der 99. März sind damit unmöglich. Bei HTML4.01 hätten Sie dazu noch umfangreichen Code in PHP bzw. JavaScript programmieren müssen.

6.9 Multimediale Inhalte einfügen

Auch dieser Abschnitt hat sich gegenüber HTML4.01 deutlich geändert. Es gibt vier neue Container, die speziell für die Einbindung von Audio- und Videodateien sowie für ein gutes Layout aller multimedialen Dateien gedacht sind. Der große Unterschied gegenüber HTML4.01 besteht darin, dass Sie jetzt alle Arten von multimedialen Inhalten im Browser abspielen können, ohne dafür ein PHP- oder JavaScript-Programm zu benötigen. Das galt früher nur für Bilder.

Wichtig:

Wir reden hier momentan ausschließlich über anzeigbare oder abspielbare Inhalte. Interaktive Formate wie Flash aber auch die Programmierung interaktiver Inhalte mit Canvas lassen wir momentan außen vor. **Canvas** ist ebenfalls eine Neuerung in HTML5, die die Gestaltung von Bildern und Animationen ermöglicht. Sie brauchen hierzu also kein zusätzliches Programm. Canvas geht aber noch weiter, denn mithilfe von JavaScript können Sie darin vollständige interaktive Anwendungen (also auch Spiele) programmieren.

6.9.1 Bilder

Der ``-Container ist der einzige Container für multimediale Inhalte, den es so bereits vor HTML5 gab. Allerdings gilt hier wie überall, dass Attribute, die bei HTML4.01 fürs Layout genutzt wurden nicht mehr unterstützt werden. (Viele Browser unterstützen sie zwar immer noch, aber wenn Sie wollen, dass Ihre Webanwendung dauerhaft nutzbar ist, dann sollten sie diese Regelung für HTML5 beachten.)

Um eine Bilddatei einzufügen, nutzen Sie den ``-Container. ``-Container haben keinen Inhalt, denn die Bilddatei, die Sie anzeigen lassen wollen wird als Attribut des Containers programmiert.

- Das Attribut `src` erhält als Wert die URL an, unter der die Bilddatei zu finden ist.
Bsp.: `src=bild.jpg`
- Das Attribut `alt` gibt einen Alternativtitel an, der so lange angezeigt wird, wie das Bild noch nicht geladen ist. Es ist vor allem für die Barrierefreiheit wichtig.
Bsp.: `alt="schönes Bild"`
- Die Attribute `width` und `height` geben an, wie breit bzw. hoch das Bild angezeigt werden soll. Sie ordnen hier jedem der beiden eine Zahl zu, die für die jeweilige Größe in Pixeln, also Bildpunkten steht.

Wichtig:

Sie überschreiben mit `width` und `height` damit das Seitenverhältnis des Bildes. Wenn Sie Bilder also für bestimmte Displaygrößen ändern wollen, dann sollten Sie zunächst über ein PHP- oder JavaScript-Programm das Seitenverhältnis berechnen und dann anhand dieser Berechnung dort (also im Programm) die Änderung von Höhe und Breite durchführen.

6.9.2 figure und figcaption

Wenn Sie ein wissenschaftliches Buch aufschlagen, sehen Sie zu jeder ergänzenden Darstellung einen Untertitel. Mit dem `<figcaption>`-Container gibt es in HTML5 die Möglichkeit genau dasselbe in standardisierter Form auf einer Webanwendung zu tun. Dieser Container darf allerdings nur innerhalb eines `<figure>`-Containers verwendet werden.

Nun fragen Sie sich vielleicht, was dieser `<figure>`-Container denn soll, wo es doch bereits den ``-Container gibt. Die Antwort ist recht einfach: Dieser Container ist dafür gedacht jede Art von multimedialen Inhalten standardisiert bereitzustellen. Es ist also egal, ob Sie nun ein Bild, ein Video, eine Audiodatei anzeigen bzw. abspielen wollen; immer nutzen Sie die gleiche Kombination aus `<figure>` und `<figcaption>`, deren Aussehen Sie über ein CSS-Skript definieren.

Und nicht nur dass: Sie können mit dem `<figure>`-Container auch gleich Kombinationen verschiedener multimedialer Dateien erstellen, die im Sinne des semantic web als solche erkannt werden können. Nehmen wir an, Sie haben im Urlaub mehrere Bilder vom Strand geschossen und zusätzlich Aufnahmen vom Meeresrauschen, aus dem Restaurant und von anderen Stellen aufgenommen. Nun wollen Sie als diese Dateien als eine Diashow mit Sound auf Ihrer Webanwendung platzieren. Dann können Sie genau das über einen `<figure>`-Container erledigen. Und im Gegensatz zu HTML4.01 erkennt jeder HTML5-kompatible Browser, dass es sich bei all diesen einzelnen Dateien um eine logische Einheit (eben Ihre Diashow mit Sound) handelt, obwohl es auf dem Rechner mehrere Dateien sind.

Hier ein einfacher `<figure>`-Container:

```
<figure>
<img src=hotel01.jpg alt="Ein Bild des Hotels,
in dem wir die furchtbarsten zwei Wochen
unseres Lebens hatten.">
<figcaption>Bates Motel</figcaption>
</figure>
```

6.9.3 figcaption für Fortgeschrittene

Eine `<figcaption>` kann aber nicht nur einen Untertitel enthalten, sondern zusätzlich bzw. unabhängig von einem Text die URL einer Audiodatei. Dann wird das Bild mit dieser Audiodatei akustisch untermalt. Dazu wird ein `<audio>`-Container verwendet. Wie das geht (und das es sehr leicht zu realisieren ist) sehen Sie in Kürze.

6.10 Weitere multimediale Formate

In diesem Abschnitt erfahren Sie, wie Sie die verschiedensten multimedialen Inhalte in Ihre HTML5-Webanwendung einbinden können. Ein Hinweis vorweg: Zwar bringt HTML5 nur für einige Formate eine Unterstützung mit, aber Sie erfahren gleich, wie Sie auch andere Formate einbinden können.

6.10.1 Einbindung eigener und frei verfügbarer Videodateien

Die Einbindung von Videodateien ist fast genauso einfach wie die Einbindung von Bildern: Nutzen Sie dazu den `<video>`-Container innerhalb eines `<figure>`-Containers. Der Unterschied besteht nun darin, dass Sie mehrere Videodateien einstellen können, von denen der Webbrowser sich eines aussuchen kann. Das ist deshalb sinnvoll, weil Sie sich so nicht darum kümmern müssen, zu prüfen, welches Videoformat vom jeweiligen Webbrowser abgespielt werden kann. Das bedeutet also, dass Sie sich nicht für ein Format wie `.mp4`, `.mov`, `.wmv` usw. entscheiden müssen, sondern Sie können Sie alle nutzen und es ist sogar ideal, wenn Sie ein Video in möglichst vielen Formaten bereitstellen können.

In einem `<video>`-Container können Sie die folgenden Attribute nutzen:

- `controls` regelt, welche Bedienelemente angezeigt werden. Die Standardeinstellungen bewirken, dass ein Play/Pause-Button, eine Zeitleiste und die aktuelle Zeit im Video angezeigt werden. Wenn Ihnen das nicht genügt oder Sie eine Vielzahl an Formaten abspielen wollen (z.B. Flash), dann können Sie im Netz eine Vielzahl an Playern finden, die Sie direkt in den Quellcode Ihrer Webanwendung einbinden können. Suchen Sie dazu schlicht nach „HTML5 Videoplayer“.

Wichtig: Auch wenn Sie lediglich die Standard-Bedienelemente anzeigen lassen wollen, müssen Sie `controls` als Attribut in den `<figure>`-Container einfügen. Sie brauchen dann aber keinen weiteren Wert zuordnen.

- Die Attribute `height` und `width` funktionieren wie bei ``-Containern. Es gelten die selben Hinweise wie dort.
- Innerhalb des `<video>`-Containers erstellen Sie für jede Videodatei einen `<source>`-Container. In diesem geben Sie zum einen die URL der jeweiligen Videodatei über das `src`-Attribut und zum anderen das Kompressionsverfahren über das `type`-Attribut an.

Kompressionsverfahren dienen dazu, um aus einer großen Audio- oder Video-Datei eine kleinere Datei zu erstellen. Ob das zu sichtbaren Qualitätsverlusten führt, hängt vom Verfahren ab. Das bekannteste Kompressionsverfahren für Audio-Dateien ist unter der Abkürzung **mp3** bekannt.

- **Wichtig:** Nichts ist für einen Nutzer ärgerlicher als wenn er nicht weiß, warum etwas nicht funktioniert. Deshalb können Sie als letzten Eintrag im `<video>`-Container einen Text eintragen, der ausgegeben wird, wenn der Webbrowser keines der Formate abspielen kann.

```
<figure controls>
```

```
<video alt="Video über die Herstellung von Büchern">
```

```
<source src=movie.mp4 type=video/mp4>
```

```
<source src=movie.ogg type=video/ogg>
```

Leider kann Ihr Browser keines der Formate abspielen, in dem die Videodatei vorliegt. Bitte prüfen Sie, ob Sie eine Erweiterung installieren können, mit dem Sie eines der folgenden Formate abspielen können:

MP4, Vorbis OGG

```
</video>
```

```
<figcaption>
```

```
Quelle: <a href=http://www.irgendeineseite.de>
```

```
www.irgendeineseite.de</a>
```

```
</figcaption>
```

```
</figure>
```

6.10.2 Anpassungsmöglichkeiten für den Video-Player

Sie wissen, dass Webanwendung von den verschiedensten Endgeräten aus aufgerufen werden: Manche User nutzen ein Smartphone mit einer langsamen Internetverbindung, andere nutzen einen Rechner, der per Kabelanschluss Daten mit bis zum 15 MB/s (entspricht ungefähr einem 100 mbps-Anschluss) herunterladen kann. Es wäre also ungeschickt, wenn Sie auf einer Webanwendung ein Dutzend Videos platzieren und den Webbrowser anweisen, alle vollständig herunterzuladen, egal ob der Nutzer sie nun sehen will oder nicht. Deshalb können Sie das Verhalten des Videoplayers

anpassen, indem Sie die folgenden Attribute bzw. Attributbelegungen in den `<video>`-Container einprogrammieren:

- `preload=none`
Diese Attributbelegung bewirkt, dass der NutzerInnen einen kleinen Platzhalter sehen, der lediglich anzeigt, dass eine Video-Datei zur Verfügung steht. Die Datei selbst wird nicht heruntergeladen, bis NutzerInnen sie anfordern. Diese Option ist gut geeignet, wenn Sie eine Webanwendung mit vielen Videos programmieren wollen, die auch mit einem Smartphone noch übersichtlich sein soll.
- `preload=metadata`
Diese Attributbelegung bewirkt, dass NutzerInnen einen Platzhalter sehen, der im Gegensatz zu `preload=none` auf der Webanwendung so groß angezeigt wird, wie das Video selbst. Das Video wird auch in diesem Fall nicht heruntergeladen, bis NutzerInnen es anfordern. Diese Option ist vor allem für Rechner gut geeignet, wenn auf einer Seite viele Videos platziert werden. Denn selbst bei durchschnittlichen DSL-Anschlüssen kann es sonst mehrere Minuten dauern, bis alle Inhalte einer einzelnen Seite herunter geladen sind. Der Vorteil dieser `preload`-Belegung besteht darin, dass sich das Layout der Seite nicht ändert, wenn Nutzer Videos starten.
- `autoplay`
dürfte selbsterklärend sein: Das Video wird automatisch gestartet, wenn NutzerInnen die Seite öffnen, auf der es eingebunden ist. Dieses Attribut macht in Verbindung mit den beiden eben vorgestellten `preload`-Varianten natürlich keinen Sinn. Dieses Attribut sollten Sie keinesfalls bei multimedialen Dateien verwenden, die eine Audio-Komponente haben, denn im schlimmsten Fall schließen Nutzer Ihre Webanwendung schlicht deshalb, weil sie sich von der Tonspur gestört fühlen.
- `loop`
Wurde das Video einmal gestartet, bewirkt dieses Attribut, dass es immer wieder von vorne beginnt. Das ist vor allem dann sinnvoll, wenn Sie ein Video anstelle eines Bildes in den Hintergrund einer Webanwendung einblenden wollen. Sie können dieses Attribut also mit `autoplay` kombinieren, wenn Sie ein Video anstelle eines Bildes im Hintergrund einer Seite abspielen wollen.
- `poster`
Wird dieses Attribut ohne weitere Zuordnung verwendet, dann wird der erste Frame (quasi das erste Bild) des Videos als Stellvertreter angezeigt. Das macht in Verbindung mit `preload=none` natürlich keinen Sinn.

Als Wert kann diesem Attribut die URL einer Bilddatei zugeordnet werden. Dann wird dieses Bild als Stellvertreter des Videos angezeigt, bis es gestartet wird.

Aufgabe

Finden Sie zwei Einsatzmöglichkeiten für `poster` mit einem Wert gebraucht und missbraucht werden kann.

6.10.3 Einbindung von geschützten Inhalten (Stichwort: DRM)

Mit den beschriebenen Möglichkeiten können Sie Videodateien einbinden, auf die Sie freien Zugriff haben. Aber wie Sie wissen ist das beispielsweise bei Videos auf YouTube nicht der Fall. Wenn Sie sicher sind, dass Sie das Recht dazu haben, dann dürfen Sie solche Videos mit einem `<iframe>`-Container einbinden. Da hier bis auf `allowfullscreen` keine neuen Attribute vorkommen, sollten Sie das folgende Codefragment ohne weitere Erklärungen einbinden können:

```
<iframe src=http://www.youtube.de/...  
allowfullscreen />
```

6.10.4 Der audio-Container

Alles, was Sie beim `<video>`-Container nutzen können und das bei einer Audio-Datei Sinn macht, können Sie genau so bei einem `<audio>`-Container nutzen. Kommen wir also zu den Unterschieden gegenüber einem `<video>`-Container:

- Ein `<audio>`-Container hat in aller Regel kein Bild, also gibt es für den Nutzer keinen sichtbaren Unterschied zwischen den Attributbelegungen `preload=none` und `preload=meta`.
- Wenn Sie ein Bild zu einer Audiodatei anzeigen wollen (oder eine Diashow), dann nutzen Sie dazu das Verfahren, auf das bei der Erklärung zur `<figcaption>` hingewiesen wurde.
- Dem `type`-Attribut müssen Sie natürlich audio-Typen zuordnen.
Bsp.: `type=audio/mp3`
- Das gilt auch dann, wenn ein Type sowohl für Audio- als auch für Video-Dateien existiert.
Bsp.: Das Kompressionsverfahren OGG ist für Audio- und Videoverfahren definiert. Also müssen Sie hier je nach Medienformat `type=video/ogg` oder `type=audio/ogg` angeben.

6.10.5 Close Captions, Untertitel, Einbindung von Webcams usw.

Neben den genannten Möglichkeiten gibt es aber auch noch Dinge wie Untertitel oder Texteinblendungen für Menschen mit beschränktem Hörvermögen. Wir werden diese Möglichkeiten nicht im Rahmen der Veranstaltung behandeln. Wenn Sie hieran interessiert sind, möchte ich Sie auf das Format WebVTT hinweisen, das Ihnen in diesen Fällen eine Vielzahl praktischer Erweiterungen für Ihre Webanwendung anbietet.

Ähnliches gilt für die Nutzung von Webcams, Mikrofonen und anderen Eingabemöglichkeiten für den Nutzer: Alles, was über die Nutzung von Tastatur und Maus hinausgeht ist nicht Teil dieser Veranstaltung. Hier sollten Sie bei Interesse nach dem Begriff `getUserMedia` suchen.

Grundsätzlich sollten Sie jedoch zunächst HTML, CSS und JavaScript beherrschen, bevor Sie sich in diese Bereiche einarbeiten.

6.10.6 Hinweis bezüglich Flash und ähnlichen Formaten

Adobes **Flash** bzw. Shockwave war über Jahre hinweg der Standard, wenn es um das Entwickeln von interaktiven Elementen bzw. Spielen auf einer Webanwendung ging. **JavaScript** bot hier zu wenige Möglichkeiten und einzig **Java** wurde so entwickelt, dass es im Browser nutzbar war. Warum die meisten Entwickler Flash nutzten soll uns an dieser Stelle nicht interessieren; Tatsache ist, dass es den de-facto-Standard für Webanwendungen darstellte. Wie schon oben angesprochen ändert sich das gerade, was nicht zuletzt daran liegen dürfte, dass JavaScript als Standardsprache für HTML5 festgelegt wurde.

Wenn Sie nun Flash-Anwendungen auf Ihrer Seite anbieten wollen, müssen Sie aus diesem Grund einen HTML5-Flash-Player integrieren. Danach suchen Sie genau wie nach HTML5-Video-Playern. Auch die Einbindung funktioniert wieder so ähnlich wie dort. Aber auch hier gilt wieder, dass das kein Thema dieser Veranstaltung ist, sondern lediglich eine Zusatzinformation für interessierte Leser.

Ein weiterer Nachteil von Flash besteht darin, dass Flash im Gegensatz zu den multimedialen Containern von HTML5 nicht per CSS angepasst werden kann. Neben dem Nachteil, dass Flash-Inhalte somit schwieriger ans Layout der Seite anzupassen sind, folgt daraus, dass sie im Regelfall nicht barrierefrei sind.

6.10.7 Hausaufgabe

Kommen wir jetzt zur Datei `index.html`, die Sie zwar in Ihrem Projektordner haben, die aber bislang leer ist. Damit dies eine Startseite für Ihre Webanwendung wird und Sie zwischen den einzelnen Seiten hin- und herwechseln können, programmieren Sie bitte folgendes: (Wenn nicht anders geschrieben programmieren Sie es bitte in der `index.html`.)

- Fügen Sie die Strukturelemente hinzu, die Sie für HTML5-Seiten kennen gelernt haben.
- Internationalisieren und Lokalisieren Sie die Seite.
- Erstellen Sie ein Log-In-Formular im `aside`-Container.
- Erstellen Sie eine neue Seite mit einem Registrieren-Formular, sodass Sie Nutzern später die Möglichkeit geben, sich zu registrieren.
- Erstellen Sie eine Zusammenfassung der geplanten und vorhandenen Inhalte Ihrer Webanwendung im `main`-Container.
- Verlinken Sie an den passenden Stellen auf die entsprechenden Unterseiten.
- Programmieren Sie umgekehrt auf allen bisherigen Seiten Links. Nutzer müssen mindestens die Möglichkeit haben, über einen Link wieder auf die Startseite zurück zu kommen.
- Nehmen Sie Bilder, Videos und Audio-Dateien auf, die zu einzelnen Passagen auf Ihrer Webanwendung passen und stellen Sie diese auf Ihrer Webanwendung ein.
- Achten Sie darauf, dass bezüglich der Barrierefreiheit zumindest die drei Punkte beachtet werden, die Sie oben kennen gelernt haben.
- Erstellen Sie zu wenigstens einer Ihrer Seiten eine Umfrage, bei der Sie auch verschiedene Auswahlmöglichkeiten programmieren.
- Nicht vergessen: Prüfen Sie, ob Sie für Firefox, Safari, IE oder Edge Polyfills nutzen müssen.

6.11 Weitere Formatierungen und Möglichkeiten in HTML

Wie es die Überschrift schon sagt finden Sie hier eine Reihe weiterer Möglichkeiten, um Inhalte auf Ihrer Webanwendung zu programmieren bzw. um die Inhalte genauer zu definieren. Die folgenden Abschnitte haben keine feste Reihenfolge, sondern sollen Ihnen lediglich einen Einblick geben, was Sie in HTML5 (zum Teil aber auch schon in HTML4.01) noch an Möglichkeiten haben:

6.11.1 Spoiler und andere ausklappbare Texte

Kennen Sie das? Sie sitzen mit Freunden zusammen und einer davon erzählt das Ende eines Filmes, den Sie noch sehen wollten. So etwas wird mit dem englischen Begriff Spoiler bezeichnet und um jemanden zu warnen, dass gleich ein Spoiler kommt, gibt es den Begriff Spoiler Alarm.

Nun nehmen wir an, Sie wollen Filmrezensionen auf Ihrer Webanwendung veröffentlichen, wollen aber dass Ihre Leser selbst entscheiden können, ob Sie die Spoiler mitlesen wollen oder nicht. In HTML5 ist das kein Problem. Nun gibt es aber keinen spoiler-Container, sondern Sie benutzen für solche Fälle zwei Container:

- Der `<summary>`-Container enthält eine kurze Beschreibung dessen, worum es geht.
Bsp.: Sie erstellen eine Webanwendung über die Geschichte Kroatiens. Im laufenden Text wollen Sie etwas über den Geburtsort des amtierenden Präsidenten schreiben. Andererseits sind Sie nicht sicher, ob das die meisten Leser interessiert. Also erstellen Sie den folgenden Container:

```
<summary>Über den Geburtsort des kroatischen
Präsidenten</summary>
```
- Anschließend ergänzen Sie nach dem Wort Präsidenten (oder an anderer Stelle innerhalb des `<summary>`-Containers noch einen `<details>`-Container, in dem Sie all das über den besagten Geburtsort schreiben, was Sie für interessant halten.
Hier ein Beispiel für den oben genannten Spoiler-Fall:

```
<summary>Das Ende von Hamlet
<details>Alle sind tot.</details>
</summary>
```

Wenn Sie innerhalb des „Spoilers“ noch weitere Spoiler unterbringen wollen ist das kein Problem; ähnlich wie bei `<article>` und `<section>` können Sie `<summary>` und `<details>` beliebig komplex verschachteln. Dabei müssen Sie lediglich darauf achten, dass der äußerste Container ein `<summary>` ist, und dass Sie die Container jeweils im Wechsel nutzen müssen.

Sprich: Sie dürfen zwar innerhalb eines `<summary>` mehrere `<details>`-Container programmieren, aber keinen weiteren `<summary>`. Den müssten Sie dann wieder innerhalb eines der `<details>` programmieren. Umgekehrt gilt das selbe.

Natürlich müssen Sie auch bei diesen Containern prüfen, ob Sie inzwischen in allen Webbrowsern unterstützt werden und ggf. ein passendes Polyfill einbinden.

6.11.2 Zeitangaben

Wichtig:

Bitte beachten Sie, dass es sich bei dem gleich vorgestellten Container um einen Container handelt, den Sie im Gegensatz zu Formularcontainern wie `<input type=datetime>` und ähnlichen `input`-Containern an beliebigen Stellen innerhalb eines `<body>`-Containers nutzen können. (Zur Erinnerung: `input`-Container bieten NutzerInnen die Möglichkeit, Eingaben durchzuführen.)

Bitte beachten Sie ebenfalls, dass der `<time>`-Container keine Ausgabe im Browser erzeugt, sondern einzig dafür sorgt, dass ein Teil des Dokuments einen Zeitstempel erhält. Das bedeutet, dass ein Browser hier direkt erkennen kann, dass ein bestimmter Bereich etwas mit einem bestimmten Zeitpunkt zu tun hat. Wenn also die EntwicklerInnen des Browsers den `<time>`-Container richtig auswerten lassen, dann können Nutzer sich einen Termin direkt aus einer Webanwendung in den eigenen Terminkalender eintragen lassen, ohne dass Sie als EntwicklerIn der Webanwendung dafür etwas programmieren müssten.

Mit dem `<time>`-Container können Sie also Zeitangaben und Zeiträume im Sinne des semantic Web definieren. Leider ist es nicht möglich, dass damit alle möglichen Zeitangaben darstellbar wären, denn im Kern wird hierdurch ein Element definiert, dass einen Zeitpunkt oder einen in Sekunden messbaren Zeitraum festlegt. Genau wie bei den entsprechenden `input`-Containern haben Sie also keine Möglichkeit, einen Zeitraum mithilfe eines einzelnen Containers zu programmieren.

Zur Erklärung:

Da eine Zeitangabe wie die vom 20. Februar bis zum 3. März nicht eindeutig ist (denken Sie an Schaltjahre, bei denen es einen 29. Februar gibt), gibt es auch keinen einzelnen `<time>`-Container, mit dem Sie diesen Zeitraum zusammen fassen können. Aus dem gleichen Grund können Sie Zeiträume nicht in Monaten oder Jahren festlegen; in solchen Fällen müssen Sie zwei `<time>`-Container programmieren, den einen für den Anfang, den anderen für das Ende des Zeitraums. Sie haben hier zwar die Möglichkeit, die Dauer als eigenständigen Container einzuprogrammieren, aber da Sie den Zeitraum explizit angeben müssen, könnten Sie hier einen Fehler einpro-

programmieren, der für NutzerInnen ärgerlich wäre.

So viel zum Negativen, kommen wir jetzt zur praktischen Anwendung von `<time>`:

- `<time>`-Container können beliebige Inhalte haben, können aber auch ohne Inhalt als `<time />` beendet werden.
- Um Zeitpunkte oder Zeiträume zu definieren wird unabhängig vom Inhalte des Containers das `datetime`-Attribut verwendet. Für dieses gibt es eine Vielzahl an Möglichkeiten, Werte zuzuordnen:
 - Hier die Varianten für Zeitpunkte:
 - * Für Jahre:
Eine vierstellige Zahl
`datetime=1905`
 - * Für Jahr und Monat:
Eine vierstellige Zahl, ein Bindestrich, eine zweistellige Zahl
`datetime=2107-03` für März 2107
 - * Für ein Datum ohne Uhrzeit:
Zusätzlich eine weitere zweistellige Zahl, verbunden per Bindestrich
`datetime=2015-09-15` für den 15. September 2015
 - * Für Monat und Tag:
Zwei zweistellige Zahlen, verbunden durch einen Bindestrich.
`datetime=12-08` für den 8. Dezember
 - Für die Uhrzeit gibt es mehrere Varianten:
 - * Zeitpunkt ohne Angabe der Zeitzone:
`datetime=17:22`
 - * Zeitpunkt für GMT: `datetime=22:13Z` (Hinweis: Richtig gesehen: Einzig durch das Z am Ende des Wertes wird hier eine Uhrzeit als Zeitpunkt nach GMT festgelegt.)
 - * Zeitpunkt für eine andere Zeitzone:
`datetime=02:18-05` (Für GMT - 5)
`datetime=14:47+5:30` (für GMT + 5½)
 - * Datum und Uhrzeit können verbunden werden, indem zunächst das Datum, dann nach einer Leerstelle die Uhrzeit aufgeführt wird:
`datetime="2017-07-21 20:15-7"` (für 20.15 Uhr in der Zeitzone GMT-7 am 21. Juli 2017)

Wie oben beschrieben können Sie eine Angabe wie „Vom 2. bis 7. März“ nicht direkt als einen Container programmieren. Aber Sie können die Dauer eines Termins als einen Container programmieren:

- `datetime=P30M` entspricht einem Zeitraum von 30 Minuten.
- `datetime="P5D 20M 7S"` entspricht einem Zeitraum von 5 Tagen, 20 Minuten und 7 Sekunden.
- Wie oben aufgeführt können keine Zeiträume in Monaten oder Jahren definiert werden.

Hier noch ein paar Beispiele für `<time>`-Container in HTML:

- Die Veranstaltung dauert voraussichtlich
`<time datetime="P5D 20M 7S"> mehr als 5 Stunden 20 Minuten</time>`.
- Der erste Termin findet am
`<time datetime="2015-09-14 13:00Z">`
14. September 2015 um 13 Uhr</time> statt.
- Veranstaltungen finden vom `<time datetime=2015-09-15>`
15. September</time> bis zum `<time datetime=2016-02-03>`
3. Februar</time> statt.

6.11.3 Hervorhebung von Texten

Bei HTML4.01 wurden Textpassagen meist mit Fettdruck, Unterstreichungen oder Kursivschrift hervorgehoben. Die entsprechenden nicht-semantischen Container lauten schlicht `` oder `` für Fettdruck, `<u>` für unterstrichen und `<i>` (Englisch für italic bzw. kursiv).

Diese können weiterhin verwendet werden. Alle drei haben jedoch Nachteile, wenn die Seite von Personen mit eingeschränktem Sehvermögen genutzt werden oder das Display veraltet ist. Außerdem werden Hyperlinks in Webbrowsern in aller Regel als unterstrichener Text präsentiert.

Deshalb bietet HTML5 den `<mark>`-Container, dessen „Attribute“ `background-color` und `color` per CSS angepasst werden können.

Bei allen vier Containern müssen Sie also lediglich eine Textpassage, die Ihnen wichtig erscheint mit dem öffnenden und schließenden Tag des jeweiligen Containers umschließen.

6.11.4 Unterdrückung von Übersetzungen für Textpassagen

Aktuelle Browser bieten häufig die Übersetzung von Webanwendung aus anderen Sprachen an. Dazu ist das `lang`-Attribut des `<html>`-Containers ein wichtiger Hinweis. Doch wenn Sie wollen, dass bestimmte Stellen nicht

übersetzt werden sollen, dann können Sie diese durch das Attribut `translate=no` von der Übersetzung ausschließen.

Wenn das nur für einzelne Wörter gilt (z.B. bei Eigennamen wie Müller oder Excel), dann können Sie den ``-Container verwenden: Dieser ändert die Formatierung des Inhalts zunächst nicht und kann genutzt werden, um beliebig wenige Zeichen innerhalb anderer Container abzugrenzen:

```
<p>
...
<span translate=no>Tony Marshall</span> sang während
<span translate=no>Andy Müller</span> ein Tor schoss.
...
</p>
```

Vererbung von Attributen

Wichtig: Attribute gelten für den gesamten Bereich eines Containers, also auch für alle Container, die sich darin befinden. Dieses Konzept werden Sie in umfangreicherer Form kennen lernen, wenn Sie einen Kurs zur objektorientierten Programmierung belegen.

Im folgenden Quellcode haben wir solch einen Fall: Das Attribut `translate=no` wird für den äußeren `<article>`-Container deklariert. Damit gilt das Übersetzungsverbot auch in allen Containern, die sich innerhalb dieses `<article>`-Containers gelten:

```
<article translate=no>
(Einleitender Text) ...
<section> ... Sein Vater war von Beruf Müller.
</section>
(Noch mehr Text)
</article>
```

In diesem Fall würde der gesamte Satz „Sein Vater war von Beruf Müller.“ nicht übersetzt werden.

Aber Sie können innerhalb von Containern weitere Container programmieren, in denen Attribute überschrieben werden. Stellen wir uns dazu vor, Sie wollen auf Ihrer Webanwendung einen lateinischen Text im Original veröffentlichen und einige Kommentare dazu posten. Dann möchten Sie natürlich nicht, dass die lateinischen Passagen übersetzt werden, während das bei den Kommentaren sinnvoll wäre. Hier ein entsprechender Quellcode:

```
<article translate=no id="Carmina Burana mit Kommentar">
<p> (Originaltext) <span translate=yes>An dieser Stelle
scheint im Text von ... ein Übersetzungsfehler
vorzukommen, denn ... </span> ... (Fortsetzung des
Originaltexts) ... </p>
</article>
```

Der „Originaltext“ und die „Fortsetzung des Originaltexts“ werden beiden nicht übersetzt, weil Sie Teil des `<article>`-Containers sind, für den die Übersetzung mittels des `translate=no` Attributs unterdrückt wird.

Dagegen wird der Text „An dieser Stelle ...“ übersetzt, weil er Teil des ``-Containers ist, für den die Übersetzung mittels des `translate=yes` Attributs explizit erlaubt wird.

Hier sei nochmal darauf hingewiesen: Das `translate=yes` Attribut gilt nur innerhalb des ``-Containers. Anschließend gilt es dann nicht mehr.

6.11.5 Aufzählungen (Ordered und Unordered Lists)

Wenn Sie eine Aufzählung von Elementen erstellen wollen, wie Einkaufslisten, To Do Listen oder ähnliches dann wird das in HTML als unordered List `` bezeichnet.

Wollen Sie dagegen eine Liste erstellen, bei der die einzelnen Einträge z.B. nummeriert sind, um die Reihenfolge anzugeben, dann wird das in HTML als ordered list `` bezeichnet. Wenn Sie die Art der Nummerierung ändern wollen (z.B. in Großbuchstaben statt Zahlen), dann können Sie dazu das `type`-Attribut nutzen.

Die einzelnen Einträge werden dann als ``-Container innerhalb eines ``- oder ``-Containers programmiert. Wenn Sie also zwischen einer ordered list und einer unordered list wechseln wollen, müssen Sie nur einen Buchstaben im öffnenden und im schließenden Tag ändern, der Rest bleibt gleich:

```
<ol>
<li>Michael Schumacher</li>
<li>Damon Hill</li>
<li>Jacques Villeneuve</li>
</ol>

<ul>
<li>5g Hefe</li>
```



```
<li>3 Eier</li>
<li>100ml Wasser</li>
</ul>
```

6.11.6 Glossare (Description Lists)

Manchmal benötigen Sie dagegen eine Listenform, bei der Sie Begriffe und Ihre Bedeutung aufzählen wollen. In dem Fall sind `` und `` nicht geeignet. Hier greifen Sie am besten auf eine description list `<dl>` zurück.

Innerhalb des `<dl>`-Containers verwenden Sie dann einen description title `<dt>`-Container, um den Namen des Eintrags festzulegen und anschließend einen description description `<dd>`-Container, um die Erklärung einzuprogrammieren: (Ob das `dd` tatsächlich für ein doppeltes description steht, ist nicht sicher, eine Erklärung konnte ich leider nicht finden.)

```
<dl>
<dt>Kaffee, schwarz</dt>
<dd>Heißes Getränk, koffeinhaltig, häufig im Becher
von Herrn Alpers zu finden.</dd>
<dt>Bohnesuppe</dt>
<dd>Kohlenhydrathaltiges Gericht, häufig in Italo-
Western von Bud Spencer konsumiert.</dd>
</dl>
```

6.11.7 Tabellen (table)

Tabellen sind einer der wenigen Container, mit denen Sie auch unter HTML5 ohne CSS das Layout einer Seite festlegen können. Diese sollten Sie aber aufgrund der vielen verschiedenen Displaygrößen von Endgeräten nur dann einsetzen, wenn es sich nicht vermeiden lässt. In dem Fall sollten Sie auf jeden Fall per PHP, JavaScript oder mittels einer anderen Sprache die Größe der Tabelle dynamisch anpassen.

Eine Tabelle definieren Sie durch einen `<table>`-Container. Für jede Zeile definieren Sie darin einen table row `<tr>`-Container, in dem Sie für jede Spalte einen `<td>`-Container programmieren.

Wenn Sie eine Spaltenüberschrift vergeben wollen, verwenden Sie anstelle des `<td>`-Containers einen table header `<th>`-Container.

Der folgende Quellcode soll Ihnen verdeutlichen, warum es wichtig ist, Quellcode übersichtlich zu programmieren, so wie Sie das bei den bisherigen Codebeispielen gesehen haben, denn hier ist das eindeutig nicht der

Fall: Benutzen Sie dazu Zeileneinzüge (Tabulatoren) und Zeilenumrühe (Enter Taste).

```
<table><tr><th>Uhrzeit</th><th>Montag</th><th>Dienstag</th><th>Mittwoch</th><th>Donnerstag</th><th>Freitag</th></tr><tr><td>8.30-10.00</td><td>PRG</td>...</tr>...</table>
```

6.11.8 Microdata

Microdata sind zwar eine zentrale Säule des semantic Web, aber aufgrund des Umfangs dieser Veranstaltung können wir sie uns nur sehr kurz ansehen.

Zur Erinnerung: Microdata haben nichts mit der Darstellung oder der Struktur einer Webanwendung zu tun, sondern Sie dienen dazu, dem Browser anzuzeigen, welche Bedeutung einzelne Elemente der Seite haben. Hier ein paar Anwendungsfälle:

- Der Browser soll eine Adresse ohne weitere Programmierung an eine Anwendung wie google maps weiterleiten können.
- Sie wollen, dass der Browser einen Termin in den Kalender Ihres Mail-Programms übertragen kann.
- Sie veranstalten ein Event, erstellen eine Webanwendung dazu und wollen, dass eine Suchmaschine erkennt, dass es sich um ein Event handelt.

All diese Fälle und noch wesentlich werden bislang unter Begriffen wie **search engine optimization** (kurz **SEO**) und **machine-readable content** zusammen gefasst. Microdata sind ein Mittel, das in HTML5 unterstützt wird, um diese Fälle zu lösen.

Programmierung von Microdata

Dazu müssen Sie als erstes einen Container mit dem Attribut `itemscope` deklarieren. Dieses Attribut ändert wie geschrieben nichts an einer Webanwendung, sondern er teilt dem Webbrowser mit, dass es in diesem Container Elemente im Sinne des semantic web geben kann.

Für diejenigen, die bereits ein wenig Erfahrung mit der objektorientierten Programmierung haben: Damit deklarieren Sie diesen Container explizit

zu einem Objekt, das Sie in einer Sprache wie JavaScript wie ein vollwertiges Objekt verwenden können.

Für alle anderen: Ein Objekt im Sinne der klassenbasierten objektorientierten Programmierung besteht im Grunde aus folgenden abstrakten Bestandteilen:

- Einem Namen oder Bezeichner, der für jedes Objekt individuell sein muss.
Die Lösung in HTML kennen Sie bereits; es ist das `id`-Attribut. Sie brauchen aber vorerst keine `id`-Attribute zu vergeben, weil wir an dieser Stelle nur Microdata programmieren. Die Änderung von Microdata können Sie durch eine Programmiersprache wie PHP oder JavaScript durchführen.
- Einer beliebigen Anzahl an Eigenschaften, die bei unterschiedlichen Objekten des gleichen Typs unterschiedlich ausgeprägt sein können. Damit beschäftigen wir uns gleich. Hier ein Beispiel: Eigenschaften können z.B. Telefonnummern sein. Und dass unterschiedliche Elemente unterschiedlich ausgeprägt sein können, bedeutet bei Elementen vom Datentyp Telefonnummer nicht anderes, als dass unterschiedliche Telefonnummern schlicht unterschiedliche Zahlenkombinationen sind.
- Einer Datenstruktur oder einem Datentyp, die für jede Eigenschaft individuell beschreibt, um was für eine Eigenschaft es sich handelt. Das klingt schwieriger, als es ist; in HTML bedeutet es nur, dass wir damit sagen, dass ein bestimmter Text der Name eines Ansprechpartners ist, dass ein anderer Text seine Anschrift ist, usw.
- Einer beliebigen Anzahl an Methoden (alte Bezeichnung: Funktionen), mit denen die Eigenschaften geändert werden können.

In HTML haben Sie keinen Zugriff auf Methoden. Diese sind Teil von Programmiersprachen wie JavaScript. Dementsprechend beschäftigen wir uns damit vorerst nicht.

Festlegung des Objekttyps

Gerade haben Sie gelernt, dass Sie das Attribut `itemscope` verwenden müssen, um festzulegen, dass ein Container Microdata enthalten soll.

Dann müssen Sie festlegen, welche Art von Microdata das sein soll. Zwar könnten Sie hier auch willkürlich eigene Typen festlegen, aber damit hätten

Sie die Idee der Microdata ad absurdum geführt, denn was Sie sich bei eigenen Typen denken, kann kein Webbrowser wissen, also wäre es weitestgehend zwecklos, so vorzugehen.

Auf der Seite <http://schema.org/docs/schemas.html> können Sie eine Vielzahl an sogenannten **Schemata** (das sind unsere Objekttypen) nachschlagen.

Wenn Sie nun eine Schema gefunden haben, dass Ihnen gefällt, dann programmieren Sie es mit dem Attribut `itemtype` in den entsprechenden Container. Im folgenden Beispiel haben wir das Schema für eine Person verwendet, die u.a. Möglichkeiten anbietet, um eine Anschrift als Microdata zu programmieren:

```
<p itemscope itemtype=http://schema.org/Person>
</p>
```

Wie Sie sehen haben wir hier noch keinerlei Angaben zur Person selbst einprogrammiert. Dieser Container würde auf einer Webanwendung also nicht sichtbar sein und er würde vorerst auch noch keinen Zweck erfüllen. Aber dieser Schritt ist wichtig, weil der Webbrowser sonst nicht wissen kann, was er mit den Microdata anfangen soll, die in diesem Container auftauchen.

Eigenschaften von Objekten

Nehmen wir an, der Name der Person lautet „Martin Schinken“. Für unsere Microdata benötigen wir jetzt also eine Möglichkeit, um eine Eigenschaft „Name“ zu programmieren und diese als Teil des Objekts einzuprogrammieren.

In HTML5 wird das wieder über ein Attribut eines Containers erledigt. Das Attribut lautet `itemprop` (kurz für the items property). Nun gibt es wieder eine Vielzahl an möglichen Arten von Eigenschaften, also müssen wir diese gleich festlegen.

Hinweis: Wenn Sie wie in diesem Fall erfahren, wofür ein „Befehl“ einer Programmiersprache steht (hier `itemprop` für the items property), dann setzen Sie in einen Programm bitte nicht diese Langform (hier „the items property“) ein, denn nur der „Befehl“ (hier `itemprop`) ist ein gültiger Teil der Programmiersprache. Die Langform soll Ihnen lediglich als eine Eselsbrücke dienen.

Aufgabe:

- Schlagen Sie nach, wie die `itemprop` heißt, die für den Namen einer Person verwendet wird.

Jetzt müssen wir nur noch einen Container innerhalb des `<p>`-Containers programmieren, der das Attribut `itemprop` mit dem Wert beinhaltet, den Sie gerade nachgeschlagen haben.

Der Quellcode sollte jetzt also so aussehen: (Für die drei Punkte setzen Sie bitte die von Ihnen gerade recherchierte `itemprop` ein.)

```
<p itemscope itemtype=http://schema.org/Person>
<span itemprop=...>Martin Schinken</span>
</p>
```

Aufgabe:

Bei diesem Quellcode sind zwei Fehler enthalten. Der eine ist die fehlende `itemprop`, die Sie nachtragen sollten. Den anderen kennen Sie schon etwas länger. Genauer gesagt sind in diesem Quellcode also nicht zwei Fehler enthalten, sondern vielmehr fehlen hier zwei Einträge. (Tipp: Wäre der Name Martin Holz oder Martin Hammer, dann wären es ebenfalls zwei Fehler. Beim Namen Martin Borowski dagegen würde der zweite Fehler nicht auftreten.)

- Ergänzen Sie den Code so, dass die beiden Fehler bereinigt werden.

Umfangreiche Microdata am Beispiel einer Person mit Adressangabe

Aufgabe:

Auch das nachfolgende Codefragment ist lückenhaft. Recherchieren Sie, welche Attributbelegungen jeweils Sinn machen:

```
<section itemscope itemtype=http://schema.org/Person>
<h1>Kontakt</h1>
<dl>
<dt>Name</dt>
<dd itemprop= ... >Ihr Name</dd>
<dt>Position</dt>
<dd>
<span itemprop= ... >Student/in</span> an der
<span itemprop= ... > HAW Hamburg</span>
</dd>
</dl>
<div itemprop= ... itemscope itemtype= ... >
```

```
<span itemprop= ... >Hamburger Str. 231</span>
<span itemprop= ... >Hamburg</span>,
<span itemprop= ... >22081</span>
</div>
<h1>Online bin ich aktiv bei:</h1>
<ul>
<li><a href=http://www.twitter.com/ihrTwitterAccount itemprop= ... >Twitt
<li><a href=http://www.blogger.com/ihrBlogAccount itemprop= ... >Webblog<
</ul>
</section>
```

6.11.9 Validator für Microdata

Um zu prüfen, ob Ihre Microdata eindeutig programmiert sind, können Sie auf der folgenden Webanwendung einen Testbereich finden:

<http://developers.google.com/structured-data>

Aufgabe:

Prüfen Sie dort, ob Ihre Lösungen zu den beiden letzten Aufgaben valide sind.

6.12 Zusammenfassung

Sie wissen jetzt:

- wie die Grundstruktur jeder Webanwendung aussieht,
- verstehen was Internationalisierung und Lokalisierung ist,
- wissen um die Bedeutung von meta-Containern,
- kennen die neuen Container header, footer, main, aside, usw.
- und wissen wie Sie Polyfills finden und nutzen können.

Sie wissen außerdem:

- wie Sie Verbindungen zwischen Webanwendung programmieren können, ■
- wie Sie Nutzereingaben in HTML ermöglichen können
- und wie Sie multimediale Inhalte in die Webanwendung einbinden können.

Außerdem verstehen Sie, was das semantische Web ist und wie Sie mittels Microdata eine semantische Webanwendung programmieren können.

Was jetzt noch fehlt und leider nicht Teil dieses Kurses ist, ist die Entwicklung von interaktiven Oberflächen in HTML5 mit Hilfe von Canvas. Denn das programmieren Sie über JavaScript.

Im nächsten Kapitel folgt die zweite Hälfte des Kursteils, in dem sie die Entwicklung statischer Webanwendung kennen lernen: CSS, die Programmiersprache, mit der Sie die Gestaltung einer Webanwendung programmieren.

Kapitel 7

ML, Teil 2 - Textverarbeitung mit LaTeX

Dieses Kapitel ist nicht Teil des Kurses „Einführung ins Programmieren“ oder von „Programmieren 1“, aber Sie sollten es dennoch durcharbeiten. Zum einen lernen Sie hier eine weitere Markup Language kennen und können dadurch einige Gemeinsamkeiten und Unterschiede erlernen. Zum anderen ist LaTeX die weltweit am häufigsten eingesetzte Textverarbeitung für wissenschaftliche Texte. An einigen Hochschulen wird sogar verlangt, dass Arbeiten als LaTeX-Dokumente eingereicht werden.

LaTeX ist aus Sicht der Programmierung eine Markup Language. Im Gegensatz zu HTML handelt es sich hier aber um eine Sprache, die dafür gedacht ist, verschiedenste (vorrangig gedruckte) Dokumente zu erzeugen. Entwickelt wurde sie von Leslie Lamport, einem der Träger des **Turing Award**. (Wem das kein Begriff ist: Der Turing Award wird als Nobelpreis der Informatik bezeichnet.) Im Gegensatz zu TeX, von dem LaTeX eine Erweiterung ist, müssen wir hier nur alles das Programmieren, was anders als beim Standard-Dokument ist.

Dieser Kurs ist eine knappe Einführung. Umfangreichere Kurse werden an den meisten Hochschulen aber auch direkt von lokalen LaTeX-Gruppen angeboten. Im Department Medientechnik bietet beispielsweise Prof. Görne alle zwei Semester eine Schulung an, die für die Mitglieder des Departments kostenlos ist. Hier können Sie natürlich auch individuelle Fragen stellen.

LaTeX hat einige essentielle Vorteile gegenüber den meisten Textverarbeitungen, die Sie aus dem Alltag kennen. Das hier ist nur eine Auswahl:

- Genau wie in HTML kümmern Sie sich um den Inhalt und nicht um die Darstellung.

- Sie können aber jedes Detail anpassen.
- Da Sie direkt im Code arbeiten und auch Einrückungen manuell programmieren kann LaTeX Ihre Arbeit nicht torpedieren.
- Es gibt eine große Community, die im Bedarfsfall helfen kann.
- Es ist vollständig kostenlos.
- Formeln lassen sich sehr einfach eintragen und ändern.
- Überschriften werden wissenschaftliche nummeriert.
- Inhaltsverzeichnisse, Glossare, nummerierte Abschnitte und ähnliches passen sich automatisch an Änderungen an.

Wie bei allen Markup Languages gibt es allerdings einen Nachteil:

- Sie müssen sich in die Programmierung einarbeiten. Aber wenn Sie HTML beherrschen, dann wird das für Sie keinen allzu großen Anspruch darstellen. Streckenweise ist es einfacher als die Einarbeitung in HTML5, da wir hier (genauer gesagt in LaTeX 2) kein semantisches Web haben und uns auch nicht um Aspekte des Responsive Design kümmern müssen.

Zur Erinnerung: Installieren Sie bitte `TeXStudio`, um mit der Programmierung zu beginnen.

7.1 Grundlagen

Das meiste, was Sie an Grundlagen wissen müssen kennen Sie jetzt schon: Sie wissen was eine Markup Language ist. Bei LaTeX gibt es eine andere Syntax und andere Bezeichnungen für Elemente und Container, die müssen Sie also lernen. Außerdem gibt es einige zusätzliche Container, mit denen Sie z.B. ein Inhaltsverzeichnis generieren können. Ähnlich wie in HTML5 gibt es aber nur im Bedarfsfall ein schließendes Tag.

7.1.1 Struktur eines LaTeX-Dokuments

In HTML kennen sie die drei zentralen Container `html`, `head` und `body`.

In Latex haben wir keinen `html`-Container bzw. keinen `latex`-Container und auch keine Doctype Declaration.

Stattdessen sprechen wir von einer **Präambel**. Diese enthält alles, was wir bei HTML als Attribut im `html`-Tag und im `<head>` untergebracht haben. Danach folgt der eigentliche Inhalt des Dokuments, für den wir keine spezielle Bezeichnung haben.

In LaTeX gibt es leider keinen Bezeichner der dem `Tag` in HTML entspricht. Um es Ihnen beim Einstieg einfacher zu machen werde ich den Begriff in diesem Kapitel dennoch verwenden. Allerdings gibt es hier den Begriff des `Elements`, der für eine kleinste Einheit eines Dokuments steht.

7.1.2 Einfache Präambel

Die folgende Präambel beinhaltet alles, was Sie für die meisten Dokumenten benötigen dürften:

```
\documentclass[11pt, a4paper, oneside, draft]{book}

\usepackage{palatino, url}
\usepackage[ngermanb]{babel}
\usepackage[utf8]{inputenc}

\setlength{\parindent}{0cm}
```

Wie Sie sehen nutzen wir in LaTeX keine spitzen Klammern, um Tags anzuzeigen, sondern wir beginnen jedes „Tag“ mit einem Backslash `\`. Bitte beachten Sie den Unterschied: Es handelt sich nicht um den Slash, den Sie mit der Tastenkombination `Shift 7` erhalten, sondern mit der Tastenkombination `Alt gr ß`. Sie müssen dazu also die rechte `Alt`-Taste (die mit `Alt gr` beschriftet sein sollte) drücken, gedrückt halten und zusätzlich das `ß`. Das ist anfangs etwas ungewohnt, gibt sich aber mit der Zeit.

Struktur von Umgebungen

- Jeder LaTeX-Container beginnt mit einem Backslash `\`.
- Danach folgt der Bezeichner.
- Es kann ein paar eckiger Klammern mit einem oder mehreren Einträgen folgen. `[]`
- Es können ein oder mehrere Paare geschweiften Klammern mit jeweils einem oder mehreren Einträgen folgen. `{ }`

Dabei gibt es keine allzu genaue Systematik dafür, was in den geschweiften oder eckigen Klammern steht. Die geschweiften Klammern entsprechen

aber meist dem Inhalt eines Containers.

documentclass

Die `documentclass` gibt in geschweiften Klammern an, um welche Art von Dokument es sich handelt. Daraus folgen einige Standardeinstellungen, die Sie aber auch ändern können. Hier eine Auswahl:

- `book` ist dafür gedacht, um die typischen Elemente eines Buches bereit zu stellen: Teile, Kapitel, usw.
- `report` ist für umfangreiche Reportagen gedacht, also auch beispielsweise für Forschungsberichte, die eine Zusammenfassung und mehrerer Kapitel umfassen können.
- `article` ist für Artikel gedacht, die z.B. in Zeitschriften erscheinen sollen. Auch hier ist z.B. eine Zusammenfassung vorgesehen.
- `letter` ist für Anschreiben gedacht.

Oben habe ich mich für die `documentclass book` entschieden und die folgenden Optionen festgelegt:

- `11pt` legt als Schriftgrad 11 Points fest. Der Standardwert liegt bei 10 pt. Wenn der Ihnen genügt, brauchen Sie also gar keine Angabe zu machen.
- `a4paper` legt fest, dass das Seitenformat Din A4 ist. Außer bei der Ausgabe als pdf-Dokument ist ein amerikanisches Seitenformat hier der Standard.
- `oneside` legt fest, dass alle Seiten an der gleichen Stelle nummeriert werden. Alternativ dazu können Sie `twoside` wählen. Dann werden Seiten abwechselnd links und rechts mit einer Nummer versehen.
- `draft` ist eine praktische Option, da sie dafür sorgt, dass Zeilen, an denen ein Wort in den Seitenrahmen hineinragt mit einem schwarzen Quadrat gekennzeichnet werden. Das ist in sofern praktisch, als Sie schneller sehen, wo Sie ein wenig Feintuning beim Zeilenumbruch machen müssen.

usepackage

Dieser LaTeX-Container ermöglicht es Ihnen Packages zu nutzen, mit denen Sie bestimmte Formatierungen anpassen können. Zum Teil erhalten Sie

dadurch neue Container, die im „Standard-“ LaTeX nicht enthalten sind.

Wenn Sie mehrere Packages nutzen wollen, ohne nähere Optionen auszuwählen, dann können Sie diese durch Kommata getrennt gemeinsam als Argument eines usepackage-Containers verwenden. Bsp.: `\usepackage{palatino, url}` fügt die Packages palatino und url hinzu. palatino ist ein Schriftsatz und url ist ein Container, mit dem Sie URLs im fließenden Text hervorheben können.

Wenn Sie dagegen bei einem Package Optionen festlegen wollen, dann müssen Sie für jedes Package einen eigenen Container programmieren. Bsp.: `\usepackage[ngermanb]{babel}` fügt das babel-Package hinzu, das die Syntaxprüfung für verschiedene Sprachen ermöglicht. Hier müssen wir eine Option wählen, da wir uns für eine Sprache entscheiden müssen. Die Option german entspricht allerdings der alten Rechtschreibung, weshalb mit ngerman eine eigene Option für die seit Ende der 90er Jahre geltenden Rechtschreibregeln entwickelt wurde.

Das Package inputenc kann die verwendete Codierung festlegen. Darüber hatten wir ja bereits bei HTML gesprochen, der entsprechende usepackage-Container sollte damit klar sein.

setlength

Der Container `\setlength{\parindent}{0cm}` ist dann ein Beispiel dafür, dass wir die Vorstellung eines Containers aus HTML nicht direkt in LaTeX übertragen können: Hier haben wir ein „Tag“, das zwei Inhalte hat, zum einen `\parindent`, was für den Einzug der ersten Zeile eines Absatzes steht, zum anderen `0cm` was naheliegender Weise für 0 Zentimeter steht.

Absätze beginnen bei LaTeX mit einem kleinen Einzug in der ersten Zeile und können nur anhand dieses Einzugs erkannt werden. Wenn Sie dagegen Absätze dadurch trennen wollen, dass Sie eine leere Zeile einfügen wollen und keinen Einzug haben wollen, dann müssen Sie in der Präambel diese Zeile einfügen.

7.2 Text, Sonderzeichen und Formeln

Fast alles, was wir von jetzt an kennen lernen wird im body des Dokuments programmiert. Dieser wird ähnlich einem HTML-Container als `\begin{document}` begonnen und mit `\end{document}` beendet.

Wenn Sie dort einen einfachen Text verfassen wollen, dann können Sie direkt damit beginnen: Anders als in HTML gibt es keine expliziten Absatz-Container wie `<p></p>` in LaTeX.

Das Ende eines Absatzes „programmieren“ Sie dadurch, dass Sie einfach die Enter-Taste drücken. Wollen Sie zusätzliche Leerzeilen einfügen, dann müssen Sie `\\` eingeben.

7.2.1 Sonderzeichen

Im wissenschaftlichen Bereich nutzen wir immer wieder Sonderzeichen, die dann bei einer Textverarbeitung wie LaTeX nicht direkt eingegeben werden kann, weil dieses Zeichen dort eine besondere Bedeutung hat. Den Backslash und einige mehr haben Sie schon kennen gelernt. Wenn Sie ein solches Zeichen verwenden wollen oder Teile von Programmtexten einfügen wollen, gibt es unter anderem zwei einfache Möglichkeiten. Die beiden folgenden Varianten funktionieren für alle Sonderzeichen identisch:

- Wenn Sie innerhalb eines Absatzes einzelne Sonderzeichen oder kurze Textpassagen mit Sonderzeichen einbinden wollen, dann nutzen Sie dazu `\verb|Text mit Sonderzeichen|`. Der `\verb`-Container hat eine Besonderheit: Sie können hier jedes Zeichen (also nicht nur geschweifte Klammern) nutzen, um ihn abzugrenzen.

Wenn Sie also das Sonderzeichen `|` verwenden wollen, dann nehmen Sie einfach ein anderes, um den Rahmen des `\verb`-Containers festzulegen. Bsp.: `\verb ~ Text mit Sonderzeichen ~` (Hier wurde `~` als Zeichen verwendet, um den Inhalt des Containers abzugrenzen.)

- Wenn Sie dagegen mehrere Zeilen mit Sonderzeichen anzeigen lassen, dann nutzen Sie die sogenannte verbatim-Umgebung:

```
\begin{verbatim}
Zeile mit Sonderzeichen
Noch eine Zeile mit Sonderzeichen
Noch viele Zeilen mit Sonderzeichen
...
\end{verbatim}
```

Anmerkung: Sollten Sie den seltenen Fall haben, dass Sie innerhalb einer verbatim-Umgebung `\end{verbatim}` eintragen wollen, dann lassen Sie einfach eine Leerstelle zwischen der geschweiften Klammer `{` und `verbatim` stehen.

7.2.2 Formeln

In vielen Texten wird erklärt, dass Sie Formeln in LaTeX mit dem Dollar-Symbol $\$$ abgrenzen. Das funktioniert zwar, allerdings handelt es sich dabei um eine TeX-Anweisung. In LaTeX gibt es für Formeln im Fließtext folgende Zeichensequenz:

```
\( ... Formel ... \)
```

Dabei gilt, das alles, was zwischen $\backslash ($ und $\backslash)$ steht im Sinne des mathematischen Modus interpretiert wird. Der mathematische Modus bietet außerordentlich vielfältige Möglichkeiten, um Formeln einzutragen. Wenn Sie in irgend einem mathematischen Buch ein Symbol sehen, dass in einer Formel verwendet wird, dann gibt es ein LaTeX-Tag, mit dem Sie dieses Symbol im mathematischen Modus erzeugen können.

Wie Sie sich vorstellen können, würde eine Einführung in den mathematischen Modus alleine schon ein Buch füllen. An dieser Stelle werde ich nur auf einige wenige Möglichkeiten eingehen, die Ihnen bei den ersten Schritten helfen werden. Alles weitere können Sie in aller Regel durch eine Recherche im Netz sehr schnell in Erfahrung bringen.

- Für Multiplikationen sollten Sie `\cdot` nutzen. Dadurch wird ein Multiplikationspunkt eingefügt.
- Einen Bruch können Sie mit `\frac{Divident}{Divisor}` darstellen. Divident und Divisor können dabei beliebig komplexe Formeln sein.
- Um eine Potenz darzustellen benutzen Sie den Hochpfeil \wedge . Wie immer gilt: Wenn der Exponent ein Ausdruck ist, dann nutzen Sie die geschweiften Klammern. Bsp.: x^{2+3} programmieren Sie als `x ^ {2 + 3}`.
- Indizes wie $X_{i,j}$ stellen Sie durch `X_{i, j}` dar.

Wenn Sie dagegen mehrere Formeln als eigenständige Absätze anzeigen lassen wollen, dann nutzen Sie dafür die `equation`-Umgebung:

```
\begin{equation}
```

Eine Zeile für jede Zeile der Formel.

In dieser Umgebung gilt wieder der mathematische Modus, wie Sie ihn gerade kennen gelernt haben.

```
\end{equation}
```

7.3 Umgebungen

Gerade haben Sie mit `\begin{verbatim}` und `\end{verbatim}` etwas kennen gelernt, das konzeptionell den Containern in HTML entspricht. Bei LaTeX werden diese Container aber als **Umgebungen** bezeichnet.

Wenn Sie beim Anfangs-„Tag“ einer Umgebung Optionen programmieren, dann gilt hier dasselbe, was Sie bei den Attributen bzw. Attributen mit Wertzuweisung in HTML kennen gelernt haben: Diese gelten für alles, was sich innerhalb der Umgebung befindet.

7.4 Inhaltsverzeichnis, Kapitel und Abschnitte

Wenn Sie längere Texte als einen Brief schreiben, dann benötigen Sie noch Möglichkeiten, um z.B. Kapitelüberschriften einzufügen. Aus diesen Überschriften wird später übrigens das **Inhaltsverzeichnis** an genau der Stelle erzeugt und ins Dokument eingefügt, an der Sie `\tableofcontents` ins Dokument eintragen.

7.4.1 Kapitel, Abschnitte usw.

Bezüglich der Größe und Strukturierung anhand von Überschriften ist LaTeX komfortabler als HTML: Während dort `<h1>` und `<h2>` nur optisch unterschiedlich sind, bewirken `\chapter{Titel}` und `\section{Titel}`, dass der eine von LaTeX als Teil des anderen interpretiert wird.

Wichtig:

Diese Überschriften sind keine Umgebungen, sondern werden wie abgeschlossene Container in HTML programmiert und behandelt. Das bedeutet, dass der Text und die Unterüberschriften z.B. innerhalb eines Kapitels für LaTeX nicht Teil des Kapitels sind. Auch wenn das in aller Regel kein Problem ist müssen wir deshalb selbst darauf achten, welche Teile unserer Texte zu welchem Kapitel gehören.

Hier nun die Hauptstrukturüberschriften in LaTeX:

- `\part[Kurztitel]{Titel}` wird in aller Regel nur bei Büchern genutzt. Es handelt sich hier um eine Überschrift, die den Inhalt mehrerer Kapitel zusammenfasst. (Denken Sie an so etwas wie die Unterteilung eines Mathebuchs in Teil 1 – Algebra, Teil 2 – Geometrie usw.)
- `\chapter` entspricht einem Kapitel.

- `\section` entspricht einem Abschnitt innerhalb eines Kapitels.
 - Um Unterabschnitte zu beginnen, nutzen Sie `\subsection`.
 - Dann gibt es noch die `\subsubsection` usw.
- `\paragraph` ist ein Absatz innerhalb eines Abschnitts, der eine eigene Überschrift erhalten soll. Auch hier können sie mit dem Präfix `sub` wie bei `sections` eine Priorisierung erstellen. Allerdings werden `paragraphs` nicht ins Inhaltsverzeichnis mit aufgenommen und auch nicht nummeriert.

Übrigens können Sie bei all diesen Typen jeweils in den geschweiften Klammern die Überschrift festlegen, die im Text angezeigt wird.

Und in eckigen Klammern können Sie eine Kurzfassung der Überschrift einfügen, die z.B. im Inhaltsverzeichnis angezeigt wird. Wenn Sie dort nichts angeben, wird auch im Inhaltsverzeichnis die vollständige Überschrift übernommen.

7.5 Auslagern von Kapiteln

Je länger Ihr Dokument wird, desto stärker wird der Wunsch werden, einzelne Teile auszulagern, damit Sie etwas übersichtlicher arbeiten können. In HTML war das nur über den Umweg von PHP möglich, in LaTeX ist es einfacher:

Hier verwenden Sie `\include{Dateiname}`, wobei der Dateiname auf `.tex` enden muss. Diese Endung wird aber in der `include`-Anweisung nicht aufgeführt, sondern nur der Dateiname vor dem `.tex`.

7.6 Titelblatt und Glossar

Bei einigen Dokumentklassen ist ein Titelblatt vorgesehen, bei anderen müssen sie über den Eintrag der Option `titlepage` zur `\documentclass` angeben, dass ein Titelblatt hinzugefügt werden soll.

Dass alleine genügt aber noch nicht. Was wir noch brauchen sind die Angaben für das Titelblatt und die Angabe, wo genau das Titelblatt eingefügt werden soll:

Für die nötigen Angaben fügen Sie am Anfang des body (also direkt nach `\begin{document}`) die folgenden LaTeX-Tags ein:

- `\title{}` enthält den Titel, der auf dem Dokument eingeblendet wird.
- `\author{}` enthält den Namen des/der Autoren.
- `\date{}` enthält ein Datum.
 - Dabei können Sie mit `\date{\today}` automatisch das aktuelle Datum eintragen.

Wenn Sie diese Angaben eingetragen haben, können Sie per `\maketitle` an beliebigen Stellen im Dokument ein Titelblatt erzeugen und einfügen lassen.

7.7 Glossar

Um ein Glossar bzw. Stichwortverzeichnis zu generieren müssen Sie leider deutlich mehr Arbeit aufwenden, aber wenn Sie das getan haben, wird genau wie beim Inhaltsverzeichnis ein Verzeichnis für Schlagwörter automatisch generiert und aktualisiert.

1. `\makeindex` muss zur Präambel hinzugefügt werden.
2. `\usepackage{makeidx}` kann zusätzlich in der Präambel aufgenommen werden, um die Darstellung des Stichwortverzeichnisses anders zu gestalten.

Wenn Sie das erledigt haben, müssen Sie an der Stelle des Dokuments, wo das Stichwortverzeichnis eingefügt werden soll, die folgenden Zeilen einfügen:

- `\renewcommand{\indexname}{Stichwortverzeichnis}` legt den Titel des Stichwortverzeichnisses fest. (Es gibt eine Standardbezeichnung.)
- `\addcontentsline{toc}{chapter}{Stichwortverzeichnis}` ■ stellt sicher, dass das Stichwortverzeichnis ins Inhaltsverzeichnis aufgenommen wird.
- `\printindex` fügt an der Stelle, an der es steht das Stichwortverzeichnis in das Dokument ein.

7.7.1 Stichwörter und Bezüge festlegen

Jetzt kommt der Teil, der die eigentliche Arbeit für das Glossar ausmacht: Jeder Begriff, der im Glossar aufgenommen werden soll muss mit `\index{Bezeichnung im Glossar}` aufgenommen werden.

Bsp.: Sie wollen einen Verweis auf eine Textpassage festlegen, in der es um die Nutzung von Dampfmaschinen im Allgemeinen geht. Das sähe dann so aus:

```
Bis zu einem dem Autor nicht bekannten und  
aus Faulheit nicht recherchierten Zeitpunkt  
waren Dampfmaschinen\index{Dampfmaschine}  
eine recht weitverbreitete Antriebsart. ...
```

Wenn Sie dabei Haupt- und Unterstichwörter nutzen wollen, dann sieht das so aus: `\index{Hauptstichwort!Unterstichwort}`.

Bsp.: Sie schreiben über die Programmierung in C und wollen einen entsprechenden Verweis im Stichwortverzeichnis erzeugen. Das sähe so aus:

```
C\index{Programmiersprache!C} ist eine  
kompilierte Sprache ...
```

Doch während das Inhaltsverzeichnis von LaTeX automatisch generiert wird, müssen wir die Generierung des Glossars manuell starten. TeXStudio hat dafür einen Assistenten, den sie per F12 oder über den entsprechenden Eintrag im Menü Assistenten starten können.

7.7.2 Weitere Verzeichnisse

Später werden Sie erfahren, wie Sie Abbildungen und die sogenannten Figures in LaTeX programmieren können. Für diese können Sie ähnlich wie beim Inhaltsverzeichnis Verzeichnisse automatisch generieren lassen. Dazu nutzen Sie `\listoffigures` und `listoftables`.

7.8 Listen und Tabellen

Immer wenn Sie mehrere Einträge gruppieren und/oder sortieren wollen, nutzen Sie sogenannten Listen bzw. Tabellen.

Aus HTML kennen Sie die `unordered` und `ordered` lists sowie die `description` lists. Und alle drei werden (wenn auch mit einer etwas anderen Syntax) nahezu identisch in LaTeX umgesetzt:

Einzelne Einträge werden durch ein vorangestelltes `\item` gekennzeichnet. Als Option können Sie hier noch Labels vergeben.

Alle Listen werden als Umgebung (also mit `\begin{}` und `\end{}`) programmiert. Die Argumente lauten dabei:

- `itemize` (für Listen mit Punkten)
- `enumerate` (für nummerierte Listen)
- `description` (für Glossare)

7.8.1 Tabellen

Hier haben wir einen Unterschied gegenüber HTML: Dort ist eine Tabelle ein Rahmen, anhand dessen wir Elemente unsers Dokuments an einer festen Position im Dokument anordnen können.

In LaTeX dagegen wird zwischen einem `table` und einem `tabular` matter unterschieden. Das was wir umgangssprachlich als Tabelle bezeichnen ist in LaTeX die **tabular matter**.

Die unterschiedlichen Bezeichnungen basieren darauf, dass LaTeX hier den Begriff Tabelle so nutzt wie das bei Schriftsetzern üblich ist. Die differenzieren hier genauer als wir das umgangssprachlich tun.

Um klar zwischen umgangssprachlicher Tabelle und den genannten formalen Tabellen bzw. tabellarischen Aufstellungen zu unterscheiden wird hier bei letzteren die englische Bezeichnung gewählt.

Hinweis, wenn Sie mehr dazu im Netz recherchieren wollen: Leider ist der Begriff des `formal table` im Englischen doppelt belegt: Dort wird er häufig für eine formale Eindeckung eines Tisches verwendet. Wenn Sie also nach `formal table` suchen, werden Sie fast ausschließlich Anleitungen für Diener finden. Bei den verbleibenden Einträgen steht in aller Regel nur, dass es sich dabei um etwas anderes handelt, als das, was wir umgangssprachlich als Tabelle bezeichnen. Eine Aussage, die uns im Regelfall gar nicht weiterhilft.

formal table

Eine Tabelle (**formal table**) in LaTeX ist dagegen eine Umgebung, die den Titel einer Tabelle und ein Label enthalten muss, auf das wir von anderen Stellen des Dokuments aus verweisen können.

Es handelt sich also um etwas, das weitgehend dem `<figcaption>`-Container in HTML entspricht. Es unterscheidet sich allerdings davon, weil ein `formal table` (`table`-Umgebung) in LaTeX eine Umgebung für beliebige Inhalte außer für Bilder, Videos und Audiodateien wie in HTML ist.

Hier ein Beispiel:

```
\begin[wie immer optional: Buchstabe, der die
Position des formal table festlegt]{table}
\caption[Kurztitel]{Titel des formal table}
\label{Referenz, entspricht einem Anker in HTML}
...
(Hier kann z.B. ein tabular matter aber auch
beliebiger anderer Inhalt eingefügt werden.)
...
\end{table}
```

Die Position (Optional der `table`-Umgebung) kann durch einen von vier Buchstaben festgelegt werden:

1. `t` (top), also am oberen Rand der aktuellen Seite
2. `b` (bottom), also am unteren Rand der aktuellen Seite
3. `h` (here), also an genau der Stelle, wo die Umgebung im Dokument eingefügt wurde.
4. `p` (page): Bei dieser Option wird die Tabelle auf einer eigenen Seite angezeigt.

tabular matter

Dieser Bereich wird ähnlich wie der mathematische Modus nur kurz angeschnitten. Wenn Sie mehr über Tabellen in LaTeX wissen wollen, dann recherchieren Sie dazu bitte im Netz.

Wichtig:

Eine `tabular`-Umgebung ist für Texte gedacht. Wenn Sie Formeln in einer Tabelle gruppieren wollen, nutzen Sie bitte die `array`-Umgebung (siehe nächster Abschnitt).

```
\begin{tabular}{ausrichtung1, ausrichtung2, ...}
erste Zeile, erste Spalte & erste Zeile,
zweite Spalte & ... \\
zweite Zeile, erste Spalte & zweite Zeile,
```

```
zweite Spalte & ... \\  
...  
\end{tabular}
```

Eine solche Tabelle wird also durch eine `tabular`-Umgebung definiert. Nach dem ersten Paar geschweiften Klammern folgt ein zweites Paar, in dem für jede Spalte die Ausrichtung definiert wird:

- `l` linksbündig
- `c` zentriert
- `r` rechtsbündig
- `p{breite}` definiert eine maximale Breite einer Spalte

Wichtig:

Es gibt im Gegensatz zu HTML keine automatische Anpassung der Breite von Spalten.

array

Neben dem `tabular` matter, der für Texte gedacht ist, gibt es noch die `array`-Umgebung, die für mathematische Formeln gedacht ist. Diese müssen Sie allerdings zusätzlich per `\usepackage{array}` in der Präambel importieren.

Um Missverständnisse zu vermeiden: In einer `array`-Umgebung brauchen Sie nicht mehr den mathematischen Modus zu aktivieren, denn er ist dort automatisch aktiviert.

7.9 figures

Jetzt kommen wir zu dem, was der `<figcaption>` in HTML entspricht: Eine Umgebung für Bilder in LaTeX. Sie nutzen die `figures`-Umgebung also genauso, wie Sie die `table`-Umgebung für Tabellen und Texte nutzen konnten.

Es stellen sich also zwei Fragen:

- Warum gibt es die `figures`- und die `table`-Umgebungen?
- Wie können wir Bilddateien in LaTeX-Dokumenten einbinden?

Die Antwort auf die erste Antwort ist simpel: Es gibt getrennte Verzeichnisse für formal tables und figures. Und diese Verzeichnisse werden aus den jeweiligen Umgebungen automatisch generiert, wenn Sie `\listoffigures` bzw. `\listoftables` verwenden, um an einer Stelle im Dokument das entsprechende Verzeichnis erzeugen zu lassen.

7.9.1 Bilddateien in LaTeX

Bevor wir uns mit der Einbindung von Bilddateien beschäftigen können, müssen wir uns mit dem Thema pdfLaTeX und LaTeX beschäftigen. Ersteres ist eine Erweiterung, mit der wir pdf-Dokumente aus LaTeX-Dokumenten erzeugen können. Dennoch gibt es bei beiden einen entscheidenden Unterschied:

- Wenn wir pdfLaTeX verwenden, können wir PNG-, JPG- und PDF-Dateien als Bilddateien einbinden.
- Wenn wir dagegen „nur“ LaTeX verwenden, können wir ausschließlich EPS-Dateien verwenden.

Um überhaupt Bilddateien einbinden zu können, müssen wir die Präambel erweitern: `\usepackage{graphics}`

Um eine Bilddatei in unserem Dokument anzeigen zu lassen verwenden wir `\includegraphics{Dateiname ohne Endung}`. Das bedeutet, dass der Compiler automatisch nach einer Datei sucht. Wenn wir also sicherstellen wollen, dass wir ein Dokument sowohl mit pdfLaTeX als auch mit LaTeX konvertieren können, dann müssen wir die Bilddatei mit gleichem Namen einmal als PNG-, JPG- und PDF-Datei und einmal als EPS-Datei im gleichen Verzeichnis speichern wie das .tex-Dokument.

Sie können noch die Breite und Höhe als optionale Argumente `width = ...cm` bzw. `height = ...cm` festlegen. Dabei können Sie auch andere Maße wie z.B. pt verwenden, so lange diese in LaTeX definiert sind.

Wichtig:

Auch bei gleicher Bezeichnung sind diese Maße nicht mit denen in Adobe-Produkten identisch; leider beharrt besagter Konzern darauf eigene Definitionen einzelner Maßstäbe zu verwenden.

Außerdem können Sie noch über Werte wie `.75/columnwidth` die Größe proportional anpassen. Allerdings bedeutet das nicht, dass (z.B. bei einer JPG-Datei) das Bild gestochen scharf ist. Das ist nur bei einer Vektorgrafik sichergestellt.

7.10 Referenzen und Labels

An ein oder zwei Stellen haben Sie bereits `\label{Text}` gesehen. Das entspricht einem Anker in HTML. Also brauchen wir jetzt noch den „Link“ auf einen solchen Anker. In LaTeX heißen die aber nicht Link, sondern **Referenz**. Eine Referenz programmieren Sie mit `ref{Text}`.

U.a. wegen Labels empfehle ich bei der Erstellung von LaTeX-Dokumenten von Anfang an die Arbeit mit einem erweiterten Editor wie TeXStudio: Dieser zeigt Ihnen alle im Dokument verwendeten Labels an. Und das ist deshalb wichtig, weil es zu Inkonsistenzen kommen wird, wenn Sie zwei Labels gleich bezeichnen.

7.11 Boxen

Wenn Sie sich eine LaTeX-Referenz ansehen, werden Sie immer wieder über Container bzw. Elemente mit `box` im Namen stolpern. Eine Box bezeichnet so etwas wie einen Bereich, der abgeschlossen ist und Inhalte einer (z.B. gedruckten) Seite enthalten kann. Die größte Box entspricht dabei dem Bereich, der auf einer Seite insgesamt bedruckt werden kann. Generell wird aber das, was wir als einzelne Einheit betrachten als Box bezeichnet.

Es gibt beispielsweise die Möglichkeit durch `\fbox{text}` eine Textpassage im laufenden Text mit einem Rahmen zu umgeben.

Weitere Boxen, mit denen Rahmen erzeugt werden können sind `\shadowbox`, `\doublebox`, `\ovalbox` und `Ovalbox`.

Dann gibt es die `\parbox[pos]{width}{text}`, mit der ein Text am oberen `t` oder unteren `b` Rand einer Seite angezeigt werden kann, die die Breite `width` hat und `text` beinhaltet.

Aber auch das waren wieder nur einige ausgewählte Möglichkeiten, um Teile Ihres Dokuments hervorzuheben.

7.12 Abschluss

Damit haben Sie jetzt neben HTML eine weitere Markup Language kennen gelernt. Doch während Sie HTML kaum im Studium nutzen werden, sollten Sie LaTeX so schnell wie möglich verinnerlichen. Es ist für wissenschaftlichen Arbeiten ein international anerkannter Standard.

Kapitel 8

Gestaltung mit CSS

Leider beginnt dieses Kapitel mit einer Enttäuschung für all diejenigen, die dachten, jetzt käme der spannende Teil. Denn da dieses Skript für Studierenden mit den Schwerpunkten Informatik bzw. Elektrotechnik im Medienumfeld entworfen wurde, erhalten Sie hier lediglich einen grundlegenden Einblick darin, wie Sie die Elemente in HTML5 gruppieren können. Der Grund ist einfach und wurde auch schon mehrfach aufgeführt: Gestaltung ist die Aufgabe von Designern. Wir gehen deshalb auch nicht auf die Besonderheiten von CSS (bzw. der aktuellen Version CSS3) ein.

Die in diesem Kapitel besprochenen Aspekte werden im Sinne des MVC als **View** bezeichnet.

Wichtig: Bevor Sie mit diesem Kapitel beginnen sollten Sie eine Webpage programmiert haben, in der Sie eine Vielzahl der HTML-Container einprogrammiert haben, die Sie im vorletzten Kapitel kennen gelernt haben. Denn die Aufgaben in diesem Kapitel setzen eine solche Seite voraus.

8.1 Einbindung von CSS in HTML

Sie haben zwei Möglichkeiten, um CSS-Code in einem HTML-Dokument zu verwenden. Zum einen können Sie CSS-Code in Anführungszeichen setzen und ihn dann als Wert dem style-Attribut eines beliebigen Containers übergeben. Dieses Verfahren hat allerdings einen massiven Nachteil: Wenn Sie bei einer umfangreichen Webanwendung so vorgehen, wird es selbst bei kleinen Änderungen sehr mühsam, diese überall zu programmieren. Außerdem wird Ihr Layout so kaum einheitlich.

Deshalb sollten Sie besser die zweite Variante wählen. Hier programmieren Sie den CSS-Code in einer oder mehreren eigenen Datei/en und binden diese mit folgender Zeile in Ihr HTML-Dokument ein, die Sie in den

<head>-Container der HTML-Dokumente programmieren:

```
<link rel=stylesheet href=style.css>
```

Die Datei kann natürlich auch anders als `style.css` heißen, wichtig ist nur, dass die Endung `.css` ist. Außerdem gilt in Bezug auf die URL dasselbe, was Sie schon im vorletzten Kapitel über relative und absolute Adressierung gelernt haben.

8.2 Programmierung einer CSS-Datei

Im Gegensatz zu einem HTML-Dokument besteht eine CSS-Datei lediglich aus den Namen von Containern eines HTML-Dokuments und jeweils einem Rumpf, in dem die Formatierungsvorgaben einprogrammiert werden.

Wenn Sie CSS wie oben beschrieben innerhalb eines HTML-Dokuments über das `style`-Attribut programmieren, dann stehen in den Anführungszeichen genau die Formatierungsvorgaben, die im Rumpf bei der CSS-Datei stehen.

Nehmen wir an, Sie wollen eine Formatierungsvorschrift für alle `<h1>`-Container einzelner HTML-Dokumente festlegen, haben sich aber noch auf keine Formatierungsvorschrift festgelegt. Dann sieht Ihre CSS-Datei so aus:

```
h1 {  
  
}
```

Quellcode 3.1: CSS-Rumpf für Designdefinitionen für den `<h1>`-Container
In Dokumentationen über CSS werden Sie des Öfteren auf den Begriff des **Selektors** stoßen. Der Selektor ist in CSS schlicht der Bezeichner des Containers, den Sie formatieren wollen.

Damit wissen Sie jetzt ungefähr 50% dessen, was Sie im Rahmen dieses Kurses für die Programmierung von CSS wissen müssen. Alles was Sie noch lernen müssen ist wie Sie den sogenannten **Properties** Werte zuordnen müssen, um die Gestaltung der Container festzulegen und was es mit dem `.` (Punkt) und `#` vor Selektoren in CSS auf sich hat.

8.3 Programmierung von Formatierungen

Schauen wir uns an, was wir nun in den Rumpf programmieren. Sie kennen ja die Attribute aus HTML. Bei denen konnte ein Wert mit einem Gleichzeichen zugeordnet werden. Bei CSS gibt es dazu zwei Unterschiede: Hier

reden wir nicht von Attributen sondern von Properties und die Zuordnung eines Wertes wird durch einen Doppelpunkts und nicht durch ein Gleichzeichen durchgeführt. Die Fortgeschrittenen unter Ihnen haben den Begriff Property schon im Zusammenhang mit Objekten im Sinne der objektorientierten Programmierung gehört. CSS hat nichts mit objektorientierter Programmierung zu tun und deshalb reden wir hier nicht von Objekten, sondern von Elementen, deren Eigenschaften (also Properties) wir ändern.

Und welche Eigenschaft können wir fast überall ändern? Genau: Die Schriftfarbe. Und wie machen wir das? Wir nehmen die Eigenschaft Schriftfarbe, die naheliegender Weise `color` heißt und ordnen Ihr einen Wert zu. Fangen wir mit so etwas profanem wie rot an. Wenn Sie also wollen, dass alle Überschriften in einem schreienden Rot angezeigt werden, müssen Sie nur das folgende in Ihre CSS-Datei programmieren:

```
h1 {  
  color : red;  
}
```

Quellcode 3.2: Rote Überschrift

Wenn Sie die CSS-Datei wie oben beschrieben in Ihr HTML-Dokument eingebunden haben und Sie im gleichen Verzeichnis gespeichert haben wie das HTML-Dokument, dann müssen sie nur noch den Browser aktualisieren (z.B. per F5 beim Firefox) und schon werden alle Überschriften in rot angezeigt. Und wenn Sie sonst nichts in der CSS-Datei geändert haben, dann ändert sich auch sonst nichts.

8.3.1 Mehr über Farben

Anstelle von vielen englischen Wörtern für Farben, können Sie auch noch präzise Farbangaben zum Beispiel in der RGB-Notation programmieren.

An dieser Stelle werden wir Themen wie RGB u.a. nicht ausführliche behandeln. Hier werden Sie nur erfahren, wie Sie diese Farbangaben in einem CSS-Dokument einprogrammieren können:

- RGB-Farben: Diese Farbangaben bestehen aus drei Zahlen, jeweils von 0 bis 255 (oder von 0 bis FF als Hexadezimalzahl). Um sie in CSS zu programmieren, können sie auch als Hexadezimalwerte vorliegen.
 - Variante a: Die Zahlen liegen als Hexadezimalzahlen vor. Nehmen wir an, es sind die Zahlen 1F, 9, 27. Dann lautet die Formatierungsvorschrift in CSS:
`color : #1f0927;` Sie schreiben also zunächst ein #-Symbol, um

zu zeigen, dass ein RGB-Wert in Hexadezimalzahlen folgt. Danach folgen die drei Zahlen ohne Leerstelle, wobei Sie für jede einstellige Zahl (wie hier die 9) noch eine führende Null ergänzen müssen.

Wenn Sie jetzt denken: Da stehen aber sechs Zahlen und nicht drei, dann verwechseln Sie Zahlen mit Ziffern.

- Variante b: Die Zahlen liegen als Dezimalzahlen vor. Nehmen wir an, es sind die Zahlen 128, 92, 7. Dann lautet die Anweisung:
`color: rgb(128, 92, 7);`

- Es gibt außerdem Fälle, in denen der Alpha-Wert (Opacity) der Farbe angegeben ist. Das ist ein Wert von 0 bis 1, der die Transparenz definiert. Nehmen wir an, Alpha wäre bei 0.3 und die Farbe ist die selbe wie bei Variante b. Dann lautet die Anweisung:

```
color: rgba(128, 92, 7, 0.3);
```

Das ist eine Kurzfassung der folgenden Anweisung:

```
color : rgb(128, 92, 7); opacity(0.3);
```

Ist es Ihnen aufgefallen? Bei der letzten Anweisung wurde zweimal ein Semikolon gesetzt, um jeweils eine Formatierung zu beenden, obwohl beide für die Property `color` gelten. Das ist sehr wichtig: Sie können einer Property mehrere Formatierungen zuordnen, aber jede einzelne muss durch ein Semikolon beendet werden.

Außerdem wird hier die englische Schreibweise für die Dezimaltrennung verwendet: Dort verwenden Sie kein Komma, sondern einen Punkt, um zwischen ganzen Zahlen und „Nachkommastellen“ zu trennen.

Für den Fall, dass Sie unsicher sind, folgt hier ein aktualisierter Quellcode:

```
h1 {  
  color : rgb(128, 92, 7); opacity(0.3);  
}
```

Quellcode 3.3: Eine Überschrift mit einem abgedunkelten Farbton.

Alternativ zu RGB können Sie auch HSL-Farben programmieren. Aber das lassen wir an dieser Stelle außen vor, weil Designer im Regelfall Farbwerte in mehreren Standards angeben.

8.3.2 Zwei weitere Properties in Bezug auf Farben

Hier folgen noch ein paar Properties, mit denen Sie verschiedene Bereiche Ihrer Webpage farblich anpassen können:

- `background`:

Damit färben Sie den Hintergrund eines Elements. Wenn Sie hier anstelle einer Farbe ein Bild einstellen wollen, z.B. `image.jpg`, dann können Sie es mittels

```
background : url(image.jpg);
```

als Hintergrund des Elements festlegen.

- `mark`

ist keine Property, sondern ein HTML-Container, mit dem Sie Textpassagen hervorheben können. Schlagen Sie im Zweifelsfall nochmal im HTML-Kapitel nach.

Aufgaben:

- Öffnen Sie die Webpage <http://www.colorpicker.com/> und wählen Sie dort eine Farbe aus, die Sie für Ihre Überschriften verwenden wollen. Am oberen Rand finden Sie den hexadezimalen Wert, den Sie einfach kopieren können, um ihn im CSS-Code zu verwenden.
- Vergeben Sie jetzt für die verschiedenen Überschriften Ihrer Webpage einen jeweils etwas helleren Farbton.
- Wählen Sie nun einen anderen Farbton aus, in dem Sie Absätze anzeigen lassen wollen.
- Stellen Sie zusätzlich einen Farbton für den Hintergrund jeder Überschrift und jedes Absatzes (das sind die `<p>`-Container) ein.

Vermutlich sieht Ihre Webpage jetzt alles andere als augenfreundlich aus, aber das macht nichts. Einzig wenn Sie merken, dass Texte gar nicht mehr zu lesen sind, sollten Sie die Einstellungen anpassen.

8.4 Ein wenig Form

Wie Sie sehen sind die Hintergründe der Absätze Ihrer Texte und der Überschriften im Moment rechteckig. Das sieht nicht wirklich schön aus, da wären abgerundete Ecken schöner. Also kümmern wir uns darum.

Das Prinzip fürs Abrunden von Ecken funktioniert wie folgt: Sie legen einen Radius fest, der in Bildpunkten (Pixel, kurz `px`) gemessen wird und ordnen diesen Wert der Property `border-radius` zu. Meist ist ein Wert zwischen 5 und 25 gut, aber Sie haben hier die freie Auswahl:

```
border-radius : 20px;
```

Manchmal finden Sie auch Quellcode, in dem `border-radius` mit bis zu vier Werten verwendet wird. Schlagen Sie ggf. bei w3schools nach, wie diese Angaben zu verstehen sind.

Unter Umständen überschneidet Ihr Text jetzt die abgerundeten Ecken. Da können Sie sich mit der Property `padding` behelfen. `padding` legt einen Abstand vom Rand fest, den der Text haben muss.

```
padding : 5 px;
```

Sie können auch die Größe festlegen, die die farbige Fläche im Hintergrund haben soll. Dazu verwenden Sie die beiden Properties `width` und `height`, die trivialerweise festlegen, welche Breite und Höhe der Hintergrund haben soll. Allerdings könnten Sie hier eine Festlegung treffen, die bei kleinen Displays zu Problemen in der Darstellung oder der Bedienung führt.

8.5 Definition von CSS für einzelne Container eines Typs

Stellen Sie sich vor, Sie wollen jetzt noch die Formatierung von einigen `<p>`-Containern anpassen, die aber z.B. nur dann angewendet werden soll, wenn in dem Container Quellcode angezeigt werden soll. Da es in HTML (auch in Version 5) keine solchen Spezialcontainer gibt, hätten Sie mit Ihrem bisherigen Wissen jetzt ein Problem.

Für die Lösung dieses Problems gibt es zwei Möglichkeiten:

- Zum einen können Sie über das `class`-Attribut einzelne Container in HTML so hervorheben, dass Sie diesen durch ein entsprechendes CSS-Skript formatieren lassen.

Bsp.: In HTML haben Sie mehrere Absätze mit dem Attribut `class=PHP` programmiert:

```
<p class=PHP>
```

Jetzt können Sie in CSS einen entsprechenden Teil programmieren:

```
.PHP {
color : blue;
font-family : monospace;
}
```

Damit werden zusätzlich zu allen CSS-Anweisungen, die für `<p>`-Container gelten bei den `<p class=PHP>`-Containern die Schrift in blau und einem etwas veränderten Textformat angezeigt.

- Zum anderen können Sie aber auch anhand der Bezeichner von `id`-Containern CSS-Skripte definieren. Anstelle des Punktes (wie bei `class`) wird hier ein `#` verwendet.

Bsp.: In HTML haben Sie einen Container mit dem Attribut `id=nutzer` programmiert. In CSS können Sie diesen wie folgt formatieren:

```
#nutzer { ... }
```

8.6 Mehr zu CSS

Dieses Kapitel ist sehr kurz ausgefallen, weil Sie sich im Rahmen dieser Veranstaltung auf gute Programmiertechniken konzentrieren sollen und CSS eine Sprache für Mediendesigner ist, um das Design in den Dokumenten einer Webanwendung festzulegen: Sie entscheiden sich dafür, wie Sie einen Container gestalten wollen und programmieren das schlicht Zeile für Zeile in einer CSS-Datei. Das ist Programmieren für Sechstklässler und hat nichts aber auch nicht das Geringste mit Medieninformatik zu tun.

Deshalb hier Ihre Hausaufgaben:

- Sehen Sie sich auf http://www.w3schools.com/css/css3_intro.asp um, was es noch für Gestaltungsmöglichkeiten in CSS3 gibt und setzen Sie auf Ihrer Webpage alles um, was Sie gestalterisch spannend finden.
- Programmieren Sie einige HTML-Container über die entsprechenden Attribute, sodass diese für Administratoren bzw. eingeloggte Nutzer eine andere Darstellung erhalten können. Momentan können die entsprechenden CSS-Container noch leer bleiben.

Kapitel 9

Dynamische Web(anwendungs)entwicklung mit PHP

Ab diesem Kapitel setzen wir uns mit der Programmierung des Controllers im Sinne des MVC-Patterns auseinander, also des Teils eines Projekts, der steuert, welche Inhalte wann angezeigt werden.

An dieser Stelle muss der Unterschied zwischen serverseitig und clientseitig besprochen werden. Wie Sie wissen, beschäftigen wir uns in diesem Kurs mit den Grundlagen der Entwicklung von Webseiten bzw. Webanwendungen. Bei allen Anwendungen, die ein Netzwerk nutzen wird von Server und Client gesprochen: Der Server bietet einen Dienst an, der Client fordert die Ausführung dieses Dienstes an. Somit kann jedes Programm sowohl Client als auch Server sein.

Bislang haben wir konzeptionell im Grunde nur über das MVC-Pattern gesprochen, also die Unterteilung der Programmierung eines Projektes in die Teile Modell, Ansicht und Controller. Das ist ein Ansatz, der aus der Erfahrung bei objektorientierten Projekten entstanden ist. Bei netzbasierten Projekten wird unabhängig davon von der Unterteilung in die serverseitige und die clientseitige Programmierung gesprochen.

Im einfachsten Fall geht es also um die Unterscheidung der Programmteile, die die angebotenen Dienste darstellen von den Programmteilen, die diese Dienste anfordern. Hier kann aber auch danach unterteilt werden, welche Daten vom Server und welche vom Client erzeugt werden. Leider gibt es hier keine einfache entweder/oder-Entscheidung. Im Grunde können die meisten Komponenten sowohl auf dem Server als auch auf dem Client betrieben werden. Wie Sie hier erkennen können, werden die Begriffe Server

und Client also sowohl für Programme als auch für vernetzte Computer benutzt.

Nehmen wir dafür CSS als Beispiel: Wir haben hier mehrere HTML-Dokumenten und vielleicht auch mehrere CSS-Dateien. Ein Server könnte nun an allen nötigen Stellen den CSS-Code mittels des `style`-Attributs in die HTML-Dokumente integrieren und dann dieses Ergebnis an einen Client übertragen. Er könnte aber auch alle Dateien so wie sie sind an den Client übertragen. Für den Nutzer, der am Client sitzt macht das keinen Unterschied, denn er bekommt in beiden Fällen das gleiche angezeigt. Für den Entwickler der Webpage kann es dagegen einen Unterschied machen. Und für den Serveradministrator macht es in jedem Fall einen Unterschied. Keine Sorge, denn wie das im Detail abläuft ist für uns gänzlich irrelevant, damit dürfen sich Informatik-Studierende im Master-Studium beschäftigen.

Es soll Ihnen lediglich deutlich machen, dass die Ausführung des programmierten Codes nicht immer auf dieselbe Art erfolgen muss. Das ist immer nur dann der Fall, wenn wir mit prozeduralen Programmiersprachen arbeiten. Wichtig für Sie ist, dass den Begriff der serverseitigen Programmierung kennen lernen, dann bei der Programmierung in PHP tun wir genau das: Wir programmieren ein Programm, dass auf dem Server laufen wird und das steuern wird, wie die Inhalte unserer HTML-Seiten geändert werden: Wenn ein Nutzer z.B. Eingaben in Formularen einträgt und dann über eine Schaltfläche, die Sie mit `<input type=submit>` programmiert haben, die Übertragung an das Programm startet, dann wird PHP (oder eine andere Programmiersprache) interessant.

Im Rahmen dieses Kurses brauchen Sie sich allerdings über die Installation und Wartung eines Servers keine Gedanken zu machen. Alles, womit wir uns hier beschäftigen können Sie auf Ihrem Rechner betreiben. Ggf. nötige Programme, die einen Server auf Ihrem Rechner betreiben bekommen Sie kostenlos im Netz.

Umgekehrt möchte ich Sie allerdings davor warnen, den Aufwand beim Betrieb eines Servers zu unterschätzen. Denn hier geht es nicht allein um das Anmieten bei einem Betreiber und um eine grundlegende Konfiguration. Vielmehr müssen Sie bei einem solchen Server sicherstellen, dass dieser so gut wie möglich gegen Angriffe geschützt ist. Denn wenn Sie hier nicht ausreichende Vorkehrungen treffen, können Sie sicher sein, dass Ihr Server von Angreifern missbraucht wird, um Angriffe gegen andere Nutzer zu starten. Im schlimmsten Fall kann das für Sie bedeuten, wegen Beihilfe zu einer Straftat verurteilt zu werden.

Wichtig: Dieses Kapitel ist eine einfache Einführung in die imperative Pro-

grammierung mit PHP. Zum Teil werden Sie hier Anmerkungen finden, die klar aufzeigen, dass vieles hier vereinfacht dargestellt ist. Insbesondere werden hier viele Dinge, die in PHP nicht möglich sind außen vor gelassen, obwohl es sich dabei um Möglichkeiten aus dem Bereich der imperativen Programmierung handelt. Die Struktur dieses Kapitels ist aber ansonsten gleich derjenigen, die im zweiten Teil dieses Buches eingeführt wird. Wenn Sie also die Grundlagen der imperativen Programmierung erlernen wollen, dann blättern Sie gleich weiter in den zweiten Teil dieses Buches. Und immer, wenn Sie dort einen Abschnitt beendet haben, schlagen Sie in diesem Kapitel nach, wie der entsprechende Abschnitt in PHP umgesetzt wird. So haben Sie die Grundlagen der imperativen Programmierung und erlernen, wie diese in PHP umgesetzt werden, bzw. welche dieser Grundlagen in PHP nicht enthalten sind.

9.1 PHP 5

Bis auf die Zahl der Version haben PHP und HTML nichts gemein. Während HTML5 nach 14 Jahren eine vollständig überarbeitete Version der Markup Language ist, wurde PHP in der Version 5.0 vor 11 Jahren veröffentlicht und seitdem immer wieder erweitert. Die letzte Version trug die Nummer 5.6. Die aktuellste Version ist nicht PHP 6 sondern 7 und im November 2015 erschienen.

Wie an der Überschrift zu erkennen ist, wird in diesem Kapitel aber noch die Version 5, besser gesagt die Version 5.6 behandelt.

Doch auch wenn es eine neue Version geben wird, werden wir bei diesem Kurs für die Medientechnik bis auf weiteres bei PHP 5.6 bleiben. Wobei alles, was Sie hier kennen lernen auch in den PHP-Versionen 5.0 – 5.5 nutzbar sein sollte.

Doch was ist PHP eigentlich? Schauen wir uns dazu den Namen an: PHP steht für Hypertext Preprocessor. Was Hypertext ist wissen Sie, aber über Präprozessoren haben wir noch nicht gesprochen. Das sind Programme, deren Zweck darin besteht, Teile eines Programms vorzubereiten, die in einer (anderen) Programmiersprache entwickelt werden. Wenn Sie das zusammennehmen ist PHP schlicht eine Sprache, die HTML-Code erzeugen kann. Und genau das wollen wir: Eine Sprache, mit der wir (neue) Inhalte für unsere Webpage erzeugen können.

Vielleicht fragen Sie jetzt, was denn mit CSS-Dateien ist. Und sie haben es richtig erkannt: Mit PHP können wir zwar CSS-Code über das style-Attribut in einen Container integrieren, aber an unseren CSS-Dateien können

wir nichts direkt ändern. Andererseits ist das kein Problem, wenn Sie sauber zwischen Controller und View trennen: In PHP programmieren wir, welche Inhalte wann dargestellt werden. In CSS (und nur dort) programmieren wir, wie diese Inhalte dargestellt werden.

Sehen wir uns nun an, was Thomas Theis in seinem Band „Einstieg in PHP 5.6 und MySQL 5.6“ über die Vorteile von PHP schreibt:

- Demnach ist PHP für Einsteiger leicht erlernbar, weil es nur all das beinhalten würde, was für die serverseitige Programmierung einer Webanwendung nötig ist.

Die Vorstellung, dass es Einsteigern leichter fällt, eine Sprache zu erlernen, die weniger Möglichkeiten bietet, als das bei anderen Sprachen der Fall ist, ist absurd. Es mag sein, dass die vollständige Dokumentation einer solchen Sprache übersichtlicher ist, aber letztlich hängt die Leichtigkeit des Einstiegs davon ab, wie gut die Dokumentation zur Sprache auf Einsteiger ausgerichtet ist. Und in der professionellen Softwareentwicklung spielt ein leichter Einstieg in eine Sprache eher eine untergeordnete Rolle. Hier ist es vor allem wichtig, welche Möglichkeiten genutzt werden können.

- PHP sei nach Theis eine gute Wahl, weil es von einer Vielzahl von Servern unterstützt wird.

Doch auch das ist eher ein schwaches Argument; letztlich ist der Preis relevant, den wir für den Betrieb aufwenden müssen. Hier ist beispielsweise JavaScript mit node.js aus Gründen im Nachteil, die daraus resultieren, dass es mehr Zugriffsrechte benötigt, was die Betreiber von Servern sich eben entsprechend bezahlen lassen.

- Weiter sei die Tatsache wichtig, dass PHP kostenlos ist.

Das ist zwar eine sympathische Tatsache, aber kostenlos nutzbare Sprachen und sogar Frameworks sind heute keine Ausnahme mehr. Egal ob Java EE, Ruby on Rails, HTML5 mit CSS3 und JavaScript oder woran Sie noch denken: Alle können kostenlos genutzt werden und bieten einen vergleichbaren, wenn nicht sogar größeren Umfang.

- Dann sei es ein Vorteil, dass PHP ausschließlich serverseitig eingesetzt werden kann.

Worin der Nachteil besteht, eine Sprache sowohl server- als auch clientseitig ausführen zu können, erschließt sich dem Autor dieses Buches nicht. So bedeutet die Tatsache, dass eine Sprache auch clientseitig ausgeführt werden kann, eben nicht, dass sie vollständig dort ausgeführt wird. Und damit kann ein Nutzer eben auch nicht wie von Theis behauptet den gesamten Code einer Webanwendung einsehen.

Im Gegenteil ist diese Beschränkung von PHP von Nachteil: Wenn bei der Entwicklung festgestellt wird, dass ein Programmteil effizienter vom Client ausgeführt werden kann, ohne dabei sicherheitsrelevante Daten preiszugeben, dann sollte ein Entwickler das tun. Genau das verhindert aber PHP mit seiner Fixierung auf die Serverseitigkeit.

Aus meiner Sicht gibt es nur einen Grund, sich mit PHP zu beschäftigen: Es ist schlicht eine der häufigsten Sprachen, die bei der Entwicklung dynamischer Webanwendungen zum Einsatz kommen. Wenn Sie also in die Webentwicklung gehen wollen, haben Sie mit Kenntnissen in diesem Bereich mehr Auswahl unter potentiellen Arbeitgebern.

Andererseits ist das der aktuelle Stand. Und der ändert sich in kaum einem Bereich so schnell wie in der Informatik. JavaScript hat aus Sicht dieses Autors zurzeit wesentlich mehr Potential, um zur meistverwendeten Sprache für die Entwicklung von Webanwendungen zu werden. PHP ist mit seinen Beschränkungen dagegen einfach nicht universell genug einsetzbar. Das könnte sich mit der Version 7 natürlich geändert haben.

9.1.1 Vorbereitung des Computers, um in PHP programmieren zu können.

Während Sie HTML- und CSS-Dateien einfach mit Hilfe eines Browsers ausführen konnten, ist das bei PHP leider nicht der Fall. Wie gerade beschrieben müssen Sie einen Server betreiben, der eine Laufzeitumgebung anbietet, die PHP ausführen kann.

Hier gibt es zwei Standardlösungen, die Sie kostenlos im Netz herunterladen können und die Ihnen die gesamte Konfiguration abnehmen: XAMPP und EasyPHP. Letzteres ist aus meiner Sicht deutlich komfortabler, aber im Grunde ist nur wichtig, dass Sie eines der beiden installieren. Nach der Installation müssen Sie dann Ihre Webanwendungsdateien (also die HTML-, CSS- und PHP-Dateien) in dem Verzeichnis speichern, auf das der Server zugreift. Welches das ist, müssen Sie anhand der Dokumentation des Pakets prüfen, das Sie nutzen.

9.2 PHP-Programme in HTML-Dokumenten integrieren

Genau wie bei CSS und JavaScript können wir innerhalb eines HTML-Dokuments PHP-Code ausführen lassen. Allerdings gibt es hier einen wichtigen Unterschied:

- Der Dateiname eines HTML-Dokuments, das das tut, sollte in `.php` umbenannt werden. Richtig! Die Datei sollte dann nicht mehr auf `.html` enden. Der Unterschied ist in sofern wichtig, als Sie bei der Bearbeitung eines Projekts so schneller die relevanten Dateien finden können.

Um PHP-Code direkt in HTML-Code zu programmieren, muss ein besonderer Container verwendet werden. Er zeichnet sich dadurch aus, dass der gesamte PHP-Code im öffnenden Tag steht und es kein schließendes Tag gibt. (Zur Erinnerung: Das war bei JavaScript anders. Dort wurde ein `<script>`-Container verwendet, dessen Inhalt der JavaScript-Code ist.)

Der PHP-Container sieht so aus:

```
<?php  
  
?>
```

Quellcode 4.1: Ein HTML-Container, in dem PHP-Code programmiert werden kann.

Wenn wir JavaScript- oder CSS-Dateien haben, dann können wir diese auch über den `<head>`-Container einbinden. Das ist bei PHP leider nicht der Fall. Dort gibt es zwei andere Möglichkeiten:

- Zum einen können wir PHP-Code, der im laufenden HTML-Code eingefügt werden soll in eine Datei speichern. Diese können wir dann innerhalb des `<?php ... ?>`-Containers durch den Befehl `include()` aufrufen:

```
<?php  
include (phpdatei.php) ;  
?>
```

Quellcode 4.2.: Einfügen von PHP-Code in ein HTML-Dokument

- Zum anderen müssen wir bei Formularen (wenn wir sie per PHP auswerten wollen) über das `action`-Attribut des `<form>`-Containers explizit den Dateinamen zuordnen. Das bewirkt, dass beim Anwählen des submit-Buttons durch einen Nutzer die Eingaben an das genannte PHP-Programm übergeben werden.

In diesem Fall müssen wir noch die Übertragungsmethode festlegen. Dazu programmieren wir das `method`-Attribut des `<form>`-Containers. Es gibt zwar zwei Möglichkeiten, welchen Wert Sie `method` zuordnen, Sie sollten hier aber immer den Wert `POST` wählen. Die Alternative zu `POST` lautet `GET`.

Aufgabe:

- Erstellen Sie jetzt für jedes Formular, das Sie in Ihrer Webpage programmiert haben eine Datei mit einem Namen, der zu dem Formular passt. Die Endung muss zwar nicht auf `.php` lauten, allerdings lässt sich so bei größeren Projekten leichter der Überblick bewahren.
- Ergänzen Sie Ihre Formulare so, dass bei jedem Formular die Nutzereingaben an das passende PHP-Programm übergeben werden.

Anm.: Wenn Sie danach die Webpage neu starten (vorausgesetzt, Sie haben die Dateien zuvor in das Verzeichnis verschoben, in dem XAMPP bzw. EasyPHP darauf zugreifen können und jeweils den Server gestartet), und bei einem Formular den Submit-Button anwählen, werden Sie wahrscheinlich eine Fehlermeldung erhalten, denn bislang haben wir ja noch kein PHP-Programm erstellt. Aber das ist vollkommen in Ordnung.

Wichtig: Dieser Abschnitts könnte von Ihnen in einer Hinsicht missverstanden werden: Streng genommen wird eben kein PHP-Programm in ein HTML-Dokument integriert, sondern es passiert folgendes: Wann immer der Server in einem HTML-Dokument auf ein PHP-Programm hingewiesen wird startet er einen Prozess, in dem dieses Programm laufen wird. Wann immer nun das PHP-Programm etwas tun soll, dann wird es auf dem Server ausgeführt. Wenn dann HTML-Code durch das PHP-Programm ausgegeben wird, wird dieser Code an der entsprechenden Stelle des HTML-Dokuments eingefügt. Danach wird das vollständige HTML-Dokument vom Server zum Client übertragen. Der Nutzer wird also nie erkennen können, dass bzw. wo die Webpage mit PHP erzeugt wurde. Im nächsten Abschnitt gibts dazu gleich ein Beispiel.

9.3 PHP-Code innerhalb von HTML – Erstes Beispiel

Rufen Sie eine Ihrer Seiten mit einem Webbrowser auf. Es sollte eine Seite mit mehreren Absätzen sein. Öffnen Sie nun das HTML-Dokument mit einem Editor. Gehen Sie hier zu einer beliebigen Zeile, in der sich lediglich Text befindet. Ändern Sie das Dokument dann so ab, wie im folgenden Beispiel und speichern Sie die Datei anschließend wieder. Vergessen Sie dabei bitte nicht die Änderung des Dateinamens.

Achten Sie außerdem bei den entsprechenden Seiten darauf, Links in den statischen Webpages zu ändern, die auf Dateien mit der Endung `.htm` bzw. `.html` verweisen.

```
<p>
.... Ihr Text ....
```

</p>

Ändern Sie diesen Teil so ab:

```
<p>
<?php echo ".... Ihr Text ...."; ?>
</p>
```

Quellcode 4.3: Textausgabe mit PHP

Aktualisieren Sie jetzt die Seite im Browser, achten Sie dabei aber darauf, dass in der Adresszeile die Endung der Datei nicht mehr `.htm` bzw. `.html` heißt, sondern `.php`. Sehen Sie den Unterschied? Nein? Natürlich nicht, denn der PHP-Teil tut ja nichts anderes, als genau den HTML-Code ins HTML-Dokument einzufügen, der da vorher schon stand. Das mag relativ sinnfrei wirken, aber Sie werden in den nächsten Abschnitten lernen, dass es Ihnen mit PHP möglich ist, eine von mehreren HTML-Code-Passagen auszuwählen und diese in ein HTML-Dokument einfügen zu lassen. So können Sie später entscheiden, ob ein bestimmter Nutzer Zugriff auf Inhalte bekommt, ob eine bestimmte Nutzerin bestimmte Auswahloptionen anwählen darf usw.

Wenn Sie Sonderzeichen per `echo()` ausgeben wollen, also Zeichen, die eigentlich eine Funktion übernehmen, dann schreiben Sie schlicht ein `\` davor. (Nicht zu verwechseln mit `/`.) Wenn Sie also Anführungszeichen ausgeben lassen wollen, die ja sonst in `echo()` bewirken, dass die Ausgabe beendet wird, dann schreiben Sie anstelle von `"` einfach `\` und schon wird `"` im HTML-Dokument eingeblendet.

Aufgaben: Probieren Sie folgendes selbst aus:

- Wie siehts mit Umlauten aus, die Sie mit `echo()` durch PHP erzeugen lassen? Internationalisierung und Lokalisierung betreffen ja nur das HTML-Dokument und nicht das PHP-Programm.
- Und was ist mit HTML-Tags? Können Sie vollständige HTML-Container per `echo()` erzeugen und in ein HTML-Dokument einfügen?

9.3.1 Kommentare

Bislang haben Sie ausschließlich Zeilen programmiert (egal ob in HTML, CSS oder PHP), die dann vom Computer in irgend einer Form ausgeführt wurden und die für die Anzeige oder den Aufbau der Webanwendung wichtig waren. Kommentare sind etwas anderes: Sie dienen dazu, dass andere Entwickler erkennen können, welche Aufgabe ein Programmteil

übernimmt.

Deshalb werden Kommentare auch häufig entweder schlecht oder gar nicht verwendet: Während Sie etwas programmieren wissen Sie im Regelfall genau, was Sie tun, also brauchen Sie in diesem Moment keine Kommentare. Aber Sie können sich darauf verlassen, dass Sie früher oder später einzelne Passagen Ihres Projektes nochmal ändern müssen und dann werden Sie die Kommentare brauchen.

Vielleicht fragen Sie sich, was das denn soll, da man doch bloß den Code lesen muss, um zu erkennen, was er tut. Wenn Ihnen diese Frage durch den Kopf geht, dann rufen Sie sich bitte in Erinnerung, was eigentlich der Sinn und Zweck des semantic web ist, bzw. was der Unterschied zwischen Syntax und Semantik ist und warum imperative Programme sehr leicht so zu programmieren sind, dass sie etwas anderes tun als Sie wollen.

Zur Erinnerung: Eine Programmzeile in einer imperativen Sprache weist den Rechner an, bestimmte Anweisungen auszuführen. Sie sagt dagegen nichts darüber aus, wofür diese Anweisungen dienen, bzw. in welchem Kontext jede einzelne Zeile steht. Wenn Sie also keine Kommentare verfassen oder in Ihren Kommentaren nur steht, was der Quellcode tut, dann werden auch Sie bereits wenige Tage später nicht mehr wissen, welche Aufgabe damit erfüllt wird. Und wenn Sie dann eine Änderung durchführen, ist die Wahrscheinlichkeit sehr groß, dass Sie zwar eine neue Funktionalität realisieren, aber gleichzeitig eine alte Funktionalität entfernen.

Hier ein einfacher Algorithmus, um das Problem zu veranschaulichen:

- Erzeuge eine Variable mit dem Namen a.
- Weise a die Adresse zu, an der der Anfang einer Liste mit elf Einträgen steht.
- Erzeuge eine Variable mit dem Namen b.
- Erzeuge eine Variable mit dem Namen c.
- Erzeuge eine Variable mit dem Namen d.
- Erzeuge eine Variable mit dem Namen e.
- Erzeuge eine Variable mit dem Namen m.
- Erzeuge eine Variable mit dem Namen n.
- Erzeuge eine Variable mit dem Namen q.

- Tu das folgende zehnmal, wobei du den Durchlauf in der Variable *m* speicherst.
 - *n* bekommt jetzt den Wert $10-m$.
 - Tu das folgende *n*-mal, wobei du die Nummer des Durchlaufs in der Variable *q* speicherst:
 - Lies den Wert aus der Speicherstelle mit der Nummer von $a + q$ und speichere ihn in *c*.
 - Lies den Wert aus der Speicherstelle mit der Nummer von $a + q + 1$ und speichere ihn in *d*.
 - Wenn *c* größer als *d* ist, tue folgendes:
 - * Speichere *c* in *e*.
 - * Speichere *d* in *c*.
 - * Speichere *e* in *d*.
 - * Speichere den Wert von *c* in der Speicherstelle mit der Nummer $a + q$.
 - * Speichere den Wert von *d* in der Speicherstelle mit der Nummer $a + q + 1$.
- sonst mache einfach weiter.

Beispiel für schlechte Kommentare.

Und? Was macht dieses Programm? Wenn Sie jetzt sagen, dass Sie das nicht erkennen können, dann ist das kein Wunder, denn Sie können zwar im Detail nachlesen, was der Computer in jeder Zeile tun soll, aber nirgends steht, welchen Zweck das ganze erfüllen soll. Dieser Zweck dagegen ist recht simpel:

Dieses Programm sortiert die Zahlen einer Liste mit elf Einträgen nach der Größe der Zahlen.

Beispiel für einen guten Kommentar.

Das Beispiel oben zeigt, wie schlechte Kommentare aussehen: Es sind Unmengen an Zeilen, die nur in anderen Worten wiedergeben, was im Quellcode steht. Ein guter Kommentar fasst dagegen zusammen, warum das passiert und ggf. warum das so passiert. Letzteres ist dann wichtig, wenn Ihr Programmteil nicht nur einen einzelnen Zweck erfüllt, sondern zusätzlich noch bestimmte Sonderfälle abfangen muss.

Jetzt wissen Sie auch, warum wir bei HTML nicht über Kommentare gesprochen haben: Wenn sie guten HTML5-Code entwickeln, ist für jeden

HTML-Entwickler sofort erkennbar, warum Sie die einzelnen Container programmiert haben. Gerade mit den Möglichkeiten des semantic web bleiben hier kaum Fragen offen.

Bei den meisten Programmiersprachen ist das anders: Dort müssen Sie Kommentare verwenden, um zu erklären, welchem Zweck einzelne Programmteile dienen. In der Praxis fehlen Kommentare aber häufig. Gerade wenn Sie ein Projekt in einem Unternehmen übernehmen sollen, sind Sie deshalb in einer üblichen aber dennoch unschönen Situation.

Es gibt zwei Möglichkeiten, um Kommentare in PHP zu programmieren: Entweder als Ergänzung zu einer Programmzeile oder als einen mehrzeiligen Absatz.

Alles, was in einer Programmzeile nach `//` steht, ist ein Kommentar. Die nächste Zeile ist dann wieder normaler Programmcode.

Einen mehrzeiligen Kommentar beginnen Sie mit `/*` und beenden ihn nach beliebig vielen Zeilen und Zeichen mit `*/`.

Wichtig: Alles, was als Kommentar gilt wird nicht ausgeführt. Wenn Sie also einen mehrzeiligen Kommentar erstellen, achten Sie bitte darauf, keine auszuführenden Programmzeilen in diesen Kommentar einzufügen. Andererseits können Sie `/* . . . */` nutzen, um Teile des Quellcodes auszukommentieren. Sie können dadurch prüfen, wie der Code ausgeführt wird, wenn diese Zeilen entfallen.

9.4 Überarbeitung: Grundlagen der Programmierung in PHP

In den Kapiteln zu HTML und CSS haben Sie gelernt, wie Sie Bereiche einer Webanwendung programmieren können und wie Sie diese gestalten können. Jetzt kommen wir dazu, was wir mittels Programmierung in PHP tun müssen, um die Inhalte einer Webpage dynamisch (also zu beliebigen Zeitpunkten) ändern zu können.

Die folgenden Grundlagen finden Sie in ähnlicher Form in einer Vielzahl von Programmiersprachen, die zum imperativen Paradigma gehören. Wie schon in den ersten Kapiteln erklärt denken deshalb viele Programmierer irrtümlicherweise, dass sie wissen, was Programmieren ist, und dass es ausreichen würde, immer nur die Syntax einer Sprache zu erlernen, um in dieser (neuen) Sprache zu programmieren. Aber wie Sie dort lesen konnten ist das schlicht und ergreifend falsch und wird auch nicht dadurch richtig,

dass es immer wieder falsch wiederholt wird.

Stellvertretend möchte ich hier die Publikationen des heise-Verlags nennen: Dort wird nahezu durchgehend das Konzept der imperativen Programmierung als das einzige Konzept der Programmierung verklärt. Daraus folgen das Aussagen wie diejenige, dass es im Grunde genügen würde eine der dort vorgestellten Sprachen zu lernen, um alle Programmiersprachen nutzen zu können. (Sonderheft „Programmieren“ 2012) Wie schon erklärt ist das so abwegig, als würden Sie denken, dass ein Führerschein für PkWs genügen würde, um eine Planierdraupe sicher zu fahren. Was dabei herauskommt können Sie in Serien wie „Hör mal wer da hämmert“ sehen.

Um die Grundlagen der imperativen Programmierung und damit z.B. die Programmierung in PHP zu erlernen, stelle ich Ihnen sechs Themen vor: Variablen, Operationen, Kontrollstrukturen, Funktionen, Datenstrukturen und abschließend Schleifen bzw. Rekursionen:

- **Variablen** sind die Teile eines imperativen Programms, in denen Daten gespeichert werden.
Wichtig: Im Gegensatz zur Mathematik und z.B. zu Programmiersprachen, die das funktionale Paradigma umsetzen, darf der Wert einer Variablen bei prozeduralen Programmiersprachen wie PHP während des Ablaufs eines Programms geändert werden.
- **Operationen** sind einfache Anweisungen, mit denen die Inhalte von Variablen geändert werden können oder durch die neue Variablen erzeugt werden können.

Im Grunde könnten wir also mit Variablen und Operationen bereits Programme erstellen. Besonders spannend wären diese Programme aber nicht.

- **Kontrollstrukturen** ermöglichen es Ihnen, zu kontrollieren, unter welchen Bedingungen welcher Teil eines Programms ausgeführt wird.
- **Funktionen** fassen mehrere Operationen zusammen, um komplexe Abläufe als eine Einheit nutzen zu können. Da Funktionen einen Bezeichner haben, über den Sie aufgerufen („gestartet“) werden, können Sie sie an verschiedenen Stellen eines Programms ausführen lassen, ohne sie jedes Mal komplett neu programmieren zu müssen.
- Funktionen ähneln sehr den sogenannten **Prozeduren** und werden in vielen Sprachen auch nicht von diesen unterschieden. Der Unterschied ist der, dass eine Funktion einen Rückgabewert hat, dass also durch den Aufruf einer Funktion (evtl. mit einem oder mehreren Werten) ein Wert erzeugt wird. Im Gegensatz dazu ist eine Prozedur

schlicht eine Abfolge von Operationen, die nicht dazu dienen, einen Wert zu berechnen, der als Ergebnis der Prozedur zurück gegeben wird. In PHP unterscheiden wir nicht zwischen Funktionen und Prozeduren.

- **Datenstrukturen** fassen mehrere Variablen zusammen, die gleichartig sind. Und so, wie wir mit einer Funktion mehr Übersicht in unseren Programmablauf bekommen, helfen uns Datenstrukturen, wenn wir mit vielen Variablen arbeiten wollen oder müssen.
- **Schleifen** und **Rekursionen** ähneln Prozeduren und Funktionen. Der Unterschied besteht darin, dass eine Funktion bzw. eine Prozedur genau einmal ausgeführt wird, wenn Sie genutzt wird. Bei Schleifen und Rekursionen führen wir den Inhalten mehrfach aus, indem wir an einer Stelle angeben, wie oft sie ausgeführt werden sollen.
- Im nächsten Kapitel führen wir die sogenannten Datenbanken ein. **Datenbanken** sind dann sinnvoll, wenn wir nicht mehr nur Dutzende von Variablen haben, die wir in einer Datenstruktur gruppieren können, sondern wir über Tausende von Variablen in Dutzenden Datenstrukturen sprechen. Ein Beispiel wäre die Kundendatenbank eines Versandhändlers. Aber keine Sorge: Dafür sind Datenbanken zwar gedacht, aber wir können sie auch bei viel weniger Variablen nutzen, um uns die Nutzung anzusehen.

Datenbanken werden anhand von eigenen Programmiersprachen programmiert. Eine dieser Sprachen heißt MySQL. Und der Grund aus dem viele Bücher eine Einführung in PHP und MySQL anbieten ist der, dass wir bei der Programmierung in PHP (wie in vielen anderen Programmiersprachen) häufig Datenbanken benötigen, um effizient große Datenmengen zu verwalten.

Jetzt also zu den Variablen.

9.4.1 Variablen

Wie eingangs beschrieben sind Variablen die Teile von prozeduralen Programmen, in denen wir auf Daten zugreifen können. Sie kennen Variablen bislang aus dem Mathematikunterricht der Oberstufe, wo jede Variable im Regelfall für eine Zahl steht. Dabei haben sie gelernt, dass der Wert einer Variablen sich nicht ändern darf: Wenn x an mehreren Stellen einer Aufgabe auftaucht, dann hat es immer denselben Wert, auch wenn wir ihn noch nicht kennen.

Wichtig:

Variablen und insbesondere Variablen in der Form, wie Sie sie jetzt kennenlernen sind nur eine Spezialform von Daten mit denen Sie in bestimmten Programmiersprachen wie C, C++, Java, Ruby, Python und vielen anderen mehr arbeiten. Insbesondere, wenn Sie einen Informatikstudiengang besuchen ist es außerordentlich wichtig, dass Sie so bald als möglich das Studium von Teil 2 dieses Buches fortsetzen. Dort werden Sie eine wesentlich allgemeinere Form von Daten in Computern kennen lernen und nur wenn Sie diese wesentlich allgemeinere Form von Daten wirklich verstanden haben, haben Sie die Grundlage, um „echte/r“ InformatikerIn zu werden. So lange Sie dagegen auf die Programmierung fixiert bleiben, die Sie in diesem Teil des Buches kennen lernen (eben die imperative Programmierung), werden Sie zwar einen Computer dazu bringen können, Aufgaben für Sie zu erfüllen, aber Sie werden keinesfalls ernsthafte Informatik betreiben oder gar ein umfangreiches Verständnis dessen haben, was Programmierung alles ist.

Beachten Sie jedoch auch, dass Sie nach Ende des gesamten Buches lediglich die Programmierung mit Sprachen erlernt haben werden, die die diskrete Mathematik zur Grundlage haben. Die Programmierung kontinuierlicher Systeme wie Steuer- und Regelsysteme sind eine Kernkompetenz von Elektro- und Nachrichtentechnikern, auch wenn InformatikerInnen mit einer entsprechenden Vertiefung in diesem Bereich gleichwertige Fachkräfte werden können.

Erzeugung einer Variablen

Wie alles andere fallen auch Variablen nicht vom Himmel, sondern wir müssen dem Rechner mitteilen, dass wir eine Variable benötigen. Wir können dazu an einer beliebigen Stelle in unserem Programm einen Namen festlegen und diesem einen Wert zuordnen. Damit das Programm erkennt, dass es sich um eine Variable handelt, müssen wir lediglich ein `$`-Zeichen an den Anfang des Variablennamens setzen.

Das ist alles.

Wenn wir außerdem einen Startwert vorgeben wollen, dann können wir das tun. Hier wie an jeder Stelle, an der wir den Wert einer Variablen festlegen bzw. ändern wollen, nutzen wir dazu ein einfaches Gleichzeichen.

Wenn Sie eine Variable für einen Text (das kann auch HTML-Code sein) erzeugen wollen, dann schreiben Sie diesen „Wert“ der Variablen in Anführungszeichen.

```
// Erzeugung einer Variablen mit der Bezeichnung name.  
$name;  
  
// Erzeugung einer Variablen, der der Wert 3.14  
zugeordnet wird.  
$pi = 3.14;  
  
$quellcode = "<p>Wir brauchen die Vorrats-  
datenspeicherung, weil wir die Vorratsdaten-  
speicherung brauchen.</p>";
```

Quellcode 4.4: Erzeugung von Variablen in PHP

Wenn Sie den Namen einer Variablen speichern und ihn dann später nutzen wollen, geht das auch:

```
$name = "var";  
$var = 12;  
echo($$name);
```

Die letzten Zeile bewirkt, dass nicht etwa var sondern 12 ins HTML-Dokument eingefügt wird.

Statische und dynamische Variablen

Wir unterscheiden zunächst zwischen konstanten bzw. statischen Variablen, die ihren Wert nicht ändern können und dynamischen Variablen, deren Wert sich jederzeit ändern kann. Konstante Variablen verwenden wir meist nur dann, wenn es wichtig ist, dass sich der Wert dieser Variablen nicht ändern darf oder es einfacher ist, einen Wert durch eine kurze Bezeichnung darzustellen. Letzteres reduziert die Wahrscheinlichkeit für Fehler im Programm.

Deshalb gibt es auch keinen besonderen „Befehl“, mit dem wir dem Programm mitteilen, dass eine Variable dynamisch ist. Bei statischen Variablen, die in PHP als Konstanten bezeichnet werden, müssen wir dagegen das Schlüsselwort `const` verwenden, dafür lassen wir das `$`-Zeichen am Anfang des Variablennamens weg:

```
const pi=3.14159265358979323846264338327950;
```

Quellcode 4.5: Erzeugung einer Konstanten mit Zuordnung eines Wertes.

9.4.2 Datentypen

Im Gegensatz zu Sprachen wie C oder Java lässt PHP zu, dass sich der Datentyp einer Variablen jederzeit ändern kann. Jetzt fragen Sie sich vielleicht,

was denn der Datentyp ist.

Alle Daten haben einen Typ (eben den Datentyp). Denken Sie hier an Dinge wie Zahlen, Buchstaben, Symbole und ähnliche Arten von Zeichen, die Sie aus dem Alltag kennen. Es geht also um Kategorien von Dingen, die der Computer speichert. Im Gegensatz Dingen Ihres Alltags richten sich Datentypen aber danach, wie der Computer Daten intern speichert und wie er sie interpretieren soll.

Im Kern speichern Computer alles in Form von Zahlen zwischen 0 und 255. Reicht dieser Zahlenbereich nicht, dann nimmt ein Computer eben mehrere dieser Zahlen und behandelt sie wie eine gemeinsame Zahl. Nichts anderes tun wir, wenn wir rechnen: Wir nehmen die Zahlen von 0 bis 9 und wenn die nicht reichen, nehmen wir eine zweite Zahl von 0 bis 9 usw. Nur haben wir für andere Zwecke noch andere Bereiche, mit denen wir arbeiten: Die Buchstaben von a bis z. Ein Computer kennt dagegen nur seine Zahlen von 0 bis 255. (Genauer dazu lernen Sie in den Veranstaltungen zur technischen Informatik und der Kommunikations- bzw. Nachrichtentechnik.)

Der Computer kann also im Grunde einen Buchstaben auch nur als Zahl speichern. Er kann auch keine Nachkommastellen als Nachkommastellen speichern. Nochmal: Alles was er hat sind Speichereinheiten, die jeweils eine Zahl von 0 bis 255 speichern können.

Wenn wir also so etwas wie einen Buchstaben eingeben, dann speichert der Computer ihn als eine Zahl in einer Speicherstelle. Und damit er weiß, dass er diese Zahl in diesem Fall als Buchstaben interpretieren muss, braucht er noch einen weiteren Speicher, in dem er genau das speichert. Wir als Softwareentwickler bezeichnen das als Datentyp. Für uns ist es egal, wie der Computer den Datentyp realisiert, wichtig ist nur, dass er nicht plötzlich anfängt, unsere Buchstaben als Zahlen auszugeben und umgekehrt. In diesem Kurs werden wir uns mit dem Mechanismus, der hinter Datentypen steht nicht weiter auseinander setzen. Wichtig ist nur, dass Sie grundsätzlich verstehen, warum jede Variable einen Datentyp hat.

Statische und dynamische Datentypen: Typecasting

Sprachen wie PHP bieten nicht nur dynamische Variablen an, also Variablen deren Wert sich ändern kann. Zusätzlich kann sich auch der Datentyp einer Variablen ändern. Nehmen wir an, Sie dividieren eine ganze Zahl in einem PHP-Programm und als Ergebnis erhalten Sie eine Fließkommazahl. Während dieser Unterschied für Sie kein merklicher Unterschied ist, ändert sich in diesem Moment der Datentyp des Wertes, denn der Computer muss eine Fließkommazahl anders speichern, da er ja wie besprochen nur ganze

Zahlen von 0 bis 255 nutzen kann, um alles zu speichern, dass er speichern soll.

Sprachen, bei denen wie in PHP der Datentyp geändert werden kann werden auch als dynamisch typisierte Programmiersprachen bezeichnet.

Bei statisch typisierten Sprachen wie C und Java müssten Sie nun eine neue Variable erzeugen, der Sie den neuen Wert zuordnen. Denn dort kann sich der Datentyp einer Variablen nicht ändern. Bei dynamisch typisierten Sprachen ist das anders: Dort ändert sich der Datentyp je nachdem, wie sich der Wert einer Variablen ändert.

Das Ändern des Datentyps wird auch als Typecasting bezeichnet.

Erfahrene Entwickler in statisch typisierten Sprachen sagen deshalb häufig, dass dynamisch typisierte Sprachen unsicher seien, weil man je nie wissen könne, welchen Datentyp eine Variable hat. Aber das ist Unsinn, jeder Softwareentwickler sollte wissen, wie in einer dynamisch und wie in einer statisch typisierten Sprache programmiert werden muss. Außer natürlich es ist jemand, der in nicht-imperative Programmiersprachen Software entwickelt oder die nur in einer Sprache programmiert. Denn dann ist dieser Unterschied irrelevant.

In kurzen Worten: Wenn Sie es mit Programmierern zu tun haben, die darauf beharren, dass dynamisch typisierte Sprachen unsicher sind, weil diese keine Typsicherheit bieten, dann behandeln Sie diese Menschen am besten wie Geisteskranke: Immer nett zu ihnen sein und halten Sie sie von allem fern, was sie nicht verstehen. Genau wie manche Authisten können einige dieser Menschen in einem kleinen Spezialgebiet wirklich großartige Dinge leisten. Aber der Rest stellt für sie eine vollständige Überforderung dar. Das Problem besteht allerdings darin, dass es nur wenige Authisten gibt, während Menschen mit der beschriebenen beschränkten Auffassung von Programmierung die Norm darstellen. Wenn Sie nichts über Authismus wissen, sehen Sie den Oscar-prämierten Film „Rainman“. Dieser zeigt zwar ein sehr einseitiges Bild der Krankheit, aber für nicht-Psychologen und zum Verständnis des Vergleichs sollte das genügen.

9.4.3 Operationen

Operationen entsprechen weitestgehend dem, was Sie in der Oberstufenmathematik kennen gelernt haben, gehen aber darüber hinaus: Sie nehmen zwei Operanden, verknüpfen diese mithilfe eines Operatoren und bekommen ein Ergebnis, indem Sie den gebildeten Ausdruck auswerten. Komplexe Operationen (also Operationen mit mehreren Operanden) bilden Sie,

indem Sie Klammern verwenden.

Arithmetische Operatoren

Die vier Operatoren +, -, * und / entsprechen der Addition, Subtraktion, Multiplikation und Division. Der Doppelstern ** entspricht der Potenzierung.

Zusätzlich gibt es noch die Modulo-Operation, die in PHP durch das %-Zeichen ausgedrückt wird. Wenn Sie den entsprechenden Teil des Mathematikunterrichts inzwischen erfolgreich verdrängt haben, sollten Sie die alten Kenntnisse wieder aufwärmen; insbesondere im Bereich der IT-Sicherheit ist die modulare Arithmetik essentiell.

9.4.4 Anonyme Variablen

Oben haben Sie erfahren, wie Sie Variablen erzeugen. Dort haben Sie gelesen, dass jede Variable einen Namen hat. Aber was ist dann eine anonyme Variable?

Stellen Sie sich vor, Sie haben folgenden PHP-Code erstellt:

```
$a = 5;  
$b = 3;  
echo ($a + $b);
```

Quellcode 4.6: Anonyme Variable

Hier haben Sie zwei Variablen addiert und das Ergebnis ausgegeben. Das klingt logisch, aber hier fehlt noch etwas: Wenn ein Computer eine Operation ausführt, dann speichert er das Ergebnis irgendwo ab. Im Regelfall geben sie dieses „irgendwo“ vor, indem Sie dafür eine Variable erzeugen oder den Computer anweisen, den Wert einer anderen Variablen zu überschreiben. Das ist bei diesem Code-Beispiel nicht der Fall. Da wir also das Ergebnis der Operation nicht über einen Namen ansprechen können, reden wir hier von einer anonymen Variable. Quellcode 4.6: Anonyme Variable

Wenn wir keine anonymen Variablen verwenden wollen oder können, dann müssten wir das obige Programm so programmieren: Quellcode 4.6: Anonyme Variable

```
$a = 5;  
$b = 3;  
$c = $a + $b;
```



```
echo ($c) ;
```

Quellcode 4.7: Nutzung einer Hilfsvariable, um keine anonyme Variable zu verwenden.

Wie Sie sehen hat dieses Programm keinen Mehrwert gegenüber dem mit der anonymen Variable, sondern ist nur länger und damit fehleranfälliger. Also nutzen sie anonyme Variablen, wenn es Sinn macht.

9.4.5 Operationen für Text-Variablen

Wenn Sie sich wundern, warum es Operatoren für Text-Variablen gibt, dann sehen Sie Variablen und Operationen noch so beschränkt, wie Sie sie intuitiv im Mathematikunterricht kennen gelernt haben. Deshalb hier nochmal eine allgemeinere Beschreibung, die für die Programmierung in prozeduralen Sprachen gilt: Variablen können beliebige Daten speichern und mit Operationen ändern Sie diese Daten oder erzeugen daraus neue Daten.

Eine einfache Operation für Texte ist die Konkatination, also die Verknüpfung von zwei Texten. Für diese wird der `.` (Punkt) verwendet.

Wichtig: Bis auf Variablen, die durch Konkatination entstehen sind bei `echo()` keine anonymen Variablen erlaubt. Eine Zeile wie `echo("Das Ergebnis ist: ".$a + $b);` wäre also nicht erlaubt.

```
\$name = ``Horst``;  
\$zustand = ``krank``;  
echo(\$name.``ist``.\$zustand);
```

Quellcode 4.8: Konkatination von Texten.

9.4.6 Nachteile der dynamischen Typisierung

Wie beschrieben wird der Datentyp bei dynamisch typisierten Sprachen von der Programmiersprache immer dann geändert, wenn das sinnvoll erscheint. Dafür gibt es natürlich festgelegte Regeln, die sich je nach Sprache unterscheiden. Überlegen Sie, was alles beim folgenden Code passieren könnte und probieren Sie es dann aus, um zu erfahren, wie PHP programmiert ist.

```
\$a = 2;  
\$b = 3;  
echo(\$a.\$b);
```

```
\$c = \$a.\$b;  
\$d = \$c*2;  
echo(\$d);
```

Quellcode 4.9: Beispiel für dynamische Typisierung

Codebeispiele wie dieses werden von vielen Programmierern als Beweis genutzt, dass die dynamische Typisierung unsicher ist. Alles was eine Beweisführung in dieser Art tatsächlich belegt, ist die Unfähigkeit der Betreffenden, über den eigenen Tellerrand zu schauen.

Kurzschreibweise für Zuordnung und Operation

Gerade bei der Konkatination werden Sie des Öfteren eine Variable mit dem neuen „Wert“ ergänzen wollen. Im folgenden Codebeispiel sehen Sie eine Kurzform, die Sie in diesen Fällen nutzen können.

```
\$return = ``Siggi``;  
\$taetigkeit = ``geht shoppen.``;  
\$return = \$return.\$taetigkeit;  
  
// Die letzte Zeile können Sie verkürzen:  
  
\$return .= \$taetigkeit;
```

Quellcode 4.10: Verkürzte Form für Zuordnung und Operation.

Aufgabe

- Probieren Sie aus, für welche Operationen das funktioniert.

9.5 Überarbeitung: Kontrollstrukturen und boolesche Ausdrücke

Wie eingangs erläutert kommen wir jetzt zum ersten Teil der Programmierung, mit dem Sie tatsächlich eine Logik in Ihr Programm bekommen können. Während Sie also mit den vorigen Elementen von PHP kaum einen Mehrwert gegenüber der Programmierung in HTML erreichen konnten kommen wir jetzt zu den eigentlich spannenden Aspekten bei der Webanwendungsentwicklung mit PHP.

9.5.1 Grundidee

Nachdem Sie jetzt wissen, wie Sie Werte in PHP (und anderen prozedurale Sprachen) über Variablen speichern und ändern können, kommen wir jetzt dazu, wie Sie abhängig vom Ergebnis einer Operation unterschiedliche Programmteile ausführen lassen. Ob diese dann eine Ausgabe erzeugen, die als Quellcode in die Webpage eingefügt werden oder ob dadurch die Werte von Variablen geändert werden ist für diese Grundidee belanglos.

9.5.2 Vergleichsoperationen

Bei den Variablen haben Sie Operatoren kennen gelernt, die die Werte von Variablen nutzen, um neue Variablen zu erzeugen oder den Wert von bestehenden Variablen zu ändern. Wenn wir dagegen nur wissen wollen, ob z.B. `$a` größer als `$b` ist oder ob `$name` der Name des Administrators ist, dann wollen wir ja keine Variablen ändern. Vielmehr möchten wir hier Operatoren nutzen, um zu entscheiden, wie das Programm weiter ausgeführt wird. Aber auch dafür nutzen wir Operationen, die wie alle Operationen eine anonyme Variable erzeugen. Diese Operationen heißen Vergleichsoperationen.

Auch hier begegnet Ihnen im Grunde nichts, was Sie nicht schon aus der Schulzeit kennen würden, nur die Schreibweise müssen Sie sich einprägen. Das einzige, was Sie anfangs verwirren wird ist die Tatsache, dass ein Vergleich auf Gleichheit durch ein doppeltes Gleichheitszeichen ausgedrückt wird, denn das einfache Gleichheitszeichen wird ja bereits verwendet, um einer Variablen einen Wert zuzuordnen.

Das Ergebnis von Vergleichsoperationen wird in aller Regel als anonyme Variable verwendet. Dennoch kann es vorkommen, dass Sie das Ergebnis speichern wollen, dass Sie also eine boolesche Variable erzeugen wollen, eine Variable, die wahr oder falsch sein kann. In diesen Fall gehen Sie wie bei anderen Zuordnungen von Variablen vor. Hier die Liste der Vergleichsoperatoren:

- `==` prüft auf die Gleichheit der Werte zweier Operanden. - `===` prüft außerdem, ob Sie den gleichen Datentyp haben. - `!=` prüft, ob zwei Operanden unterschiedliche Werte haben. - `!==` prüft außerdem, ob auch der Datentyp unterschiedlich ist. - `>` prüft, ob der erste Operand größer ist, als der zweite. - `>=` prüft, ob der erste Operand nicht kleiner ist, als der zweite. - `<` prüft, ob der zweite Operand größer ist, als der erste. - `<=` prüft, ob der zweite Operand nicht größer ist, als der zweite.

Sie finden, dass das zu viele Operatoren sind? Vollkommen richtig: Fürs

erste brauchen Sie nur `=`, `!=` und `;` oder `:`. Wie Sie in der technischen Informatik lernen (also in der Veranstaltung Informatik bzw. Informatik 1) brauchen Sie dazu noch Konjunktionen und Disjunktionen. Wie Sie die in PHP programmieren lernen Sie jetzt.

9.5.3 Conditionals 1 – if-then-else

Es gibt in PHP zwei Möglichkeiten, um Kontrollstrukturen zu programmieren. Beide werden im Englischen als Conditionals bezeichnet.

Um eine Kontrolle durchzuführen, überlegen Sie sich alle möglichen Fälle und erstellen für jeden Fall einen sogenannten if-Zweig. Wenn Sie mehrere Fälle nicht mit einem eigenen if-Zweig behandeln, dann haben Sie zwei Möglichkeiten: Entweder Sie ignorieren diese Fälle ganz, lassen also das Programm einfach weiterlaufen oder Sie programmieren einen else-Zweig, in dem all das passiert, was eben in den verbliebenen Fällen passieren soll.

Die Bedingung, die geprüft werden soll steht in runden Klammern hinter dem „Befehl“ `if`. Das Schlüsselwort „then“ gibt es in PHP nicht. Wenn ein Programmteil bei einem if-Zweig ausgeführt werden soll, dann steht er direkt hinter dem Conditional des `if`.

Rümpfe

Bislang haben Sie immer nur Fälle kennen gelernt, in denen Ihr Programm Zeile für Zeile abgearbeitet wird. Jetzt kommen wir dagegen zu Fällen, in denen mehrere Zeilen nur in bestimmten Fällen ausgeführt werden sollen. In anderen Worten: Wenn diese Fälle nicht eintreten, dann sollen die entsprechenden Zeilen übersprungen werden. Also brauchen wir eine Möglichkeit, um im Quellcode anzuzeigen, welche Teile ggf. übersprungen werden sollen. Solche zusammengehörenden Mehrzeiler werden als Rumpf (z.B. Rumpf einer Funktion) bezeichnet. In PHP werden Rümpfe durch geschweifte Klammern markiert:

```
\{ ... \}
```

Anstelle der Punkte wird schlicht der Programmcode eingetragen.■

Der folgende PHP-Code ist ein einfacher if-then-else-Fall:■

```
if (\$login=='admin')
```

```
\{
```

```
echo('`<!-- Hier den Code einfügen, damit der Nutzer auf die Administrat
```

```
\}
```

```
else
```

```
\{
```

```
echo('`Die angeforderte Seite steht nur Administratoren zur Verfügung.`')
\}
```

Quellcode 4.11: Einfaches if Conditional

9.5.4 Conditionals 2 – Switch-Cases

Wenn-Dann-Kontrollstrukturen sind leicht zu verstehen, aber gerade wenn Sie den weiteren Programmablauf von der Belegung einer Variablen steuern wollen, sind die Switch-Cases häufig mit weniger Programmieraufwand verbunden.

Dabei tragen Sie in den Klammern hinter dem „Befehl“ switch den Namen der Variablen ein und bei den einzelnen Cases den Wert der Variablen. Hier ein einfaches Beispiel, das genau dasselbe bewirkt wie der if-then-else-Fall oben:

```
switch(\$login)
\{
case('`admin`'):
echo('`<!-- Hier den Code einfügen, damit der Nutzer auf die Administrat
break;
default:
echo('`Die angeforderte Seite steht nur Administratoren zur Verfügung.`')
\}
```

Quellcode 4.12: Einfaches Switch-Case-Conditional

Im Gegensatz zum if-then-else-Conditional wird bei Switch-Cases der default-Fall immer ausgeführt. Für jeden Fall, bei dem Sie das vermeiden wollen müssen Sie noch die Anweisung break ans Ende des jeweiligen Cases programmieren. Das ist auch der Grund, warum Sie bei Switch-Cases keine geschweiften Klammern benötigen.

Letztlich bleibt es Ihnen überlassen, ob Sie nun if-then-else oder Switch-Case verwenden. Aus Gründen der besseren Lesbarkeit sollten Sie allerdings immer dann Switch-Cases verwenden, wenn das möglich ist, da hier der Zusammenhang zwischen den Fällen leicht erkennbar ist.

Noch ein abschließender Hinweis: Ob Sie nun (wie bei den obigen Beispielen) für die geschweiften Klammern eine eigene Zeile reservieren oder nicht, ist Geschmacksfrage. Die folgenden Beispiele sind genauso gültig wie die beiden obigen. Beachten Sie bitte noch, dass das Weglassen der geschweiften Klammern bei if-then-else nur dann erlaubt ist, wenn der Fall

mit einer Programmzeile erfüllt wird.

```
if (\$login=='admin') \{
echo('`<!-- Hier den Code einfügen, damit der Nutzer auf die Administrat
\} else \{
echo('`Die angeforderte Seite steht nur Administratoren zur Verfügung.`')
\}

if (\$login=='admin') echo('`<!-- Hier den Code einfügen, damit der Nu
else echo('`Die angeforderte Seite steht nur Administratoren zur Verfügun

switch(\$login) \{
case('`admin') : echo('`<!-- Hier den Code einfügen, damit der Nutzer au
break;
default: echo('`Die angeforderte Seite steht nur Administratoren zur Verf
\}
```

Quellcode 4.13: Eine Frage des persönlichen Geschmacks: Position der geschweiften Klammern.

Neben den hier vorgestellten Möglichkeiten if-then-else oder Switch-Cases zu programmieren, gibt es noch eine weitere Möglichkeit, die Sie dann nutzen können, wenn Sie sehr umfangreichen HTML-Code generieren wollen. Hier unser obiges Beispiel mit den nötigen Änderungen innerhalb eines HTML-Dokuments:

```
<?php
...
if (\$login=='admin') \{ ?>
<!-- - Hier den Code einfügen, damit der Nutzer auf die Administrationsobe
<?php \} else \{ ?>
<p>Die angeforderte Seite steht nur Administratoren zur Verfügung.</p>■
<?php \} ?>
```

Quellcode 4.14: Einfaches if Conditional, wenn umfangreicher HTML-Code zu übergeben ist

Prüfung mehrerer Bedingungen

Wenn Sie nun einen bestimmten Fall nur dann ausführen wollen, wenn mehrere Bedingungen oder eine von mehreren Bedingungen erfüllt sind, gibt es zwei Möglichkeiten: (Um den Code übersichtlich zu halten, werden jetzt keine konkreten Fälle genannt, sondern es wird für jede Bedingung

einfach der Buchstabe B und eine Zahl verwendet. B5 ist also im Folgenden die fünfte Bedingung, wobei hier nicht von Belang ist, was diese Bedingung ist.

Mehrere Bedingungen müssen gelten

Nehmen wir an, drei Bedingungen (B1, B2 und B3) sollen gelten, damit ein Programmteil ausgeführt wird. Dann können wir verschachtelte if-then-else-Fälle programmieren, um das umzusetzen:

```
if(B1)\{ if(B2) \{if(B3) \{ /* Hier steht der auszuführende Code */ \}\}\}
```

Quellcode 4.14: Verschachteltes if

Leider ist diese Form aber eher unübersichtlich. Deshalb gibt es den Konjunktionsoperator: && (Doppeltes Und-Symbol) Wenn wir das einsetzen, wird unser Code besser lesbar:

```
if(B1 \&\& B2 \&\& B3)\{ /* Hier steht der auszuführende Code. */ \}
```

Quellcode 4.15: Konjunktion bei if-Cases

Eine von mehreren Bedingungen muss gelten

Wenn dagegen eine der drei Bedingungen gelten soll, aber in jedem Fall der gleiche Code ausgeführt werden soll, erhöht die erste Variante die Fehleranfälligkeit unseres Codes massiv:

```
if(B1)\{ /*Hier steht der auszuführende Code. */ \}  
if(B2)\{ /* Hier steht er nochmal. */ \}  
if(B3)\{ /* Und hier schon wieder. */ \}
```

Quellcode 4.16: Disjunktive if-Cases

In diesen Fällen ist es ein must-have, Disjunktionen anzuwenden. Die werden als — — programmiert:

```
if(B1 || B2 || B3)\{ /* Hier steht der auszuführende Code. */ \}
```

Quellcode 4.17: Disjunktion von Bedingungen

Alles weitere in Bezug auf die Kombinationsmöglichkeiten von Konjunktionen und Disjunktionen lernen Sie in den Veranstaltungen zur technischen Informatik und den einführenden mathematischen Veranstaltungen Ihres Studiums.

Aufgabe:

- Erweitern Sie Ihre Seite wie folgt: (Lesen Sie zunächst alle Aufgabenteile einmal durch. Es ist nicht gesagt, dass Sie die einzelnen Teile in genau der Reihenfolge umsetzen müssen, die hier vorgegeben ist.)

- Programmieren Sie eine Variable \$login, die standardmäßig den Wert guest hat.
- Programmieren Sie außerdem Teile Ihrer Webpage, sodass diese nur angezeigt werden, wenn \$login den Wert admin hat. Dazu benötigen Sie die CSS-Property visibility. Es sollte sich zwar von selbst verstehen, aber zur Sicherheit sei auf folgendes hingewiesen: Recherchieren Sie dazu, welche Belegungen visibility haben kann. Denn nur mit diesem Wissen können Sie die Aufgabe erfüllen.
- Programmieren Sie Teile Ihrer Webpage so um, dass sie nur dann angezeigt werden, wenn \$login den Wert user oder admin hat. Damit ist gemeint, dass diese Teile sowohl für user als auch für admin angezeigt werden sollen.

Hinweis: Da Sie noch keinen Log-in entwickelt haben, können Sie zurzeit noch nicht prüfen, ob Sie alles richtig gemacht haben. Diese Aufgabe dient vorrangig dazu, dass Sie selbst prüfen können, ob Sie die Inhalte der vorigen Kapitel soweit verinnerlicht haben, dass Sie sinnvoll damit arbeiten können:

- Wenn Sie Probleme damit haben, zu verstehen, was Sie tun sollen, sollten Sie dringend die bisher behandelten Themen wiederholen. Versuchen Sie vor allem zu verstehen, was der Sinn der einzelnen Abschnitte bzw. der darin behandelten Themen ist. Wenn Sie die bislang behandelten Themen nur überflogen haben, dann sollten Sie das dringend abstellen oder sich nach einem Ausbildungsplatz umsehen: Studieren bedeutet, sich intensiv mit den Inhalten von Veranstaltungen auseinander zu setzen... und zwar vor und nach den Veranstaltungen... mehrere Stunden pro Woche.

- Wenn Sie dagegen ein Problem damit haben, das Verstandene in ein Programm umzusetzen, dann bedeutet das lediglich, dass Sie mehr Übung brauchen. Andererseits ist das das Hauptproblem der meisten Studienanfänger: Sie wenden das Gelernte zu selten an und verinnerlichen es deshalb nicht.

- Wenn Sie sich dagegen wundern, warum Sie solche einfachen Schritte programmieren sollen, bzw. warum das hier so detailliert steht, dann sind Sie entweder schon deutlich weiter oder Sie sollten sich einen systematischeren Arbeitsstil angewöhnen. Denn auch dafür ist diese Übung gedacht.

Grundsätzlich gilt aber im Studium immer: Es ist gerade zu Beginn vollkommen normal, sich von den Inhalten und Übungen überfordert zu fühlen. Wichtig ist es dann, konzentriert weiter zu arbeiten. Außerdem ist es vollkommen normal, einzelne Veranstaltungen erst im dritten Anlauf zu bestehen. Manche Inhalte brauchen viel Zeit, damit Sie sie wirklich verinnerlicht haben.

9.6 Überarbeitung: Funktionen

Wenn die Seiten Ihrer Webpage immer umfangreicher werden, werden Sie früher oder später merken, dass Teile Ihrer Seite sich wiederholen. So kann es sein, dass Sie mehrere `summary`-Container programmiert haben, die nur von bestimmten Nutzern geöffnet werden dürfen. Hier wäre es unschön, wenn Sie jedes Mal die entsprechende Kontrollstruktur in PHP programmieren würden. Für solche Fälle gibt es die sogenannten Funktionen, die bei der objektorientierten Programmierung als Methoden bezeichnet werden.

Funktionen kennen Sie aus der Oberstufen-Mathematik. Dort tauchten Sie in Formen wie $f(x)$ oder ähnlichem auf. Bei der Programmierung in PHP können Sie längere Bezeichnungen als ein einfaches f verwenden. Dadurch ist es möglich, dass Sie eine Funktion so benennen, dass bereits aus dem Namen hervorgeht, was diese Funktion tun soll.

Im Gegensatz zu dem, was Sie in der Mathematik kennen gelernt haben steht es Ihnen hier aber frei, ob und wenn ja, wie viele Parameter einer Funktion übergeben werden sollen. Denn da Ihr Programm ja bereits Variablen hat, können Sie aus einer Funktion auf diese Variablen zugreifen.

Jede Funktion besteht aus einem Namen und einem Klammernpaar, das direkt neben dem Namen steht. In dieses Klammernpaar programmieren Sie dann all die Parameter, die von der Funktion verarbeitet werden sollen.

Mindestens eine solche Funktion haben Sie schon benutzt.

Aufgabe:

- Wie heißt diese Funktion?

(Wenn Sie sich mit der Antwort schwer tun, sehen Sie nochmal in Ihren bisherigen Programmen nach, wo etwas auftaucht, das aus einem Wort und einer anschließenden Klammer besteht.)

- Was tut diese Funktion?

9.6.1 Was ist ein Parameter?

Der Parameter der Funktion `echo()` ist immer ein String. Wenn Sie hier eine Zahlenvariable (nennen wir sie einfach `$zahl`) einsetzen, dann wird der Wert von `$zahl` in einer neuen Variable gespeichert, deren Bezeichner Sie nicht kennen und auch nicht kennen brauchen, weil für Sie nur wichtig ist, dass `echo()` diesen Wert ausgibt. Diese (für uns anonyme) Variable wird nun zu einem String gecastet. Das ändert jedoch nichts am Datentyp oder Wert der Variablen `$zahl` selbst.

Beim Aufruf einer Funktion brauchen wir also nichts über die Bezeichnungen der Variablen zu wissen, die innerhalb der Funktion verwendet werden. Diese Tatsache wird später bei Rekursionen wichtig: Bislang haben Sie gelernt, dass Sie den Bezeichner einer Variablen im gesamten Programm nur einmal verwenden dürfen, weil Sie sie an jeder Stelle des Programms nutzen können. Jetzt kommt eine Einschränkung für diese Aussage: Wenn eine Variable nur innerhalb eines Funktionsrumpfes vorkommt, dann kann Sie außerhalb des Funktionsrumpfes nicht verwendet werden, weil sie dort unbekannt ist. Das bedeutet auch, dass Sie eine Funktion mehrfach aufrufen können, denn die Variablen, die innerhalb eines Funktionsaufrufes verwendet werden, werden in einem eigenen Speicherbereich gespeichert. Wenn Sie das momentan noch nicht so recht einordnen oder verstehen können, lassen Sie sich davon nicht irritieren; wir kommen beim Thema Rekursion darauf zurück.

Wenn Sie dagegen eine Funktion selbst programmieren wollen, dann müssen Sie darin Parameter selbst programmieren.

Als Parameter werden Variablen bezeichnet, die der Zwischenspeicherung von Werten dienen, die bei einer Funktion verwendet werden. Im folgenden Funktionskopf wären das die Variablen `$param1`, `$param2` und `$param3`:

```
function meineFunktion(\$param1, \$param2, \$param3) \{ ... \}
```

Die Entwickler von PHP haben z.B. die Funktion `echo()` in etwa so programmiert:

```
function echo(\$param) \{  
/* Quelltext, der dafür sorgt, dass \$param in ein HTML-Dokument eingefügt  
\}
```

Bei der Programmierung des Funktionsrumpfes verwenden Sie dann die Parameter, als wären es ganz normale Variablen, die Sie an anderer Stelle

des Programms programmiert haben.

Wichtig: Beachten Sie bitte, dass diese Parameter außerhalb des Rumpfes nicht genutzt werden können. Um die Fehlerwahrscheinlichkeit zu reduzieren sollten Sie Variablen, die Sie außerhalb eines Funktionsrumpfes nutzen nicht als Parameter verwenden.

Für Fortgeschrittene: Es gibt auch die Möglichkeit, die Variable innerhalb einer Funktion zu überschreiben, von der der Parameter an die Funktion übergeben wurde. In diesem Fall wird bei PHP von einer Referenz auf die Variable gesprochen. Dabei kommt in PHP das &-Zeichen zum Einsatz. Da wichtig ist, dass Sie einen sauberen Programmierstil lernen, lassen wir diesen Fall hier außen vor. Wenn wir von der Nutzung von Parametern in Funktionen sprechen, so wie Sie es hier kennen gelernt haben, dann wird bei PHP von der Kopie einer Variablen gesprochen. Die Bezeichnung Parameter, wie in diesem Kurs verwendet ist dagegen in anderen Sprachen üblich. Deshalb werden wir in diesem Kurs dabei bleiben. Schließlich sollen Sie hier Dinge lernen, die in möglichst vielen Sprachen gelten.

Rückgabewert einer Funktion

Allerdings gibt es etwas, was Sie bei `echo()` nicht kennen gelernt haben, das Sie aber aus der Oberstufenmathematik kennen: Funktionen haben einen Rückgabewert. Das bedeutet, dass eine Funktion im Regelfall nicht nur etwas tut, bzw. etwas mit den übergebenen Werten tut, sondern dass sie auch ein Ergebnis zurückgibt. Dieses Ergebnis kann der Wert einer beliebigen Variablen sein. Es kann auch einfach eine boolesche Variable sein, die nur anzeigt, ob die Funktion wie gewünscht ausgeführt wurde und sonst nichts weiter mit dem genauen Ablauf der Funktion zu tun hat.

In diesen Fällen können Sie also bei einem Programm eine Funktion als Wert einer Variablen programmieren: `$a = eineFunktion();` ist somit eine legale Anweisung, die der Variablen `$a` den Wert zuordnet, der nach dem Aufruf von `eineFunktion()` zurückgegeben wird.

Aufgabe:

- Angenommen es gibt eine Funktion namens `derNutzerIstAdmin()`, die eine anonyme boolesche Variable zurückgibt, die dann wahr ist, wenn der angemeldete Nutzer ein Administrator ist. Wie müssten Sie den folgenden Code erweitern, damit der `summary`-Container nur dann angezeigt wird, wenn der Nutzer ein Administrator ist?

```
<?php
```

```

...
if( ..... ) \{ ?>
<summary> ....
<?php \} else ... \{ >
...
<?php \} ?>

```

Quellcode 4.18: Aufruf einer Funktion bei Conditionals

Übrigens müssen Sie natürlich für diesen Fall nicht unbedingt eine Funktion verwenden. Sie könnten auch nach dem Log-in des Nutzers einen Programmteil durchlaufen lassen, in dem geprüft wird, ob der Nutzer ein Administrator ist oder nicht. Und das Ergebnis dieser Prüfung könnten Sie dann einer Variablen zuordnen, die Sie anschließend weiter verwenden würden.

Das ist ein Beispiel für etwas, das Sie bei der Programmierung in jeder prozeduralen Programmiersprache (aber auch bei anderen Arten von Sprachen) beachten müssen: Vieles lässt sich auf unterschiedlichste Art und Weise realisieren. Und meist gibt es keine beste Lösung, sondern viele Lösungsmöglichkeiten, von denen einige besser, andere schlechter und viele aus den unterschiedlichsten Gründen sehr schlecht sind.

9.6.2 Eigene Funktionen programmieren

Wenn Sie einen Programmteil zu einer Funktion machen wollen, dann setzen Sie den gesamten Programmteil in geschweifte Klammern und schreiben in der Zeile davor das Schlüsselwort `function` und anschließend einen Funktionsnamen, der mit runden Klammern abgeschlossen wird.

```

function meineFunktion(\$p1, \$p2)\{
... // Hier programmieren Sie, was innerhalb der Funktion passieren soll,
// beliebiger Teil Ihres Programms.
// Allerdings können Sie dafür die Parameter \$p1 und \$p2 verwenden.■
// Die Werte dieser Parameter werden beim Funktionsaufruf vergeben.■
\}

```

Quellcode 4.19: Definition einer Funktion ohne Parameter

Wann immer Sie in Zukunft wollen, dass der Rumpf der Funktion aufgerufen wird, müssen sie nur noch den Funktionsnamen an der entsprechenden Stelle einprogrammieren.

9.7 Überarbeitung: Datenstrukturen

Dieser Abschnitt ist etwas problematisch, weil wir bislang vorrangig über die Möglichkeiten gesprochen haben, die die Sprache PHP Ihnen bietet. Datenstrukturen gehören genau wie Rekursionen in einen Bereich, bei dem viele Programmierer nur das nutzen, was die Programmiersprache Ihnen anbietet. Und genau dadurch können Sie einfache Programmierer von Menschen mit einem Verständnis für Informatik unterscheiden.

Zunächst einmal ist eine Datenstruktur etwas, womit Sie eine Vielzahl gleichartiger Variablen strukturiert als eine Einheit abspeichern können. Nehmen wir an, Sie haben eine Webpage mit einem Spiel entwickelt und wollen eine High-Score-Liste generieren.

Das würde natürlich voraussetzen, dass jeder Spieler einen Punktestand für sein Spiel bekommt. Soweit ist es auch kein Problem, aber alles, was Sie mit dem bisherigen Wissen tun können, ist den aktuellen Punktestand mit einem Namen jeweils in einer Variablen zu speichern. Sie könnten weiterhin für die zehn besten Spieler jeweils ein solches Variablenpaar erstellen. Aber Sie hätten keine Möglichkeit, um alle Ergebnisse von Spielern in dieser Weise zu speichern. Denn auch wenn Sie mit PHP eine dynamische Webanwendung entwickeln können, können Sie eben nicht im Vorfeld programmieren, dass Ihr Programm je nach Bedarf zusätzliche Variablen generieren soll.

Die Lösung für dieses Dilemma heißt Datenstrukturen. In aller Regel bieten prozedurale Sprachen Ihnen zumindest ein Array als Datenstruktur an, dass Sie ohne weitere Kenntnisse nutzen können. In Veranstaltungen wie Algorithmen und Datenstrukturen lernen Sie dann, wie Sie andere Datenstrukturen selbst programmieren können und wann welche Datenstruktur Sinn macht. Prozedurale Sprachen wie Pascal ermöglichen es Ihnen außerdem beliebige Datenstrukturen selbst zu programmieren.

Leider ist PHP derart beschränkt, dass Sie keine eigenen Datenstrukturen programmieren können. Erinnern Sie sich noch an meine Kritik an den „Vorteilen“ die der Autor Thomas Theis bezüglich PHP formulierte? Genau hier haben wir einen Fall, in dem es eben ein massiver Nachteil ist, dass PHP Sie derart in Ihren Möglichkeiten einschränkt. Seit Version 5.3 gibt es allerdings eine Reihe an Klassen in der PHP Standard Library, die weitere Datenstrukturen realisieren. Bei O'Reilly wird betont, dass es ein großer Vorteil ist, dass diese in C programmiert ist, was in einer höheren Geschwindigkeit resultiert. Nur bedeutet das für Sie, dass Sie hier keine individuelle Anpassung durchführen können.

In PHP können Sie deshalb leider nur zwei Arten von Datenstrukturen direkt verwenden: Felder und Assoziative Felder. Gleich vorweg: Fehlgeleitete Naturen bezeichnen Assoziative Felder als Hash-Tables. Ein Hash-Table ist aber etwas ganz anderes. Was genau, das lernen Sie z.B. in Veranstaltungen wie Algorithmen und Datenstrukturen. Und nein, das was bei Java als Hash-Table bezeichnet wird ist ebenfalls kein Hash-Table...

Nach dieser ausführlichen Einleitung, die vorrangig dazu dient, dass Sie sich im Laufe Ihres Studiums intensiv mit Algorithmen und Datenstrukturen beschäftigen, kommen wir nun zu den beiden Feldertypen, die PHP anbietet:

9.7.1 Felder bzw. Arrays

Ein Array ist eine Art Liste, bei der jeder Eintrag über eine Zahl angesprochen werden kann. Die Nummerierung beginnt jeweils bei 0, sodass Sie den fünfzigsten Wert eines Arrays über die Nummer 49 abrufen.

Leider wissen die Entwickler von PHP nicht, was ein Array ist, denn das, was sie als Array implementiert haben ist eine verkettete Liste mit Indizierung jedes Elements. Damit haben die Entwickler von PHP eine Datenstruktur entwickelt, die weder den Geschwindigkeitsvorteil eines Array bietet, noch die Flexibilität beim Einfügen und Entfernen von Elementen einer verketteten Liste.

Deshalb hier zunächst die Erklärung, was ein Array ist:

Ein Array ist ein Teil des Speichers eines Computers, der in gleichgroße Einheiten unterteilt werden, die genau so groß sind wie es für eine bestimmte Variable nötig ist. Denn bei der Programmierung eines Arrays wird am Anfang festgelegt, welchen Datentyp alle Variablen haben sollen.

Der Vorteil eines Array besteht darin, dass das Auslesen oder Überschreiben einer Stelle des Arrays sehr schnell zu erledigen ist. Dafür ist diese Datenstruktur sehr ineffizient, wenn wir ein Element löschen wollen oder ein neues Element hinzufügen wollen, denn beides erfordert bei einem Array einen sehr großen Kopieraufwand. Eine Vergrößerung eines Arrays (z.B. indem ans Ende noch zehn neue Elemente angehängt werden) ist dagegen unmöglich, weil ja in der Zwischenzeit der Speicher „hinter“ dem Array in aller Regel mit anderen Daten beschrieben wurde.

Auch hier gilt wieder: Wenn Sie das im Detail verstehen wollen, belegen Sie bitte einen Kurs in Algorithmen und Datenstrukturen.

Und jetzt zu dem, was Felder in PHP können, die dort als Arrays bezeichnet werden:

Genau wie bei echten Arrays können Sie dort einen Eintrag des Array mithilfe eines Indexes überschreiben oder auslesen. Sie können aber zusätzlich jederzeit neue Elemente ans Ende des Feldes anfügen.

Verstehen Sie die hier geäußerte Kritik bitte nicht falsch: Die Datenstrukturen, die PHP anbietet sind sinnvoll. Allerdings gibt es hier ein Problem: Sie lernen jetzt etwas unter dem Namen Array kennen, das kein Array ist. Wenn Sie dann eine andere Programmiersprache kennen lernen, die Ihnen ein echtes Array unter dem Namen Array anbietet werden Sie im schlimmsten Fall nicht verstehen, warum Sie es nicht so programmieren können, wie Sie das bei PHP gelernt haben.

Zum anderen müssen Sie bei komplexen Aufgabenstellungen mit hohen Datenmengen bestimmte Datenstrukturen verwenden, damit das Programm nicht zu langsam wird. Und genau das ist in PHP nicht möglich. Viele Webanwendung (oder auch Browsergames) sind also nicht deshalb langsam, weil die Internetanbindung zu langsam ist, sondern weil PHP mit der fehlenden Möglichkeit zur Implementierung von Datenstrukturen einen Flaschenhals darstellt.

Programmierung eines Arrays

Um ein PHP-Array zu erzeugen können Sie einerseits die Zuordnung `$name = array(ersterWert, zweiterWert, ...);` verwenden.

Sie können aber auch direkt über `$name[1] = zweiterWert;` ein Array erzeugen, indem Sie einem Eintrag des zu erzeugenden Arrays einen Wert zuordnen.

Um einen Eintrag auszulesen, verwenden Sie dann beispielsweise `$name[27]`, um das 28-igste Element des Array auszugeben. Sie erhalten durch diesen Zugriff wie gewohnt eine Variable. Aber diese ist nicht anonym, denn `$name[27]` ist genau wie `$a` ein Name für eine Variable. Dass es außerdem einen Zugriff auf das Element eines Arrays darstellt ist davon unabhängig.

Assoziative Felder (eigentlich Dictionary)

Der Unterschied zwischen einem Feld und einem assoziativen Feld besteht darin, dass die Indizes eines assoziativen Feldes Texte sind, also etwas, das zwischen "..." steht. Hier wird dann nicht mehr vom Index, sondern vom Schlüssel gesprochen. Und weil der Rechner nicht ahnen kann, welchen

Schlüssel Sie verwenden wollen, müssen Sie die Schlüssel bei der Erzeugung explizit angeben.

Zum Vergleich: Um Elemente in ein Array einzutragen, konnten Sie bei der Erzeugung des Arrays `$array = array("Kuchen", "Brötchen", ...);` eingeben. Dadurch wurde festgelegt:

```
$array[0]="Kuchen"; $array[1]="Brötchen";
```

Um ein assoziatives Feld zu erzeugen, können sie auch wieder auf zwei Arten vorgehen: Entweder Sie legen mehrere Einträge fest: `$name = array("Erster Eintrag" => 22, "Zweiter Eintrag" => 234234, ...);` Wenn Sie ein Array so programmieren könnten, dann würde das so aussehen: `$array = array(0 => "Kuchen", 1=> "Brötchen", ...);` Aber so programmieren Sie kein Array.

Einzelnen Eintrag für ein assoziatives Feld programmieren Sie so: `$name["Noch ein Eintrag" => 232];`

Der Folgefeil `=>` dient hier schlicht dazu, um aufzuzeigen, dass Sie einen Schlüssel und einen Wert hinzufügen, während Sie ja beim Feld einfach einen Wert hinzufügen, der dann in die nächste Zeile des Arrays eingefügt wird.

Wichtig: Wie beim „Array“ haben die Entwickler von PHP auch hier eine falsche Bezeichnung verwendet: Eine solche Datenstruktur wird als Dictionary bezeichnet (engl. für Wörterbuch, bzw. Nachschlagewerk).

Bei einem Dictionary handelt es sich um eine Ansammlung von sogenannten key-value-Paaren. Der key (dt. Schlüssel) ist ein beliebiger Wert, unter dem die Datenstruktur Werte oder Datenstrukturen als value (dt. Wert) speichert. Ein echtes Wörterbuch könnten sie beispielsweise so speichern: Sie erzeugen einen Dictionary, der die Buchstaben von a bis z als keys speichert. Unter jedem key könnten Sie nun einen weiteren Dictionary speichern, der ebenfalls die Werte von a bis z als keys hat, wenn es Wörter gibt, die mit den beiden keys der beiden Dictionaries anfangen. Und auch die values dieser Dictionaries hätten wieder jeweils einen Dictionary mit den keys von a bis z als Wert, usw. usf. Das wäre zwar nicht unbedingt effizient, aber wie gesagt könnten Sie so ein Wörterbuch als Datenstruktur speichern.

Aufgabe:

- Wie lautete der Befehl, um Wert zum Schlüssel "Horst" aus dem assozia-

tiven Feld `$Kneipengaeste` auszulesen?

9.8 Überarbeitung: Wiederholungen von Programmteilen (Schleifen und Rekursionen)

Im letzten Teil der Einführung in die prozedurale Programmierung mit PHP sprechen wir darüber, wie wir Teile unseres Programms so lange wiederholen können, bis eine bestimmte Bedingung erfüllt ist. In den meisten Einführungen zu diesem Thema lernen Sie lediglich Schleifen kennen. Leider haben Schleifen aber eine ganz böse Einschränkung, die bei den besagten Einführungen ignoriert wird: Sie ermöglichen es nicht, mehrere Ausführungen parallel zu erlauben. Hier sprechen wir allerdings nicht über „echte“ Parallelprogrammierung; die würde für einen Kurs im ersten Studienjahr etwas zu weit führen.

Damit Sie verstehen, wovon hier die Rede ist, folgt ein Beispiel, bei dem Wiederholungen eine Rolle spielen, die pseudo-parallel ausgeführt werden. Pseudo-parallele Abläufe sind Dinge, die unabhängig voneinander passieren können und scheinbar gleichzeitig ausgeführt werden. Nehmen wir an, Sie erstellen ein Programm für Vertreter eines Unternehmens. Diese Vertreter benötigen eine Reiseplanung, bei der sie jeweils Kunden besuchen, die in der Nähe voneinander wohnen. Nun sind die Daten der Kunden aber nach dem Namen sortiert. Was Sie also programmieren müssen ist eine Sortierung nach der Postleitzahl.

Und hier sind wir wieder bei einem Problem, das Informatikstudierende in einer Veranstaltung mit dem Namen Algorithmen und Datenstrukturen besprechen. Denn die Frage ist jetzt: Wie können wir dieses Umsortieren so durchführen, dass es auch bei Zehntausenden von Kunden noch schnell funktioniert.

Hier eine Übung, die Ihnen veranschaulicht, dass die zeitlichen Unterschiede beim Sortieren störend sein können:

- Ein naiver Ansatz zum Sortieren sähe so aus: Sie vergleichen die Postleitzahl des ersten Kunden mit der des zweiten Kunden und tauschen ggf. beide aus. Dann vergleichen Sie die Postleitzahl des zweiten Kunden (der evtl. ursprünglich der erste Kunde war und jetzt durch den Tausch zum zweiten Kunden wurde) mit der des dritten Kunden und machen so weiter, bis Sie einmal zum letzten Kunden gekommen sind. Das wiederholen Sie so oft, bis Sie keine Kunden mehr vertauschen. Nehmen wir an, Sie hätten eine Anzahl Kunden, die wir mit n bezeichnen. Dann müssen Sie im Extremfall (auch bekannt als worst case) n -mal $(n-1)$ Vergleiche (also n^2-n

Vergleiche) durchführen und jedes Mal einen Austausch durchführen. Bei zehntausend Kunden dauert das selbst auf einem schnellen Rechner knapp eine Minute. Es sind rund 108 Operationen.

So etwas können Sie mit den sogenannten Schleifen programmieren.

- Nehmen wir dagegen an, Sie teilen Ihre Kunden in zwei Gruppen, ohne dabei zu sortieren. Und jede dieser beiden Gruppen teilen Sie dann wieder in zwei Gruppen. Und das machen Sie so lange, bis Sie nur noch Gruppen mit einem oder zwei Kunden haben. Jetzt kommt der pseudo-parallele Teil: Nun sortieren Sie die Kunden jeder Gruppe. Das macht also in jeder Gruppe höchstens eine Sortier- und Austauschoperation. Und da wir durch die Teilungen genau halb so viele Gruppen wie Kunden haben, macht das maximal $n/2$ Vergleichsoperationen.

Im nächsten Schritt nehmen wir jeweils zwei unserer kleinen Gruppen. Hier vergleichen wir das erste Element der ersten Liste mit dem ersten Element der zweiten Liste. Das größere speichern wir in einer neuen Liste. Danach haben wir maximal noch zwei Vergleiche und schon ist aus unseren zwei kleinen Listen eine größere geworden. Da wir jetzt $n/4$ Listen haben, die jeweils maximal dreimal verglichen werden mussten, haben wir maximal $n/2 + n/4$ Operationen für diesen Schritt unserer Sortierung gebraucht. (Wenn Sie nachgerechnet haben, fragen Sie sich vielleicht, warum es nicht $3 * n/4$ sind. In dem Fall fragen Sie am besten Ihren Mathedozenten, damit er Ihnen nochmal das Bruchrechnen erklärt.) Bis jetzt haben wir also $2 * n/2 + 1 * n/4$ Operationen.

Von jetzt an führen wir den letzten Schritt immer wieder auf die neuen Listen aus, bis wir eine sortierte Liste erhalten. Wenn Sie dabei jeden Schritt genau notieren und die Anzahl Operationen berechnen, alles summieren und Ihre mathematischen Kenntnisse anwenden, erhalten Sie am Ende eine Anzahl, die sich in einen Ausdruck umformen lässt, der $n * \log_2 n$ ähnelt. Wenn wir nun wieder unsere 10.000 Kunden für n einsetzen, dann landen wir bei ca. $1,4 * 10^5$. Diese Sortierung ist also fast 1.000-mal so schnell wie die oben genannte. Während wir bei der Sortierung mit einer Schleife knapp eine Minute warten müssen, ist bei dieser Art der Sortierung das Ergebnis scheinbar sofort da. Was meinen Sie, welche Variante Ihr Auftraggeber haben möchte.

Obwohl auch diese Variante Wiederholungen beinhaltet, können Sie sie NICHT mit einer Schleife programmieren, sondern ausschließlich mit einer sogenannten Rekursion. Denn wie Sie sehen werden hier einige Aufgaben pseudo-parallel erledigt und das ist mit einer Schleife unmöglich.

Das was Sie gerade gesehen haben ist übrigens Informatik für Erstsemester, es ist also nicht allzu anspruchsvoll. Es wird als divide and conquer-Algorithmus bezeichnet. Wie Sie sehen hat das nichts mit der Beherrschung einer Programmiersprache zu tun, sondern mit Logik und Mathematik. Wenn Sie Informatik an einer Universität studieren würden, müssten Sie die mathematischen Teile selbst berechnen und anhand der Ergebnisse beweisen, wie viel effizienter die eine Variante gegenüber der anderen ist. Informatik-Studierende an einer FH müssen dagegen lediglich wissen, welche Effizienz welches Verfahren bietet. Denn Informatiker von Universitäten werden darauf vorbereitet, neue Verfahren zu entwickeln und müssen deshalb im Stande sein, formal zu beweisen, welchen Mehrwert ihr neues Verfahren hat. FH-Informatiker werden dagegen darauf vorbereitet, neue Verfahren sinnvoll in zum Teil neue Anwendungen zu integrieren. Also brauchen sie die Beweisführung nicht zu kennen, müssen aber die Effizienz eines Algorithmus kennen und in Abhängigkeit davon Algorithmen einsetzen. Wenn Sie dagegen einfach „nur“ programmieren wollen, dann suchen Sie eigentlich nach einer Ausbildung zum/zur Fachinformatiker/in.

9.8.1 Rekursionen

Da also Rekursionen wesentlich mehr bieten als Schleifen und meist auch wesentlich schneller arbeiten, kommen wir zuerst zu dieser Art der Wiederholung in Programmen. Rekursionen sind nur eine Sonderform von Funktionen und deshalb gibt es keine neuen Elemente einer Programmiersprache, die Sie jetzt lernen müssen. Das Besondere dabei ist, dass eine Rekursion eine Funktion ist, die nicht sofort einen Rückgabewert zurückgibt, sondern die so lange mit einem jeweils neuen Zwischenergebnis erneut aufgerufen wird, bis eine bestimmte Bedingung erfüllt ist. Das kann sie fast beliebig oft tun. Die maximale Häufigkeit folgt aus der Größe des Computerspeichers. Bei diesen Bedingungen spricht man von Abbruchbedingungen, denn nur wenn sie erfüllt werden hört die Rekursion bzw. der wiederholte Aufruf der Funktion auf.

Es gibt übrigens nichts, was Sie mit einer Schleife programmieren können, das nicht auch als Rekursion programmierbar wäre. Deshalb ist es eigentlich unsinnig, sich überhaupt Schleifen anzusehen, aber wir machen es natürlich trotzdem, weil viele Programmierer nur mit Schleifen programmieren können. Und natürlich würde es blöd aussehen, wenn Sie nicht wissen, was eine Schleife ist.

Die folgenden Beispiele zeigen, wie Sie einfache Rekursionen programmieren können: // Rekursion, die den Abstand zwischen zwei ganzen Zahlen berechnet.

```
function abstand(\$zahl, \$andereZahl){
  if (\$zahl > \$andereZahl) \{
    \$andereZahl += 1;
  \} else if (\$zahl < \$andereZahl) \{
    \$zahl += 1;
  \} else \{
    return 0;
  \}
  return 1 + abstand(\$zahl, \$andereZahl);
\}
\$a = 1;
\$b = 10;
\$c = abstand(\$a, \$b);
echo('`Der Abstand zwischen \$a und \$b beträgt: \$c.`');■
```

Quellcode 4.20: Rekursion um der Rekursion willen

Der Anfang der Funktion sollte Ihnen klar sein:

- Wir haben hier drei Fälle:

- $\$zahl < \$andereZahl$ • $\$andereZahl < \$zahl$ • $\$zahl == \$andereZahl$ (das ist der else-Zweig).

- In jedem der beiden ersten Zweige wird die Zahl um eins erhöht, die kleiner als die andere Zahl ist.

- Wenn der Abstand gleich Null ist, ist die Abbruchbedingung erreicht, also muss hier ein Wert zurückgegeben werden und die Funktion darf nicht nochmal aufgerufen werden.

- Die letzte Zeile der Funktion ist dann das, was die Funktion zu einer Rekursion macht: Hier wird die Funktion mit neuen Werten erneut aufgerufen. Aber warum steht hier $1 +$ vor dem Rekursionsaufruf? Ganz einfach: Da der Abstand zwischen den beiden Zahlen um 1 geringer geworden ist, müssen wir irgendwie speichern, dass er ursprünglich um ein 1 größer war. Und das passiert durch dieses $1 +$.

- Eine andere häufige Fragen an dieser Stelle lautet: Wie soll denn das Programm weiterlaufen, wenn es hinter dem return einen Funktionsaufruf hat? Das ist im Grunde ganz simpel: Das Programm gibt hier erstmal nichts zurück, sondern es führt quasi eine Zwischenspeicherung des Zustands durch, ruft dann die Funktion mit den neuen Parametern aus und erst, wenn von dieser Funktion ein Rückgabewert kommt, berechnet es den Rückgabewert dieser Funktion aus. Und das geht relativ oft (einige hundert- bis tausendmal).

Aufgabe

- Anhänger statisch typisierter Programmiersprachen würden diese Funktion zum Beweis nutzen, dass dynamisch typisierte Sprachen unsicher sind. ■
Was genau führt dazu, dass diese Funktion endlos laufen kann, bzw. mit einer Fehlermeldung abbricht?
- In anderen Worten: Was fehlt hier, das aber längst nicht bei jeder Rekursion nötig ist?

Nehmen wir jetzt unsere Sortieraufgabe. Da dieser Kurs sich an Einsteiger richtet, wird die folgende Rekursion in Pseudo-Code notiert; in PHP wäre sie umfangreicher und damit nicht mehr übersichtlich. Eine detaillierte Erklärung zum Ablauf folgt nach dem Pseudo-Code.

// Rekursion, die eine Liste von Zahlen nach dem devide & conquer-Ansatz ■
sortiert.

```
function sortList(Liste){
  if(Liste enthält mehr als eine Zahl){
    \$geordneteListe = teileListe(Liste)
  } else return Liste;
  return \$geordneteListe;
}
```

/* Diese Hilfsfunktion teilt die Liste in zwei Teillisten auf und ruft au

```
function teileListe(Liste){
  \$n = Anzahl Zahlen in der Liste;
  return sortListen(neue Liste mit den ersten n/2 Zahlen der Liste, neue Li
}
```

/* Zusammen mit teileListe() teilt diese Funktion die Liste so lange auf,

```
function sortListen(Liste1, Liste2)
  if(Liste1 enthält mehr als eine Zahl){
    \$geordneteListe1 = teileListen(Liste1)
  }
  if(Liste2 enthält mehr als eine Zahl){
    \$geordneteListe2 = teileListen(Liste2)
  }
  return mergeListen(\$geordneteListe1, \$geordneteListe2); ■
}
```

```

/* In der folgenden Funktion wird öfter += verwendet. Das soll in diesem

function mergeListen(Liste1, Liste2){
  if(Erste Zahl von Liste1 > Erste Zahl von Liste2) {
    \$geordneteListe += Erste Zahl von Liste1;
    Lösche die erste Zahl aus Liste1;
    if (Liste1 ist nicht leer){
      return \$geordneteListe += mergeListen(Liste1, Liste2);■
    } else return \$geordneteListe += Liste2;
  }
  else {
    \$geordneteListe += Erste Zahl von Liste2;
    Lösche die erste Zahl aus Liste2;
    if (Liste2 ist nicht leer){
      return \$geordneteListe += mergeListen(Liste1, Liste2);■
    } else return \$geordneteListe += Liste1;
  }
}

```

Pseudo-Code 4.21: Beispiel für eine Endrekursion.

Was passiert hier?

Anfangs wird die Funktion `sortListe` mit einer Liste aufgerufen. Da wir aber in Zukunft immer zwei Listen haben werden (schließlich bewältigen wir das Problem, indem wir die Liste immer wieder teilen) brauchen wir eine zweite Funktion, die bei jedem Aufruf zwei Listen annehmen kann. Bei solchen Fällen, in denen wir eine Hilfsfunktion vor der eigentlichen Rekursion haben reden wir von einer Endrekursion.

Die erste Funktion prüft zunächst, ob unsere Liste schon sortiert ist, was dann der Fall ist, wenn sie entweder leer ist oder nur ein Element enthält. Ist das der Fall, dann wird die Liste so zurückgegeben, wie sie ist. Das passiert oben im `else`-Zweig. Sonst wird die Liste in zwei Hälften zerlegt, die gleichgroß sind, bzw. deren Größe sich maximal um 1 unterscheidet. Diese beiden Listen werden nun an die eigentliche Rekursion übergeben, die die Listen immer weiter zerlegt. Diese Zerlegung passiert in der zweiten Funktion.

Die beiden ersten `if`-Cases der zweiten Funktion sorgen lediglich dafür, die Listen so lange zu teilen, dass am Ende nur mehr Listen mit einem Element existieren.

Damit kommen wir zur dritten Funktion `mergeListen()`, das den zweiten

Teil unserer Rekursion löst. `mergeListen()` vergleicht nur die erste Zahl der beiden übergebenen Listen und fügt das größere der beiden in eine andere Liste an, die `$geordneteListe` heißt und die später zurückgegeben wird. Danach wird diese größte Zahl aus der ursprünglichen Liste gelöscht. Wenn in beiden Listen noch wenigstens eine Zahl ist, werden die beiden Listen mit `mergeSort()` aufgerufen und das Ergebnis dieses rekursiven Aufrufs wird an `$geordneteListe` angehängt.

Ist dagegen eine der beiden Listen leer, dann wird einfach der Rest der verbliebenen Liste an `$geordneteListe` angehängt, bevor `$geordneteListe` zurückgegeben wird. An dieser Stelle fragen Sie sich vielleicht, wo denn die rekursiven Schritte nach dem Zusammenfügen (mergen) der Listen auftauchen, die aus mehr als einem Element bestehen. Schauen wir uns dazu den Ablauf an, nachdem aus jeweils zwei Listen mit einem Element eine sortierte Liste mit zwei Elementen geworden ist:

Der Rückgabewert (also unsere sortierte Liste) landet in der Funktion `sortListen()`, die ihn wiederum an die Funktion `teileListen()` zurückgibt. Und dort wird dieser Rückgabewert entweder als `$geordneteListe1` oder als `$geordneteListe2` gespeichert. Damit wird diese kurze geordnete Liste zusammen mit einer anderen geordneten kurzen Liste an die Funktion `mergeListen()` übergeben, die dann diese beiden kurzen zu einer längeren geordneten Liste zusammenfügt.

Ganz zum Schluss (wenn die ganzen Teillisten endlich wieder zu einer zusammengefügt wurden), landet der Rückgabewert von `mergeListen()` wieder im `if`-Case unserer anfänglichen Hilfsfunktion `sortListe()` und wird von dieser schließlich als Rückgabewert ausgegeben.

Aufgabe

- Programmieren Sie die oben in Pseudo-Code gegebene Rekursion mit einem Array in PHP.
- Begründen Sie, warum diese Rekursion im Gegensatz zum vorigen Beispiel nicht endlos weiterlaufen kann.

9.8.2 Schleifen

Bei Schleifen führen Sie im Gegensatz zu Rekursionen einen Programmteil immer auf die gleiche Weise mehrfach aus. Sie können hier zwar auch Kontrollstrukturen programmieren, um ggf. genauer zu steuern, wie die einzelnen Schleifendurchläufe programmiert werden sollen, aber eine pseudo-parallele Ausführung wie bei Rekursionen ist hier nicht möglich.

foreach-Schleifen, for- und while-Schleifen

Wenn Sie mit jedem Eintrag eines Feldes etwas tun wollen, dann brauchen Sie dafür keine Rekursion zu programmieren. Sie können stattdessen eine foreach-Schleife nutzen. Dazu brauchen Sie den Bezeichner des Arrays und Sie vergeben zusätzlich eine Variablenbezeichnung (im Beispiel `$element`). Bei jedem Durchlauf der foreach-Schleife ist der Wert der von `$element` der Wert eines Indexes des Arrays.

Wenn am Anfang nicht feststeht, wie oft die Schleife durchlaufen werden soll, müssen Sie eine while-Schleife nutzen, sonst können Sie auch eine for-Schleife nutzen.

While-Schleifen gibt es in zwei Varianten: Bei der while-Schleife wird er nur so lange ausgeführt, wenn eine bestimmte Bedingung erfüllt ist. Bei der do-while-Schleife wird er dagegen immer einmal ausgeführt, danach erfolgt die Ausführung wie bei einer while-Schleife.

Die Bedingung wird bei einer while-Schleife genau so programmiert, wie Sie das bei if-Conditionals kennen gelernt haben. Allerdings sollte sich das Ergebnis der Bedingung im Laufe der Schleifenausführung ändern, denn sonst haben Sie eine Endlosschleife. Das ist ein Programmteil, in dem das Programm stecken bleibt, bis es vom Nutzer komplett abgebrochen wird.

Bei for-Schleifen müssen drei Dinge festgelegt werden:

- Eine Variable mit einem ganzzahligen Wert, - Eine Abbruchbedingung, bei der diese Variable genutzt wird. - Ein Befehl, der steuert, wie sich der Wert nach je einer Ausführung der Schleife ändert.

Hier für jede der Schleifen ein Beispiel:

```
\$a = 2;
\$b = 10;
$array ist ein Array, das Sie mit Zahlen gefüllt haben.■
```

```
// foreach-Schleife: Das Schlüsselwort as müssen Sie hier immer in dieser
foreach($array as $element) {
/* Im foreach-Schleifenrumpf programmieren Sie schlicht, was mit dem aktu
}
```

```
// Einfache while-Schleife
```



```
while(\$a < \$b){
  \$a += 1;
}

// Einfache do-while-Schleife
do {
  \$a +=1;
} while (\$a < \$b);

// Einfache for-Schleife
for (\$i = 0; \$i < 5; \$i++){
  if (\$array[i] < i) { ... }
  else ...
}
```

Quellcode 4.22: Schleifen

9.9 Überarbeitung: Hinweis bezüglich objektorientierter Softwareentwicklung in PHP

PHP ermöglicht es Ihnen auch, klassenbasierte objektorientierte Programme zu entwickeln. Objektorientierte Softwareentwicklung geht aber weit über die Nutzung zusätzlicher Programmiermöglichkeiten in prozeduralen Sprachen hinaus. Deshalb müssten wir hier im Grunde zunächst eine umfangreiche Einführung in das Konzept durchführen, bevor wir uns ansehen könnten, wie es in PHP umgesetzt wird. Und dafür bleibt in diesem Kurs leider keine Zeit.

Diejenigen von Ihnen, die die Entwicklung von Webanwendungen vertiefen wollen, sollten diesem Thema aber bearbeiten. An dieser Stelle sei noch folgendes gesagt: Werfen Sie jeden Kurs in die Mülltonne, der Ihnen Objektorientierung damit erklärt, dass Sie dort Objekte der realen Welt modellieren. Das ist zwar mit objektorientierter Programmierung möglich, aber wer ernsthaft glaubt, dass es bei objektorientierter Softwareentwicklung darum geht, hat ungefähr so viel Ahnung vom Thema wie ein Maulwurf vom Fliegen. Dem fliegt ja gelegentlich mal was aufs Dach...

9.10 Überarbeitung: Auswertung von Nutzereingaben in HTML durch PHP

Alles, was Sie bislang über PHP gelernt haben erlaubt Ihnen zwar, dynamisch HTML-Code in HTML-Dokumente zu integrieren, aber bislang

können Sie noch keine Nutzereingaben verarbeiten. Damit können Sie also bislang weder einen Log-In, noch eine Registrierung realisieren. Ein Spiel könnten Sie hier realisieren, indem Sie eine Funktion über das onclick-Attribut eines HTML-Containers aufrufen lassen.

Im Folgenden werden wir grundsätzlich mit Daten arbeiten, die Nutzer in Formulare eintragen oder über das Aktivieren von Schaltflächen auslösen. Bei der Einführung in Formulare in HTML haben Sie all die Attribute kurz angesprochen, die dafür nötig sind. Deshalb schauen wir uns zunächst an, welche das sind.

Wichtig: Sie können zwar an beliebigen Stellen eines HTML-Dokuments HTML-Code durch PHP generieren, aber Eingaben von Nutzern können ausschließlich über HTML-Container durchgeführt werden, die entweder in Formularen eingesetzt werden sollen oder in denen das Attribut onclick programmiert werden kann.

Zunächst müssen Sie jeden `<form>`-Container um zwei Attribute erweitern, wie schon zu Beginn dieses Kapitels erläutert: Das `action`-Attribut erhält als Wert die URL der PHP-Datei, die die Auswertung durchführt. Das `method`-Attribut erhält das Wort `POST` als Wert, auch wenn hier alternativ `GET` möglich wäre. Es steuert, wie die Daten vom Client zum Server übertragen werden. Kurz gesagt muss ein Angreifer bei der Nutzung von `POST` etwas mehr Arbeit investieren, um die Datenübertragung abzuhören. Das ist zwar nicht viel und ersetzt auch keine Verschlüsselung, aber es ist ein Schritt in die richtige Richtung.

Außerdem müssen Sie jeder Eingabemöglichkeit (außer Buttons) ein `name`-Attribut zuweisen. Der Wert jedes `name`-Attributs darf innerhalb eines Formulars nur einmal verwendet werden.

Hier ein HTML-Code-Fragment, wenn Sie sich unsicher sein sollten:

```
...  
<form action=registrierung.php method=POST>  
<label ...  
<input id= userid name=userid>  
...
```

Quellcode 4.23: HTML-Formular für die Verarbeitung mit PHP

Im Quellcode oben wird also eine Variable mit der Namen `userid` erzeugt, deren Wert an die Datei `registrierung.php` übertragen wird, nachdem Nutzer die entsprechende Schaltfläche der Webanwendung angewählt haben.

Alle Eingaben, die dabei übertragen werden werden in einem assoziativen Feld mit dem Namen `$_POST` gespeichert. Um also die Variable `userid` in Ihrem PHP-Programm zu verwenden, müssen Sie lediglich den Wert von `$_POST["userid"]` einer Variablen Ihres PHP-Programms zuweisen.

Das ist alles.

9.10.1 Aufgabe

- Programmieren Sie eine Checkbox in den Registrierungs- und Log-In-Formularen Ihrer Webpage. Wenn diese Checkbox aktiviert ist, soll die Eingabe des Passworts am Monitor lesbar sein. Ist sie deaktiviert, dann soll das Passwort nicht lesbar sein.

9.11 Überarbeitung: Abschluss

Sie haben jetzt alles kennen gelernt, was Sie benötigen, um eine dynamische Webpage zu programmieren. Wenn Sie beispielsweise ein Programm entwickeln wollen, um einen Schaltplan für die Laborversuche in Elektrotechnik zu entwerfen, wissen Sie jetzt alles, was dazu nötig ist. Wenn Sie denken, dass das ein langweiliges Programm wird, kann ich Ihnen leider nur zustimmen, aber dafür ist die Fehleranfälligkeit niedrig.

Es gibt noch einige Themen, die wir in diesem Kapitel nicht besprochen haben, die Sie sich aber ansehen sollten, wenn Sie langfristig als Webentwickler mit PHP arbeiten wollen. Neben der klassenbasierten objektorientierten Programmierung wären da die folgenden Dinge zu nennen:

- Welche Datentypen kennt PHP und wie können Sie den Datentyp einer Variablen erfahren? - Wie werden Exponentialzahlen in PHP programmiert? - Was genau ist der Unterschied zwischen POST und GET und wie müssen Sie Werte in einem PHP-Programm verarbeiten, die per GET übertragen wurden? - Welche Reihenfolge gilt bei Operatoren? (Bsp.: Wird `==` vor `;` ausgewertet oder umgekehrt?) - Was sind Pseudo-Zufallszahlen und wie können Sie sie für ein PHP-Programm nutzen? - Was bewirken `break` und `continue` bei Schleifen? - Was hat es mit `include` und `require` bzw. `include_once` und `require_once` auf sich? - Was ist der Unterschied zwischen Kopie und Referenz bei einem Funktionsaufruf? - Wie können Sie Funktionen programmieren, die eine dynamische Anzahl von Parametern annehmen können? (Auch bekannt als variable Parameterliste) - Welche Funktionen bietet PHP an, um z.B. mathematische Funktionen zu realisieren? - Mit welchen Funktionen können Sie Strings z.B. auf eine bestimmte Länge kürzen? - Was hat es mit dem Gültigkeitsbereich einer Variablen auf sich? - Was ist

eine Generatorfunktion? - Welche Fehler können Sie durch die sogenannten Exceptions lösen? Und warum ist das wichtig? - Wie können Sie Verschlüsselungsverfahren einprogrammieren? - Wie können Sie Nutzereingaben ohne eine Datenbank oder den Einsatz von Cookies und ähnlichem so lange speichern, wie der Nutzer sie braucht?

Kapitel 10

Langfristige Datenspeicherung mit MySQL 5.6

Liegt bereits vor, muss „nur“ noch in LaTeX konvertiert werden.

Teil II

Fortsetzung für Studierende der Informatik

Kapitel 11

Einführung in den Aufbau von ARMv6-Prozessoren am Beispiel eines Cortex-M0-Prozessors

Wichtig (1): Dieses Kapitel ist wichtig, auch wenn hier noch nicht eine Zeile Programmcode auftaucht: Erst wenn Sie es durchgearbeitet haben, haben Sie eine Chance zu verstehen, was alles im Hintergrund passieren muss, damit die Dinge in PHP so funktionieren, wie Sie das dort kennen gelernt haben. Wenn Sie diese Kenntnisse nicht haben, werden Sie bei komplexen Problemen wie der nebenläufigen Programmierung (Programmierung von Systemen mit mehreren Prozessorkernen) bekommen.

Wichtig (2): Dieses Kapitel ist zurzeit lediglich ein Einstieg in die Grundlagen von Mikroprozessoren, konkret die Programmierung von ARM-Prozessoren der Baureihe Cortex-M0. Die tatsächliche Programmierung ist noch nicht Teil dieses Kapitels. Ggf. wird es hier später noch eine Ergänzung geben, zurzeit ist es lediglich eine Unterstützung, wenn Sie Probleme beim Verständnis des Aufbaus dieses Prozessortyps haben.

In den letzten Kapiteln haben Sie eine erste Einführung in die Programmierung von Anwendungen mit Hilfe einer Kombination aus vier Programmiersprachen erlernt. Je nachdem, in welchen Bereich der Programmierung als (Medien-)Informatik Sie sich später vertiefen wollen benötigen Sie aber zusätzlich ein detailliertes Verständnis aus einem oder mehreren der Bereiche, die wir im letzten Kapitel kurz angeschnitten haben oder die Ihnen dort von einer der Programmiersprachen abgenommen wurden.

In diesem Kapitel können Sie sowohl einige Grundlagen der maschinenna-

KAPITEL 11. EINFÜHRUNG IN DEN AUFBAU VON ARMV6-PROZESSOREN AM BEISPIEL EINER

hen als auch die daraus resultierenden Grundlagen der imperativen Programmierung erlernen. Auch wenn Sie an der maschinennahen Programmierung nicht interessiert sind, benötigen Sie die verschiedenen Informationen dieses Kapitels, da Sie nur durch diese verstehen werden, warum bestimmte Dinge in der imperativen Programmierung genauso passieren, wie das der Fall ist.

Bitte beachten Sie jedoch, dass dieses Kapitel nur eine Ergänzung zu einer eigentlichen Einführung in die Programmierung von Mikroprozessoren sein kann. Ursprünglich hatte ich geplant, nach der Einführung der Konzepte eine tatsächliche Einführung in die maschinennahe Programmierung zu erstellen. Interessierte Studierende hätten sich so in diesen Bereich vertiefen können. Aus zeitlichen Gründen habe ich diesen Teil vorerst gestrichen.

Dieses Kapitel basiert auf den folgenden Quellen:

- Das Standardwerk *The definitive Guide to the Cortex-M0* von Joseph Yiu, das Sie erwerben sollten, wenn Sie mit der Softwareentwicklung mit ARM-Prozessoren des Typs Cortex-M0 weiter machen wollen. Prüfen Sie jedoch, ob es inzwischen eine aktuellere Version des Buches gibt; die zurzeit vorliegende Version wurde 2011 veröffentlicht und beinhaltet dementsprechend teilweise veraltete Inhalte.
- Das Skript zur Veranstaltung Informatik 3 – Informatik und Elektrotechnik des Studiengangs Media Systems an der HAW Hamburg von Michael Berens und Jens Ginzler. Diese erhalten Sie ggf. zusammen mit dem Zugriff auf die Veranstaltungsinhalte im Rahmen des seminaristischen Unterrichts.
- Dem LPC-P1114 development board Users Manual von 2011. Da wir die Version des Boards von 2011 verwenden ist dieses Dokument auf aktuellem Stand.
- Dem Cortex-M0 Devices Generic User Guide ARM DUI 0497A (ID 112 109)
- Der UM 10398 Dokumentation. Hierin finden Sie detaillierte Informationen, welche Belegung eines Registers beim Entwicklerboard welche Aufgabe erfüllt.
- Die Dokumentation zum CMSIS, der Sprachbibliothek zur Cortex-M0-Programmierung in C.

Bevor wir zur eigentlichen Programmierung kommen, müssen Sie zunächst einen Überblick darüber bekommen, was Sie da eigentlich im Detail programmieren. Das ist nirgends so wichtig wie bei der maschinennahen Programmierung, weil Sie hier durchgehend die Werte der Komponenten des Prozessors bzw. der Peripherie beeinflussen. Und wie sollen Sie das tun, wenn Sie genau diese Komponenten gar nicht kennen oder wissen, was sie tun?

Wenn Sie nicht an der Veranstaltung „Informatik 3“ teilnehmen, in der die

Programmierung des Cortex-M0 behandelt wird, z.B. weil Sie nicht Media Systems sondern Medientechnik studieren, dann können Sie natürlich gerne im Labor nachfragen, ob Sie dort ein Entwicklungssystem nutzen können. Alternativ können Sie jedoch auch ein günstiges Entwicklerboard mit den nötigen Anschlüssen erwerben. Zum Teil gibt es beispielsweise bei IAR Sonderangebote, bei denen Sie ein günstiges Starter-Kit erhalten. Für höchstens 40 Euro (ggf. plus Versandkosten) sollten Sie dort alles finden, was Sie für den Einstieg benötigen. Wenn Sie ein System erwerben wollen, das dem in unseren Laboren verwendeten ähnelt, dann suchen Sie nach einem NXP LPC1114. Die Modelle mit den Bezeichnungen LPC11x14 sind nahezu baugleich, also genauso geeignet; hier steht anstelle des x ein anderer Buchstabe, der für eine zusätzlich verbaute Komponente steht. Das LPC11U14 beinhaltet beispielsweise die Unterstützung von USB 2.0 Schnittstellen.

Wenn Sie bereit sind, wesentlich tiefer in die Tasche zu greifen, möchte ich Sie auf den Hersteller Keil hinweisen; die Produkte dieses Herstellers werden in der Wirtschaft häufig eingesetzt, sind aber nichts für den kleinen Geldbeutel. So müssen Sie für eine gute Debugging Unit (die Schnittstelle, die nötig ist, um den Computer und ein Entwicklungsboard zu verbinden) dort zum Teil deutlich mehr als 1.000 € ausgeben. Gerade wenn Sie zunächst nur in diesen Bereich hereinschnuppern wollen ist das eigentlich zu viel. Aber wenn Sie Ihre berufliche bzw. forschende Zukunft im Bereich der Mikroprozessoren sehen, sollten Sie später darüber nachdenken.

11.1 Mikroprozessoren und eingebettete Systeme

Die Bezeichnung Mikroprozessor ergibt sich schlicht und ergreifend daraus, dass die ersten Mikroprozessoren in einer Bauweise angefertigt wurden, die in Mikrometern gemessen wurde (heute in Nanometern), und dass Sie Prozesse abarbeiten. So weit, so simpel.

In der elektrotechnischen Welt ist es üblich das Symbol μ (ausgesprochen Mü), das Sie aus dem Bereich der Maßeinheiten kennen, auch außerhalb von Maßeinheiten als Ersetzung für das Präfix Mikro- zu verwenden. Beispielsweise wird die IDE Mikrovision von Keil üblicherweise als μ Vision beschriftet.

Wann immer Sie Kürzel wie CPU (Central Processing Unit), GPU (Graphical Processing Unit), APU (Accelerated Processing Unit) usw. sehen, haben Sie es mit Mikroprozessoren zu tun. Und ob es sich dabei nun um eine CPU, eine GPU oder eine APU handelt, hat vorrangig mit dem konkreten

Aufbau des jeweiligen Mikroprozessors und weniger mit dem geplanten Einsatzbereich zu tun. Beispielsweise werden GPUs auch eingesetzt, um die virtuelle Währung Bitcoin zu generieren. Mit Computergrafik hat das nichts mehr zu tun. Im Rahmen dieses Kurses brauchen Sie nicht zu wissen, was eine CPU, eine GPU usw. voneinander unterscheidet.

Dementsprechend macht es in diesem Bezug auch keinen Unterschied, ob wir nun über Prozessoren von Intel, AMD, ARM, Motorola oder welchem Hersteller auch immer reden: Es sind alles Mikroprozessoren und im Kern basieren sie alle auf denselben Prinzipien. Allerdings gibt es mittlerweile eine Vielzahl an zusätzlichen Komponenten, durch die sie sich dann wiederum deutlich voneinander unterscheiden. Aber auch ohne diese zusätzlichen Komponenten unterscheiden sich die Bezeichnungen von Komponenten bei verschiedenen Prozessoren desselben Herstellers häufig. Diese Komponenten eines Mikroprozessors zu kennen und ihre Programmierung zu beherrschen zeichnet professionelle EntwicklerInnen aus.

Wenn die Rede von Mikroprozessoren ist, dann wird häufig auch von eingebetteten Systemen (embedded system) gesprochen. Im Gegensatz zu einem Rechner, den Sie als solchen leicht erkennen können, sind eingebettete Systeme stets so in andere Systeme eingebettet, dass sie als eigenständige Computer nicht erkennbar sind. Ihre Programmierung verläuft aber genauso wie die von einem handelsüblichen Computer. Es gibt eingebettete Systeme seit weit mehr als 20 Jahren.

Wenn wir über die Programmierung eines Mikroprozessors reden, nutzen wir gelegentlich auch die Bezeichnung Bare Metal Target. Damit ist ein Mikroprozessor gemeint, auf dem kein Betriebssystem aktiv ist. In diesem Bereich werden Betriebssysteme als embedded OS bezeichnet, um sie als Betriebssystem für eingebettete Systeme von anderen Betriebssystemen abzugrenzen.

Kontrolle

Sie wissen jetzt, was ein Mikroprozessor ist.

11.2 SoC, Mikrocontroller und die Umgebung von Mikroprozessoren

Da es wenig Sinn macht, einen Mikroprozessor an den Strom anzuschließen und ihn ansonsten nur herumstehen zu lassen, gibt es immer eine Umgebung, an die er angeschlossen wird. Um nun mit dieser Umgebung interagieren zu können, benötigt der Mikroprozessor aber noch einige zum

Teil optionale Komponenten:

- Ein Timer dient dazu, mit einer gewissen Genauigkeit zeitliche Abläufe zu koordinieren. Gerade die Formulierung „mit einer gewissen Genauigkeit“ ist hier wichtig: Innerhalb eines Rechners können wir mit einer Genauigkeit im Nanosekundenbereich Daten austauschen. Aber bereits wenn wir zwei Rechner auf kurzer Distanz miteinander verbinden, dann sinkt diese zeitliche Genauigkeit. Und je komplexer die Verknüpfung (das heißt je mehr Rechner verbunden sind) bzw. je größer die Distanz, desto unsicherer ist die Genauigkeit.

Wenn Sie sich beispielsweise mit den Grundlagen des Internet ein wenig auskennen, dann wissen Sie, dass es etwas gibt, das als Latenz bezeichnet wird. Das ist der zeitliche Versatz zwischen Versand und Empfang, der bei einer Datenübertragung auftritt. Diese Latenz ist aber kein konstanter Wert, sondern er unterliegt einer Schwankung, die durchaus mehrere Sekunden betragen kann. Und Sie kennen wenigstens einen Fall, in denen ungenaue Zeitabstimmung die Ursache für ein Problem ist: Denken Sie an das letzte Mal, als Sie einen Film im Netz gesehen haben, bei dem die Tonspur und die Bildspur nicht synchron liefen.

- I/O (kurz für Input/Output, teilweise auch nur IO) bezeichnet alle Arten von Komponenten, die es einem Mikroprozessor ermöglichen, Eingaben anzunehmen und Ergebnisse auszugeben. Wir reden hier aber nicht über Displays, sondern über die Schnittstellen, über die ein Mikroprozessor Daten weiterleitet bzw. empfangen kann. Displays, Lautsprecher, Mikrofone, Tastaturen, usw. usw. haben wiederum eigenen Interfaces, die entweder direkt oder mittelbar an die I/O-Interfaces von Prozessoren angeschlossen werden können.

- Interrupt-Controller sind essentiell: Sie signalisieren dem Mikroprozessor, wenn er einen bestimmte Prozess pausieren und zunächst einen anderen ausführen soll. Das kann beispielsweise dann der Fall sein, wenn ein Nutzer am Rechner eine Taste drückt. Wenn Sie diese Funktion verstanden und das ein oder andere Programm entwickelt haben, das mit Interrupts arbeitet, dann werden Sie die Gründe für eine Vielzahl von Fällen verstehen, in denen der naive User ungefähr die folgenden Worte von sich gibt: „Warum macht die sch... Kiste nicht, was ich will?!“

Hinweis bezüglich Exceptions: Wenn Sie schon in höheren Programmiersprachen programmiert haben, dann haben Sie die sogenannten Exceptions kennen gelernt. Für den Moment merken Sie sich bitte, dass Exceptions im Rahmen der maschinennahen Programmierung ein Spezialfall von Interrupts sind. Teilweise werden Exceptions und Interrupts hier auch als

Synonyme verwendet.

- ROM und RAM sind zwei Bezeichnungen, die für Speicher verwendet werden, in dem sich Programme und Daten befinden. Für den Augenblick genügt es, wenn Sie sich merken, dass der Inhalt vom RAM gelöscht wird, wenn der Strom ausfällt bzw. abgeschaltet wird. ROM-Speicher dagegen behält seinen Inhalt auch ohne Stromversorgung dauerhaft. Dafür können die Daten beim ROM nicht geändert werden. Es gibt außerdem noch eine dritte Art von Speicher, bei dem der Inhalt auch ohne Stromfluss erhalten bleibt, der aber jederzeit überschrieben werden kann. Diese Art von Speicher nennt man Flash oder EPROM (kurz für Erasable Programmable ROM). Je nach Bauart ist nur ein vollständiges Löschen und neu Speichern von Daten möglich.

Wenn nun ein Prozessor mit einzelnen dieser Komponenten zu einer Einheit verbunden wird, spricht man vom Mikrocontroller. Eine Abkürzung für Mikrocontroller lautet MCU (Micro Controller Unit).

Darüber hinaus gibt es noch größere Verbünde von elektrotechnischen Komponenten, die dann als SoC (kurz für System-On-Chip) bezeichnet werden. Wie so oft ist der zentrale Grund für diese immer stärkere Vereinigung von immer mehr Komponenten in immer weniger Bauteilen schlicht der Kostenfaktor. SoCs sind nicht Teil dieser Veranstaltung, da Sie zunächst die Programmierung von Mikrocontrollern und FPGAs beherrschen sollten, bevor Sie sich mit diesen Geräten auseinander setzen können.

11.2.1 Das Olimex LPC1114 Entwicklungsboard

Im Falle des Entwicklungsboards, das in unseren Laboren zum Einsatz kommt, haben Sie einige Komponenten, die bislang nicht aufgeführt wurden, die Sie aber im Laufe der Labore programmieren werden. Diese sind zusammen mit den zugehörigen Befehlen und möglichen Belegungen im UM 10398 User Manual aufgeführt, das Sie dementsprechend für diesen Kurs benötigen:

(In einigen Fällen werden Sie die Bezeichnung der MCU benötigen. Hierbei handelt es sich wie geschrieben um den LPC 1114 des Herstellers NXP.)

- Der SPI-Controller (kurz für Serielles Peripherie Interface). Dieser Controller steuert die Datenübertragung vom Prozessorbus zu den Interfaces der verschiedenen externen Komponenten, die Sie an das Entwicklerboard anschließen können. Ihrer Phantasie sind dabei kaum Grenzen gesetzt, weshalb die letzten Wochen dieser Veranstaltung darin bestehen, dass Sie sich

eine (oder mehrere Komponenten) auswählen, diese mit dem Entwicklerboard verbinden und dann den Prozessor entsprechend programmieren. Teilweise reden wir hier auch von der Programmierung der SPI-Schnittstelle. ■

- Ein Oszillator, der für zeitabhängige Anwendungen eine deutlich höhere Präzision bietet als der im Cortex-M0 enthaltene Timer oder der ebenfalls auf dem Entwicklerboard enthaltene WDT (kurz für Watch Dog Timer).

11.3 Grundlagen des Cortex-M0

Einige Kenndaten des Cortex-M0 müssen Sie kennen, um Programmierfehler zu vermeiden. Außerdem gibt es einige Begriffe, die speziell von ARM in den offiziellen Referenzen verwendet werden. Wenn Sie sie dann nicht kennen, dürften sie Sie verwirren, weshalb Sie hier mit aufgenommen wurden:

- Es handelt sich um einen 32-Bit-Prozessor. Das bedeutet, der Prozessor kann in jedem Arbeitsschritt ausschließlich Werte zwischen 0 und 232-1 verwenden. Damit ist der Hauptspeicher auf rund 4 GB begrenzt.

Anm.: Im Falle der Evaluationsversion des Keil MDK, die wir für die Softwareentwicklung verwenden stehen 32 KB zur Verfügung. Da der LPC1114 aber ohnehin nur über 32 KB verfügt, ist das effektiv keine relevante Einschränkung.

- Sie müssen hexadezimal rechnen. $0x10 - 4$ ist also nicht $0x06$!
- Sie müssen von Angaben wie „setze das zweite und fünfte Bit“ auf die entsprechende hexadezimale Zahl (hier: $0xA$) umrechnen können.
- WICHTIG! Adressen entsprechen jeweils Einheiten von 8 Bit. Da der Cortex-M0 jedoch bei jedem Schritt 32 Bit einliest oder ausgibt, müssen Sie häufig mit dem Faktor 4 multiplizieren, um beispielsweise eine „benutzbare“ Adresse zu erhalten.

Beispiel: Sie wollen einen Wert im Speicher des Rechners ablegen und wissen, dass bis zur Adresse $0xFD$ Daten im Speicher liegen. Dann dürfen Sie erst ab der Adresse $0x100$ Daten ablegen, da $0xFE$ und $0xFF$ nicht durch 4 teilbar sind.

- Dennoch handelt es sich bei den Befehlen, die auf dem Cortex-M0 ausgeführt werden können, fast ausschließlich um 16-Bit lange Befehle. Dafür

gibt es bei ARM-Prozessoren die Bezeichnung Thumb State. Wenn Sie es mit vollwertigen „32-Bit-Befehlen“ zu tun haben, dann werden diese bei ARM Prozessoren als das ARM Instruction Set bezeichnet.

- Der Prozessor arbeitet mit einer sogenannten Load-Store-Architektur. Das bedeutet in letzter Konsequenz, dass Sie Daten aus dem Speicher immer in die sogenannten Register laden müssen, um etwas mit Ihnen zu tun. Register sind eine spezielle Form von Speicher, die genauso viele Bit speichern können, wie es der Bittigkeit des Prozessors entspricht. Sie sind Teil des eigentlichen Prozessors.

Anm.: Diese Beschränkung macht den Einstieg in die Programmierung etwas leichter, denn so können Sie leichter erkennen, ob eine Zahl bei einer Operation einen Wert oder die Adresse einer Speicherstelle ist.

- Am Rande sei noch bemerkt, dass ARM zwischen zwei Stati und zwei Modi unterscheidet: Im Thread Mode führt der Prozessor beliebigen Programmcode aus, im Handler Mode werden bestimmte Arten von „Unterbrechungen“ behandelt. Diese Modi spiegeln verschiedene Fälle wieder, mit denen Sie es bei der Programmierung zu tun bekommen. Ebenfalls nicht direkt sichtbar ist der Debug State. Den bekommen sie automatisch dann zu Gesicht, wenn Sie den Debugger benutzen. Genau wie der Thumb State, der die beiden oben genannten Modi (Thread/Handler) zusammenfasst, ist er wie gesagt eher eine virtuelle Abgrenzung, die Sie fürs erste kaum wahrnehmen werden.

11.4 Die Register des Cortex-M0

Wie eingangs erwähnt können Sie nur die Daten verändern, die Sie aus dem Speicher in die Register geladen haben. Es gibt verschiedene Arten von Registern, von denen Sie die folgenden kennen müssen, um mit der Programmierung zu beginnen:

11.4.1 R0 bis R12 – Die Allzweckregister

Zunächst wären da die 13 GPRs (kurz für General Purpose Registers), die der Einfachheit halber mit R0 bis R12 angesprochen bzw. programmiert werden. Diese Register können Sie als ProgrammiererIn so mit Inhalten füllen, so wie es Ihnen passt. Wichtig: In aller Regel werden Sie im Thumb-State arbeiten. In diesem können sie von den Registern 0 bis 12 ausschließlich die Register R0 bis R7 nutzen. Das sollte aber in aller Regel genügen.

Mehr gibt es zu den Allzweckregistern nicht zu sagen.

Hier ein Beispiel in Pseudocode, um die Nutzung der Register zu verdeutlichen:

- Lade den Wert von Speicheradresse 0x1FC in R0.
- Lade den Wert von Speicheradresse 0x200 in R1.
- Vergleiche den Wert in R0 mit dem in R1.
- Ist R1 größer oder gleich R0, dann speichere den Wert von R1 in 0x1FC.
- Vergleiche den Wert in R0 mit dem in R1.
- Ist R0 kleiner als R1, dann speichere den Wert von R0 in 0x200.

Kontrolle

Dieses kleine Programm in Pseudocode ist der Kern eines größeren Algorithmus, in dem es vielfach wiederholt wird. Was ist die Aufgabe, die dieser Algorithmus löst?

11.4.2 Einschub: LSB = 1 für Thumb State

Wie Sie wissen, können (Speicher-)adressen, auf die beim Cortex-M0-Prozessor verwiesen wird nur gerade Zahlen (da durch 4 teilbar) sein. Also darf das LSB (kurz für Least Significant Bit) einer verwendbaren Adresse nur eine Null sein.

Es gibt jedoch eine Besonderheit: Wenn Sie im Thumb State programmieren, dann muss (!) bei einigen Befehlen das LSB einer Adresse auf 1 gesetzt werden. Das entspricht natürlich einer Erhöhung des Wertes um 1. Bei der Ausführung der Operation prüft der Prozessor also zunächst, ob das LSB gleich 1 ist. Ist das der Fall, zieht er 1 von der übergebenen Adresse ab und führt dann die Operation als Thumb State Operation also 16-bittig mit der jetzt richtigen Adresse aus. Bei welchen Befehlen das der Fall ist, können Sie sich nicht durch logisches Denken erschließen, sondern ausschließlich durchs Lesen der entsprechenden Dokumentation erfahren.

11.4.3 SP – Der Stack Pointer (alternativ R13)

Es kommt im Laufe eines Programms immer wieder vor, dass Daten vorübergehend zwischengespeichert werden müssen, die in einer bestimmten Reihenfolge gespeichert werden sollen. Das ist beispielsweise dann der Fall, wenn ein Prozess unterbrochen aber später fortgesetzt werden soll.

In diesem Fall wird dann die Adresse, an der der Prozess unterbrochen wurde in einer Datenstruktur abgelegt, die als Stack bezeichnet wird.

KAPITEL 11. EINFÜHRUNG IN DEN AUFBAU VON ARMV6-PROZESSOREN AM BEISPIEL EINER

Für den Fall, dass Sie noch keine Veranstaltung zu Algorithmen und Datenstrukturen gehört haben, hier eine kurze Einführung in das Thema: Informatiker sind Meister der Abstraktion und Strukturierung. Nicht missverstehen: Abstraktion und Strukturierung sind zentrale Werkzeuge aller naturwissenschaftlichen Disziplinen und Ingenieurwissenschaften, aber im Gegensatz zu Elektrotechnikern beginnen Informatiker und Mathematiker ab dem ersten Semester mit Abstraktionen und Strukturen. Und so dreht sich dann gleich zu Beginn die Frage darum, wie man denn in bestimmten Situationen Daten am sinnvollsten aufbewahren kann und wie man am effizientesten auf sie zugreifen kann. Lösungen für die sinnvolle Aufbewahrung nennt man Datenstrukturen, Lösungen für effiziente Zugriffsweisen u.a. nennt man Algorithmen. Und ein Stack ist eine Datenstruktur. Wobei beide Begriffe auch dann verwendet werden, wenn sie keine effizienten Ansätze umsetzen. Aber mit solchen Fällen sollten Sie sich nur auseinandersetzen, wenn Sie wissen wollen, warum ein anderer Ansatz sinnvoller ist.

Einen Stack kann man sich als simplen Stapel vorstellen, bei dem Elemente bzw. Daten nur in der umgekehrten Reihenfolgen abgenommen werden können, in der sie aufgelegt wurden. Aber wohlgemerkt: Wir reden hier über Informatik, die Vorstellung eines Stapels ist also nur ein Modell, das der Anschaulichkeit dient.

Konkret wird ein Stack wie folgt umgesetzt: Nachdem man den Speicher so behandelt, als wenn er (bei einem 32-Bit-Prozessor) aus Einheiten mit 32 Bit Länge bestehen würde und man diesen Einheiten fortlaufenden Nummern gegeben hat, legt man willkürlich eine dieser Adressen als Startadresse eines neuen Stack fest. Um zu speichern, wo das aktuellste Element des Stack sich befindet, benötigt man nun einen Stack Pointer. Das ist schlicht ein Speicherbereich bzw. ein Register, dessen einzige Aufgabe darin besteht, zu speichern, wo sich das zuletzt gestackte Element befindet. Beim Cortex-M0 gibt es dafür das Register SP. Sobald ein Wert im Stack gespeichert oder von dort entfernt wird, muss der Wert des Stack Pointer angepasst werden. Das passierte beim Cortex-M0 bei den Branch-Befehlen im Hintergrund, sodass Sie es dann und nur dann nicht selbst machen müssen.

Jetzt kommt die erste Hürde für Einsteiger: Es gibt keinerlei generelle Vorschrift dafür, wie der Stack Pointer initialisiert oder geändert werden darf. Was nun beschrieben wird, dürfen Sie also ausschließlich bei Cortex-M0-Prozessoren voraussetzen; bei allen anderen Prozessoren müssen Sie prüfen, wie der Ablauf funktioniert, bzw. ob es dort überhaupt eine entsprechende Konvention gibt. Denn selbst das dürfen Sie nicht voraussetzen.

Greifen Sie jetzt am besten zu Stift und Papier und versuchen Sie sich das

KAPITEL 11. EINFÜHRUNG IN DEN AUFBAU VON ARMV6-PROZESSOREN AM BEISPIEL EINER

folgende Beispiel mittels einer Skizze zu veranschaulichen.

- Die erste Zeile Ihres Programms weist dem Speicher 0x0 bis 0x3 des Cortex-M0 die Adresse zu, auf die der SP zu Beginn verweisen soll.

Beispiel: Angenommen, die Adresse, auf die der Stack Pointer zu Beginn des Programms verweisen soll lautet 0 x 2EF5 2C04. Dann weist Ihr Programm

der Adresse 0x0 den Wert 0x2C zu, der Adresse 0x1 den Wert 0x4, der Adresse 0x2 den Wert 0x2E und der Adresse 0x3 den Wert 0xF5.

Im Debugger würde das so aussehen:

```
0x00000000 2C04 DCW 0x2C04 0x00000002 2EF5 DCW 0x2EF5
```

DCW ist der Befehl, der einem Speicherbereich einen Wert zuordnet. Und auf der linken Seite steht lediglich vor den Speicheradressen ein 0x, weil Werte im Speicher grundsätzlich als Hexadezimalwerte interpretiert werden.

Wichtig: Sehen Sie bitte genau hin: Sie programmieren die Adresse 0x2EF52C04 in zwei Schritten, bei denen sie zuerst die hinteren vier Ziffern eingeben und erst danach die vier vorderen Zahlen! Das gilt für alle Fälle, in denen Sie eine 32-bittige Zahl zuweisen.

- Im Gegensatz zum anschaulichen Beispiel wächst der Stack beim Cortex-M0 nicht von unten nach oben, sondern von größerer Adresse zu kleinerer Adresse. In anderen Worten: Die Adressen des Stack werden mit jedem gespeicherten Wert um 0x4 kleiner!
- Von der Logik her verweist der SP jeweils auf die Adresse, an der das zuletzt gespeicherte Element liegt. Sie weisen hier also beim Programmstart nicht die erste Adresse des Stack zu, sondern die um 0x4 größere Adresse.

Zur Erinnerung: Der Cortex-M0 liest bei jedem Arbeitsschritt 32 Bit bzw. 4 Byte und damit 4 Adressen am Stück ein. Und deshalb haben wir es gerade mit dem Faktor 4 bei der Adressierung zu tun.

Beispiel: Sie wollen den ersten Wert an der ersten möglichen Adresse unterhalb 0x200 speichern. Daraus ergibt sich die Adresse 0x1FC. Dennoch legen Sie als Initialwert für den SP 0x200 fest, da der Wert des SP auf die Adresse verweist, an der zuletzt ein Element gespeichert wurde. Und das gilt eben auch, wenn im Stack noch kein Element gespeichert wurde.

In anderen Worten: Der SP wird nicht (!) mit der Adresse initialisiert, in der das erste Element gespeichert werden soll, denn bei der Speicherung eines Elements im Stack wird zuerst der SP auf die nächste mögliche Adresse gesetzt und dann wird an der dortigen (!) Adresse das Element gespeichert. Diese ist die nächste um 4 bzw. 0x4 kleinere Adresse als diejenige, auf die der SP zuletzt verwies.

Wenn wir den Initialwert des Beispiels, also die Adresse 0x200 verwenden, dann liegt das erste Element, das wir auf dem Stack ablegen unter den Adressen 0x1FC, 0x1FD, 0x1FE und 0x1FF. Der SP verweist anschließend auf die Adresse 0x1FC. Legen wir ein weiteres Element auf dem Stack ab, so liegt dieses (selbst wenn es nur eine Länge von einem Bit hat) unter den Adressen 0x1FB, 0x1FA, 0x1F9, 0x1F8 und der SP beinhaltet den Wert 0x1F8.

Für die Zugriffe auf den Stack gibt es wieder festgelegte Bezeichnungen: Fügen wir ein Element zum Stack hinzu, so nennen wir das PUSH, entfernen wir das zuletzt hinzugefügte, bezeichnen wir das als PULL. Außerdem gibt es noch den Befehl PEEK. Damit wird ein Zugriff auf das oberste Element eines Stack bezeichnet, das dabei aber nicht entfernt wird. In aller Regel kommen aber nur Push und Pull zum Einsatz.

Ob der Prozessor bei einem Pull tatsächlich den Wert an der Adresse löscht (wie auch immer ein Löschen aussehen soll), von der gepullt wurde, das steht nicht fest. Somit könnte ein Pull im Grunde wie ein Peek sein, bei dem lediglich der SP verschoben wurde.

Aufgabe:

Warum gibt es kein Löschen bei der maschinennahen Programmierung?

Kontrolle

Führen sie die nachfolgenden Schritte mit Stift und Papier durch und notieren Sie in Pseudocode: Sie wollen den Stack initialisieren und ihn für eigene Operationen verwenden.

(i) Initialisieren Sie den SP mit einer Startadresse, die beim Cortex-M0 existiert und größer als 0xFFFF + 2 ist. (ii) Pushen Sie den Wert 17 auf den Stack. (iii) Unter welcher Adresse wurde der Wert gespeichert? (Hier gibt es mehrere mögliche Antworten, die alle vier richtig sind.) (iv) Welcher Wert steht jetzt an dieser Adresse? (v) Pushen Sie den Wert 32 auf den Stack. (vi) Auf

welche Adresse verweist der Stack Pointer jetzt? (vii) Welcher Wert steht dort? (viii) An welcher Adresse würde ein Element abgelegt, wenn Sie es jetzt pushen würden?

Hinweis: Lassen Sie sich nicht irritieren: Wenn Sie die Funktionsweise des Stack, des Stack Pointers und die Nutzung des Speichers beim Cortex-M0 verstanden haben, ist das eine ganz simple Übung. Teilweise sind die Antworten auf Fragen hier tatsächlich identisch. Es handelt sich aber nicht um einen „Idiotentest“.

11.4.4 LR – Das Link Register (alternativ R14)

Wenn ein Assemblerprogramm nach Abarbeitung einer Programmzeile nicht in die nächste Zeile, sondern in eine beliebige andere Zeile des Programms springt, sprechen wir von einem Branch, bzw. vom branchen. Alternativ ist die Rede vom Aufruf einer Subroutine. Eine solche Subroutine kann beispielsweise ein Programmteil sein, der an verschiedenen Stellen im Programm benötigt wird. Um ihn dann nicht mehrfach programmieren zu müssen, wird er in einem anderen Bereich des Speichers abgelegt und mit einer Zeile beendet, die sinngemäß bedeutet: Branche (zurück) auf die Adresse, die im LR gespeichert ist. Umgangssprachlich würde man sagen: Setze das Programm an der Stelle fort, von der aus du in diese Subroutine gesprungen bist.

Hier der Assembler-Befehl, um am Ende einer Subroutine zurück zu der Stelle im Programm zu springen, an der die Ausführung zuletzt war:

BX LR

Damit wären wir auch beim Sinn dieses Registers: Das Link Register speichert beim branchen die Adresse, an der das Programm sich gerade befindet, bzw. an der es nach Ende der Subroutine fortgesetzt werden soll.

Kontrolle:

Warum müssen Sie nach bisherigem Kenntnisstand davon ausgehen, dass ein Branch aus einer Subroutine zu einem schweren Fehler im Programmablauf führt bzw. wodurch würde der verursacht?

11.4.5 PC – Der Program Counter (alternativ R15)

In diesem Register speichert der Prozessor diejenige Adresse, an der sich die Programmzeile befindet, die als nächstes ausgeführt wird. Um also die

aktuelle Programmzeile zu finden, müssen Sie von dieser Adresse 4 abziehen.

11.4.6 xPSR – Das kombinierte Statusregister

Das xPSR nutzen Sie, wenn Sie den Debugger nutzen, um bestimmte Informationen über die Zustände zu erhalten, die durch eine Operation erreicht wurden. Das geht lediglich beim Debuggen, weil die Werte sich bei jeder ausgeführten Operation ändern. Das xPSR ist kein eigenständiges Register, lässt sich aber bei der Programmierung wie ein Register verwenden und stellt die für uns relevanten Inhalte der drei folgenden „echten“ Register gemeinsam dar:

APSR (kurz für Application Program Status Register), für das auch die Bezeichnung ALU Register üblich ist, IPSR (kurz für Interrupt Program Status Register) und EPSR (kurz für Execution Program Status Register).

Die drei PSR zeigen über die folgenden Bits die in der nachfolgenden Tabelle eingetragenen Hinweise dar. Bitte beachten Sie, dass hier die tatsächliche Nummer jedes Bits aufgeführt wird (beginnend bei 0) und nicht die Anzahl (beginnend bei 1).

Hinweis: Die PSR beinhalten diese Werte nach der Ausführung einer Operation; sobald eine weitere Operation ausgeführt wurde, werden die PSR mit neuen Daten überschrieben. Um also zu sehen, welche Bits im (x)PSR gesetzt werden, wenn eine bestimmte Operation ausgeführt wurde, müssen Sie im Debugger eine Markierung an der entsprechenden Stelle des Quellcode setzen.

Ein typischer Anfängerfehler besteht darin, eine Markierung an der falschen Stelle im Quellcode zu setzen bzw. bei einem „Programmfehler“ die PSR zur falschen Programmzeile zu untersuchen.

Bit Nr. Abkürzung Bedeutung Originäres Register 31 N Negative APSR 30 Z Zero 29 C Carry 28 V Overflow 24 Keine 0 = ARM, 1 = Thumb EPSR 5...0 Keine Exception Type IPSR

Für alle folgenden Angaben gilt wie immer bei der maschinennahen Programmierung: Ob etwas Sinn macht, ist für den Prozessor irrelevant; er führt seine Aufgaben nach festen Vorgaben aus und Sie als Entwickler müssen prüfen, was das für Ihr Programm bedeutet. Hier gibt es drei Optionen:

- (i) Es macht Sinn / erfüllt eine gewünschte Aufgaben. (ii) Es ist irrelevant.
- (iii) Sie haben einen Fehler einprogrammiert.

- Bit 31 (N / Negative) wird bei einer Operation gesetzt, wenn Sie eine signed operation ausgeführt haben und das Ergebnis ein negativer Wert ist.
- Bit 30 (Z / Zero) wird gesetzt, wenn das Ergebnis einer Operation die Zahl 0 ist. Häufigster Anwendungsfall: Vergleich zweier Zahlen.
- Bit 29 (C / Carry): Hier wird zwischen zwei Fällen unterschieden: o Bei einer unsigned addition wird das Bit gesetzt, wenn ein unsigned overflow aufgetreten ist. o Bei einer unsigned subtraction wird das Bit genau dann gesetzt, wenn Bit 28 nicht gesetzt wird.
- Bit 28 (V / Overflow): wird gesetzt, wenn eine signed operation durchgeführt wurde und außerdem ein signed overflow aufgetreten ist.

Für Bit 24 steht die Bedeutung ja bereits in der Tabelle und die Exception Types folgen im Abschnitt zu Interrupts und Exceptions.

11.4.7 PRIMASK – Das Interrupt-Maskierungs-Register

Hier handelt es sich zwar um ein vollwertiges 32-Bit-Register, aber für uns ist lediglich das LSB von Belang: Wenn wir es auf 1 setzen, dann wird der priority level effektiv auf 0 gesetzt. Da wir bislang nur kurz über Interrupts gesprochen haben, ist Ihnen nicht klar, was das bedeutet, deshalb sie hier nur kurz angemerkt, dass Interrupts die einzige Möglichkeit sind, damit der Prozessor beispielsweise auf Eingaben an einer Tastatur reagiert. Je nach Prioritätslevel werden aber bestimmte Interrupts grundsätzlich ignoriert. Und wenn dieser auf 0 gehoben wird, dann werden ausschließlich die allerwichtigsten Interrupts verarbeitet. Dazu gehört beispielsweise der Interrupt, der durch das Drücken des Resetschalters ausgelöst wird.

11.4.8 Weitere Register

Neben den hier aufgeführten Registern gibt es noch das CONTROL Register, das aber nur dann von Belang ist, wenn Sie ein Betriebssystem entwickeln wollen. In dem Fall müssten Sie sich außerdem mit dem Unterschied zwischen MSP und PSP auseinandersetzen; zweier Register, die jeweils einen individuellen Stack Pointer verwalten. Im Rahmen unserer Veranstaltung werden wir uns hiermit nicht beschäftigen.

11.4.9 Zusammenfassung

Sie kennen jetzt die wichtigsten Register, die Sie für die Programmierung des Cortex-M0 nutzen werden. Wenn Sie später mit C arbeiten, ist für Sie lediglich die Initialisierung des Stack Pointers sowie die Nutzung des xPSR von Belang. Die anderen Register „verwaltet“ C für Sie.

11.5 Die Aufteilung des Speichers

Bislang haben wir lediglich allgemein über Speicher gesprochen, also darüber, dass Speicher beim Cortex-M0 in 8-Bit Segmente unterteilt wird, die über Adressen angesprochen werden, und dass Sie regelmäßig nur Einheiten von 4-Byte-Blöcken gemeinsam aus dem Speicher in die Register (und zurück) übertragen können.

Was wir damit außen vor gelassen haben sind insbesondere zwei Aspekte: Zum einen haben wir noch nicht darüber gesprochen, dass beim Cortex-M0 bestimmte Speicherbereiche für bestimmte Aufgaben vorgesehen oder sogar reserviert sind. Zum anderen haben wir nicht über externe Speicher gesprochen und wie diese Daten in den Prozessor geladen werden können.

Deshalb sehen wir uns jetzt den zweiten Teil des Mikrocontrollers an, mit dem Sie arbeiten werden: Den internen Speicher. (Externen Speicher sprechen Sie über das SPI an, aber dazu kommen wir später.)

11.5.1 Die Aufteilung des Speichers des Cortex-M0

Sie wissen bereits, dass der Cortex-M0 bis zu 4 GB an Speicher direkt ansprechen kann, weil es sich um einen 32-Bit-Prozessor handelt. Im Falle des von uns verwendeten Entwicklerboards von Olimex, auf dem ein LPC1114 von NXP zum Einsatz kommt, sind jedoch lediglich 32 KB programmierbar. Allerdings könnten wir mit der Evaluationsversion des MDK von Keil gar nicht mehr als 32 KB internen Speicher nutzen. Es folgt eine Aufstellung, aus der hervorgeht, wie beim Cortex-M0 4 GB an Speicher aufgeteilt werden:

Speicherbereich	Größe	Inhalt
0x 0 ... 0x 1F FF FF FF	0,5 GB	Programm Code und Exception Vector Tabelle
0x 20 00 00 00 ... 0x 3F FF FF FF	0,5 GB	Datenspeicher (z.B. static RAM)
0x 40 00 00 00 ... 0x 5F FF FF FF	0,5 GB	Peripherie
0x 60 00 00 00 ... 0x 9F FF FF FF	1 GB	Externer Speicher
0x A0 00 00 00 ... 0x DF FF FF FF	1 GB	Externe Peripherie
0x E0 00 00 00 ... 0x FF FF FF FF	0,5 GB	Systeminterna und private Peripherie

Beachten Sie bitte, dass es sich bei dieser Aufstellung nur um einen möglichen Fall handelt. Sie müssen für jeden Mikrocontroller prüfen, wie dort die Speicherverteilung organisiert ist. Dazu greifen Sie auf die jeweilige Dokumentation des Herstellers zurück.

Nachtrag zum SP: Bei ausreichend Speicher wird der Stack Pointer mit der Adresse 0x40000000 initialisiert, da er so bei der höchsten Adresse des Da-

tenspeichers beginnt und damit so spät wie möglich mit Programmteilen kollidieren dürfte. Letzteres ist auch der Grund, warum er in umgekehrter Reihenfolge adressiert.

11.5.2 Weitere Bezeichnungen für Speichereinteilungen

Im vorigen Abschnitt haben Sie die Aufteilung des Speichers nach den enthaltenen Daten kennen gelernt. Gelegentlich wird bei der Einteilung des Speichers auch eine Dreiteilung vorgenommen. Dann bezeichnet man Speicherbereiche mit Data, Heap und Stack.

Es gibt auch Fälle, in denen es mehrere Data/Heap/Stack-Bereiche gibt. Damit wären wir aber bei der Programmierung von embedded OS und/oder einer Form der Verarbeitung paralleler Prozesse. Und das geht über den Inhalt dieses Kurses hinaus. Also werden wir hier davon ausgehen, dass es jeweils nur einen Stack, einen Heap und einen Data-Bereich gibt.

Den Stack haben Sie bereits bei der Einführung in die Arbeitsweise des Stack Pointers im Bereich der Register des Cortex-M0 kennen gelernt. Deshalb an dieser Stelle keine weiteren Erläuterungen dazu.

Mit Data meint man bei dieser Aufteilung den Speicherbereich, der mit den Programmdateien belegt ist, die vor dem Start des Rechners belegt sind.

Mit Heap ist dagegen der Speicherbereich gemeint, der während der Laufzeit des Programms belegt wird. Dieser ist also nicht fest abgegrenzt, sondern ändert sich ähnlich wie der Stack kontinuierlich während der gesamten Laufzeit des Programms. Im Gegensatz zum Stack beginnt der Heap direkt nach dem Data-Bereich und wächst in naiver Reihenfolge. Damit ist gemeint, dass neue Daten im Heap an Adressen abgelegt werden, die höher als die bisherigen sind. Der Stack dagegen wächst ja von hoher zu niedriger Adresse.

Somit kann es bei größeren Programmen passieren, dass Stack und Heap kollidieren, dass also ein neuer Eintrag im Heap einen bestehenden Eintrag im Stack überschreibt oder umgekehrt. Diese Fälle werden im Rahmen dieses Kurses im Regelfall nicht auftreten. Sie müssen sich aber bewusst sein, dass auch diese Fälle von Ihnen als Programmierer später berücksichtigt werden müssen. Denn auch hier gilt: Der Mikroprozessor ist gar nicht in der Lage, solche Fälle zu erkennen, also müssen Sie sich darum kümmern.

11.5.3 Anbindung von externem Speicher über den SPI Controller

Sie wissen bereits, dass das Entwicklerboard eine Schnittstelle zur Nutzung externer Peripherie beinhaltet. Über diese können Sie unter anderem einen Speicherbaustein ansprechen, um so die ersten Schritte beim Arbeiten mit externem Speicher durchzuführen. Wie das genau funktioniert folgt später.

Kontrolle

Sie wissen jetzt, dass es zwar ein Standardmodell für die Verteilung der verschiedensten Daten im Speicher eines Cortex-M0 gibt, dass Sie aber für jedes Modell in der Dokumentation des Herstellers prüfen müssen, wie diese Verteilung im Einzelfall realisiert wurde. Das müssen Sie wissen, da Sie bei der maschinennahen Programmierung selbst die Nutzung des Speichers programmieren müssen. Wenn Sie hier Fehler machen, kann es passieren, dass Sie Daten überschreiben, die für die Systemsteuerung von Belang sind.

Sie wissen weiterhin, dass die Bezeichnungen Stack, Heap und Data verwendet werden, um die Speicherbelegung während der Laufzeit von Programmen abzugrenzen.

Außerdem wissen Sie, über welches Interface Sie üblicherweise externen Speicher anschließen können.

11.6 Exceptions und Interrupts

Diejenigen von Ihnen, die bereits eine objektorientierte Sprache erlernt haben kennen deshalb die sogenannten Exceptions. Dort haben Sie Exceptions vorrangig als eine Spezialform von Fehlern kennen gelernt, die während der Laufzeit eines Programms bereinigt werden können. Außerdem sind Exceptions dort auch eine Möglichkeit, um unterschiedliche Fälle im Programmablauf effizient zu steuern.

Bei der maschinennahen Programmierung ist ebenfalls die Rede von Exceptions, hier stellen Sie aber einen Spezialfall der sogenannten Interrupts dar. Wie Sie schon mehrfach gelesen haben sind Interrupts (dt.: Unterbrechungen) so etwas wie Änderungen im Programmablauf. Im Gegensatz zu einem Branch geht es hier aber nicht darum, häufig auftretende Programmabläufe in Form von Subroutinen auszulagern. Vielmehr ist ein Interrupt oder besser gesagt ein IRQ (kurz für external Interrupt ReQuest) die Anfrage einer Peripherie, mit der diese den Prozessor auffordert, seinen aktu-

KAPITEL 11. EINFÜHRUNG IN DEN AUFBAU VON ARMV6-PROZESSOREN AM BEISPIEL EINER

ellen Programmablauf zu unterbrechen, und dafür eine externe Eingabe zu verarbeiten. (Das passiert z.B. wenn der Nutzer eines Rechners eine Taste drückt.)

Sie müssen sich zum Verständnis klar machen, dass ein Prozessor im Grunde nur Zeile für Zeile ein Programm abarbeiten und ansonsten keine weiteren eigenständigen Entscheidungen treffen kann. Bei Mehrkernprozessoren trifft das zwar auf jeden Kern individuell zu, aber am Grundprinzip ändert sich nichts. Es muss also eine Möglichkeit geben, damit Komponenten eines Rechners, die nicht Teil des Prozessors sind, den Prozessor dazu bewegen können seine Arbeit zu unterbrechen, um andere Prozesse auszuführen. (Im Falle eines Tastendrucks durch den Nutzer könnte das z.B. die Speicherung des Buchstaben sein, dessen Taste der Nutzer gedrückt hat.) Danach soll er wieder seine unterbrochene Arbeit fortsetzen.

Wenn Sie das verstanden haben, dann haben Sie auch begriffen, warum ein Mehrkernprozessor u.U. genauso langsam ist wie ein ansonsten identisch gebauter Einzelkernprozessor: Die einzelnen Kerne haben keine Möglichkeit ohne externe Kontrolle als Verbund zu agieren.

Das mag Ihrer bisherigen Vorstellung von Computern widersprechen, weil Sie es i.d.R. gewohnt sind, dass Sie per USB eine neue Komponente an den Rechner anschließen können und Sie dann quasi als Teil des Computers nutzen können. Doch das ist eben nur die oberflächliche Perspektive, die einem naiven Nutzer vorgaukelt, so ein Computer sei ein ganz simples Gerät, das trotzdem unglaubliche Möglichkeiten bietet. Tatsache ist: Ein Computer ist wesentlich komplexer als es für einen Nutzer den Anschein hat. Interrupts sind das einzige Mittel, damit externe Komponenten überhaupt irgendwie genutzt werden können. Und wir reden hier von wesentlich mehr Dingen als nur von Maus, Tastatur, Drucker, usw.

Sie können beim Cortex-M0 übrigens bis zu 32 Arten von IRQs definieren. Es ist zusätzlich möglich einen IRQ von mehreren Komponenten gemeinsam nutzen zu lassen.

Kontrolle:

Ein Interrupt ist ein so etwas wie ein Signal, mit dem eine beliebige Komponente den Prozessor auffordert, den aktuellen Programmablauf zu unterbrechen und eine andere Aufgabe zu erfüllen. Im Gegensatz zu höheren Programmiersprachen ist hier bereits die Verarbeitung eines Tastendrucks eine eigenständig zu verarbeitende Aufgabe.

Aufgabe:

Was ist der Unterschied zwischen einem IRQ und einem Branch?

11.6.1 Exception Types

Sie erinnern sich noch? Die letzten fünf Bit des xPSR zeigen den Exception Type an. Jetzt kommen wir dazu, wie Sie Exception Types programmieren: An Anfang des Quelltextes legen sie für jede Exception eine Speicheradresse fest, an der der Code steht, der auszuführen ist, wenn die Exception auftritt.

Es gibt 35 Exception Types, die Sie nutzen können: Die Exceptions 1 bis 3 (Reset, NMI, Hard fault) und 11, 14, 15 (SVCall, PendSV und SystTick) haben feste Bedeutungen, aber freie Adressen. Die Exceptions 16 bis 47 sind frei belegbar. Ausschließlich über diese 33 IRQs können Sie dem Prozessor Eingaben vermitteln.

Hinweis:

Damit programmieren Sie lediglich, wo sich der Programmcode befindet, der bei einem bestimmten IRQ ausgeführt werden soll, aber nicht, wodurch er ausgelöst wird. Das tun Sie z.B. bei der Zuordnung von Eingaben des SPI.

Beispiel:

Sie wollen ein Spiel programmieren, bei dem Sie vier Tasten für die entsprechende Richtung und zwei Tasten für anwählen und abbrechen nutzen wollen. Dann könnten Sie beispielsweise den Exception Types 0x20 bis 0x23 Startadressen zuordnen, die den Code enthalten, der besagt, was passiert, wenn in eine Richtung gesteuert wird. Und den Exception Types 0x25 und 0x26 ordnen Sie die Startadressen mit dem Code für anwählen und abbrechen zu.

Sollte Sie diese Formulierung „Startadresse zuordnen“ verwirren, dann machen Sie sich bitte bewusst, dass bei der maschinennahen Programmierung ein Programm aus Zeilen besteht, die an aufeinander folgenden Zeilen im Speicher abgelegt sind. Die Startadresse ist dann schlicht die Adresse, an der ein Programm bzw. ein Programmteil beginnt.

Alle IRQs, die soeben nicht aufgeführt wurden sind für das System reserviert und müssen mit bestimmten vorgegebenen Werten initialisiert werden. (Ignorieren Sie bitte für den Augenblick alle IRQs außer den Nummern 1 und 16 bis 47.)

Den Reset IRQ können Sie im Bereich der maschinennahen Programmierung als eine Art Neustart verstehen. Im Gegensatz zur Digitaltechnik ist damit also nicht gemeint, dass ein Eingang auf 0 gesetzt wird. Er muss so programmiert werden, dass er auf den Anfang des eigentlichen Programms verweist.

! LSB + 1 !

Hier (im Gegensatz zur Initialisierung des SP, aber dafür bei allen IRQ Initialisierungen) haben wir wieder einen Fall, in dem wegen des Thumb State das LSB auf 1 gesetzt werden muss. Wenn die erste auszuführende Zeile also in der Adresse 0x10C steht, so müssen Sie dem Reset IRQ die Adresse 0x1D zuordnen.

Wenn Sie die Formulierung „auf den Anfang des eigentlichen Programms“ verwirrt: Am Beginn Ihres Quellcodes initialisieren Sie die Adressen für den SP und die IRQs. Diese Initialisierung brauchen Sie bei einem Reset IRQ nicht erneut durchführen lassen, da Sie bis zur Überschreibung des Speichers durch eine Neuprogrammierung erhalten bleibt. Sie können also direkt zum auszuführenden Teil Ihres Programms springen. Dieser ist hier mit dem Anfang des eigentlichen Programms gemeint.

11.6.2 Initialisierung von IRQs

Eine Initialisierung ist die erstmalige Belegung eines Elements mit einem Wert. Elemente in diesem Sinne können so etwas wie die Bezeichner von IRQs oder auch andere Bezeichner in einem Programm sein. Ob dieser Wert nun als Zahl oder als Adresse zu interpretieren ist hängt vom Kontext ab, macht aber für das, was Sie als Initialisierung programmieren keinen Unterschied.

An dieser grundsätzlichen Bedeutung ändert sich übrigens nichts, wenn Sie z.B. über die imperative Programmierung in einer höheren Sprache reden. Dort werden dann zwar andere „Dinge“ initialisiert, aber es geht immer noch um nichts anderes, als das erstmalige Abspeichern eines Wertes zu dem jeweiligen „Ding“.

Die Initialisierung eines IRQ besteht also darin, dass Sie (wie zuvor beim SP) schlicht eine Adresse des Speichers „eintragen“. An dieser Adresse beginnt dann der Programmcode derjenigen „Subroutine“, die im Falle dieses IRQs ausgeführt werden soll. Diese Idee unterscheidet sich im Grunde nicht groß von einem Branch, nur dass ein Branch einen Programmteil aufruft, der grundsätzlich zu einem bestimmten Zeitpunkt bei der Ausführung

eines Programmes ausgeführt werden soll. Ein Interrupt dagegen (deshalb auch die Bezeichnung als Unterbrechung) soll diesen Programmablauf unterbrechen und einen anderen Programmteil einschieben.

Wie mehrere IRQs behandelt werden, die parallel zueinander auftreten, werden wir hier nicht explizit ansprechen. Sie brauchen sich über solche Fälle vorerst keine Gedanken zu machen. Wichtig ist, dass Sie das Grundprinzip von IRQ und Branch verstanden haben.

11.6.3 IRQs im Quellcode

Nun aber zur eigentlichen Programmierung: Sie erinnern sich noch daran, wie Sie in einem Programm den Stack Pointer initialisieren mussten? Wesentlich einfacher aber programmiertechnisch sehr ähnlich initialisieren Sie die IRQs: Sie legen für jeden IRQ fest, an welcher Adresse der Programmcode zu finden ist, der ausgeführt werden soll, wenn dieser IRQ erfolgt. Im Gegensatz zum Stack wird der Programmcode aber in der naiven Reihenfolge abgearbeitet: Der Anfang eines Programms befindet sich bei einer kleineren Adresse als das Ende.

Denken Sie bitte wieder daran, dass die Startadressen jeder Subroutine durch 4 teilbar sein muss, weil der Prozessor immer 32 Bit am Stück einliest, auch wenn eine Programmzeile nur 8 oder 16 Bit lang ist.

11.6.4 Codebeispiel

Das folgende Beispiel ist nur ein Auszug, der einige IRQs außen vor lässt, die Sie ebenfalls programmieren müssen:

Sie wollen dem SP den Wert bzw. die Adresse 0x1000 0268, dem Reset IRQ (IRQ 1) den Wert 0x1B0 und dem ersten frei programmierbaren IRQ (IRQ 16) den Wert 0x1C zuordnen, dann sieht Ihr Quellcode wie folgt aus:

DCW 0x0268 ; Die LSB+1-Regel gilt nicht bei der SP-Initialisierung. DCW 0x1000 DCW 0x01B1 ; Thumb! Bei den IRQs muss wieder das LSB auf 1 gesetzt werden: 1 nicht 0! DCW 0x0000 ... Jetzt folgen 28 Zeilen, die den 14 IRQs zwischen Reset IRQ und dem ersten frei programmierbaren IRQ Adressen zuordnen. ... DCW 0x001D ; Thumb! Bei den IRQs muss wieder das LSB auf 1 gesetzt werden: D nicht C! DCW 0x0000 ...

Jetzt folgen noch 62 Zeilen, die den verbleibenden 31 IRQs jeweils einen Wert zuweisen. IRQs, die nicht verwendet werden sollen, können beispielsweise mit der Adresse initialisiert werden, mit der der Reset IRQ initiali-

KAPITEL 11. EINFÜHRUNG IN DEN AUFBAU VON ARMV6-PROZESSOREN AM BEISPIEL EIN

siert wurde. Wie angekündigt haben wir hier eine Reihe an IRQ Initialisierungen ignoriert, damit Sie das Grundprinzip verstehen. So können Sie sich leichter auf diejenigen Initialisierungen konzentrieren, die für Sie am wichtigsten sind.

Fällt Ihnen das schwer? Kein Problem, auch diese Arbeit nimmt Ihnen bei der Arbeit mit C die Programmierumgebung ab. Dennoch sollten Sie grundsätzlich verstanden haben, was hier passiert, damit Sie nachvollziehen können, warum bestimmte Teile im C-Code Ihres Programms auftauchen, ohne dass Sie sie programmiert hätten.

Anmerkung: Die Reset Exception wird im Thread Mode ausgeführt, in dem ja auch die Ausführung des Programmcodes stattfindet. Bei anderen Exceptions werden Sie später den Wechsel zum Exception Handler und zurück zum Thread Handler programmieren müssen.

Kontrolle

Sie kennen den Unterschied zwischen einem Branch und einem Interrupt bzw. einer Exception. Sie wissen, wie Sie in Assembler beim Cortex-M0 Adressen zu einem bestimmten IRQ zuordnen, und dass man das Initialisierung nennt.

Sie denken stets daran, dass Sie bei den Initialisierungen der IRQs jeweils eine 1 addieren müssen, auch wenn das bei der Initialisierung des SP nicht der Fall ist.

Sie wissen weiterhin, dass Sie bei jedem Programm für den Cortex-M0 in Assembler insgesamt 47 IRQ Initialisierungen durchführen müssen, auch wenn Sie insbesondere bei den vom System reservierten Fällen noch nicht wissen, welche Adressen hier jeweils zuzuordnen sind.

Sie wissen, dass für die Initialisierung eines IRQ bzw. des SP jeweils zwei Programmzeilen nötig sind.

Sie wissen auch warum das so ist, weil Sie die Inhalte der vorigen Abschnitte dieses Kapitels nicht vergessen haben.

Und vor allem freuen Sie sich, weil Sie all das in C nicht zu beachten brauchen.

11.7 NVIC – Der Nested Vectored Interrupt Controller

Im vorigen Abschnitt haben Sie erfahren, wie Sie dem Prozessor Programmcode übermitteln können, den dieser im Falle eines bestimmten Interrupts ausführen soll. Der NVIC (kurz für Nested Vectored Interrupt Controller) kontrolliert schlicht, wann welcher IRQ tatsächlich an den Prozessor weitergeleitet wird. Für diese Kontrolle ist unter anderem der priority level jedes IRQs von Belang.

Damit er seine Aufgabe möglichst schnell ausführen kann wird ein Virtualisierungsverfahren angewendet, das als Memory Mapping bezeichnet wird. Bei der Programmierung werden Sie also nicht den NVIC selbst programmieren, sondern Sie werden Speicherbereiche ändern, was am Ende dazu führt, dass der NVIC seine Arbeit so durchführt, wie Sie das wollen. Außer bei denjenigen von Ihnen, die diese ganzen Erklärungen ignorieren und einfach den Programmcode kopieren, den sie irgendwo im Netz oder bei anderen Kommilitonen kopiert haben und die dann gnadenlos durch die Klausur fallen, weil sie nicht wissen, was all das hier soll.

11.7.1 Memory Mapping

Im Regelfall stellen die Datenübertragungswege zwischen den Komponenten eines Rechners einen Flaschenhals dar, weil sie Daten zum Teil wesentlich langsamer übertragen, als die Komponenten sie verarbeiten können. (Erinnern Sie sich bezüglich der Gründe bitte an die Hinweise zur Nachrichtentechnik aus dem ersten Kapitel.) Eine Möglichkeit, um diesen Flaschenhals zu entschärfen besteht nun darin, Daten im Speicher abzulegen und sie danach erst zu übertragen. Und das bezeichnet man als Memory Mapping.

Auf die Details werde ich hier nicht eingehen, da Sie für die Zwecke dieser Veranstaltung lediglich wissen müssen, dass Sie auf andere Register zugreifen, indem sie Werte ändern, die im Speicher gespeichert sind. Sie werden in dieser Veranstaltung also weder erfahren, wie Memory Mapping den genannten Flaschenhals entschärft, noch werden Sie erfahren, wie der Mikrocontroller die Datenübertragung durchführt, nachdem Sie die entsprechenden Werte geändert haben.

Mehr Details zu Virtualisierungstechniken wie dem Memory Mapping erfahren Sie in Veranstaltungen zu Betriebssystemen. Alternativ empfehle ich Ihnen den Band „Operating Systems: Three Easy Pieces“ von Remzi H. und Andreas C. Arpaci-Dusseau, das Sie als open book über die Webseite <http://www.ostep.org> beziehen können.

11.8 Das SPI – Serial Peripheral Interface

Wie am Beginn des Kapitels beschrieben handelt es sich hier um eine Schnittstelle, die auf dem Olimex Entwicklerboard enthalten ist und über die Peripherie angesprochen werden kann, die dem eigentlichen Mikrocontroller nicht bekannt ist. Das bedeutet, dass Sie hierüber Komponenten in den Programmablauf einbinden können, die Sie selbst mit dem Entwicklerboard verbunden haben.

11.9 Grundlagen der eigentlichen Programmierung

Wenn Sie dieses Buch vom Anfang an gelesen haben, dann werden Sie schon mehrfach gedacht haben: Wann fangen wir denn endlich mit der Programmierung an?

Vielleicht erinnern Sie sich noch an die einführenden Hinweise dazu, was an Java der Vorteil für Programmierneinsteiger ist? Genau, Sie brauchen bei Java (im Gegensatz zur maschinennahen Programmierung) nichts darüber zu wissen, wie ein Computer aufgebaut ist; Sie können nach der Installation des JDKs und eines Editors direkt loslegen. Bei der maschinennahen Programmierung müssen Sie dagegen zunächst all das wissen, was in den bisherigen Abschnitten dieses Kapitels enthalten war, um auch nur ein einfaches Programm erstellen zu können: Register und Speicheraufteilung. Von den wirklich interessanten Dingen wie der Eingabe von Daten durch einen Nutzer sind wir immer noch weit entfernt. Die fangen dann an, wenn wir den SPI-Controller mit ins Spiel bringen.

11.9.1 Die Startsequenz des Cortex-M0

Mit Startsequenz ist gemeint, in welcher Reihenfolge der Cortex-M0 welche Schritte ausführt, nachdem er gestartet wurde. Die ersten zwei Schritte kennen Sie jetzt schon: Die Initialisierung des SP und die Initialisierung der Exception Types.

Es gibt weiterhin die Möglichkeit, dass noch vor diesen Initialisierungen ein Bootloader zum Einsatz kommt, aber das ist etwas, womit wir es nur dann zu tun haben, wenn ein embedded OS gestartet wird. Also brauchen wir uns darum im Rahmen dieses Kurses nicht zu kümmern.

Nach den genannten Initialisierungen springt das Programm an die Adresse, die im Reset IRQ genannt ist. Hier kann nun die Systeminitialisierung erfolgen. Damit ist die Initialisierung verschiedener Komponenten wie beispielsweise der Clock Control Circuitry oder des PLL (Phase Locked Loop)

gemeint. Keine Sorge, wenn Ihnen diese Begriffe noch nichts sagen; für den Moment müssen Sie nur wissen, dass es dabei um Einstellungen geht, die steuern, wie sich der Mikrocontroller generell verhält. So müssen Sie beispielsweise in Abhängigkeit von angeschlossener Peripherie teilweise die Taktrate senken. Aber dazu später mehr. Diese Systeminitialisierung kann auch im Rahmen der `main()`-Methode eines C-Programms erfolgen. Im Falle des MDK brauchen Sie sich um die Systeminitialisierung nicht selbst zu kümmern. Das MDK generiert den nötigen Code automatisch.

Nach der Systeminitialisierung wird bei C-Programmen noch der C startup code benötigt. Wie Sie wissen programmieren Sie den Cortex-M0, indem Sie memory gemappte Register verändern, die dann die Einstellungen der verschiedenen Komponenten des Mikrocontrollers ändern. Bei Assembler nutzen Sie hierfür entweder die tatsächlichen Speicheradressen oder Bezeichner, die vom eingesetzten Assembler abhängen. Im Gegensatz dazu verwenden Sie in Sprachen wie C Variablen, Pointer und Funktionen. Deshalb wird noch ein Programmbereich benötigt, der die Variablen und den Speicherbereich zuordnet, die von Ihrem C-Programm genutzt werden. Dieser Programmbereich wird als C Startup Code bezeichnet. Genau wie bei der Systeminitialisierung brauchen Sie sich beim MDK nicht um den C Startup Code zu kümmern; das MDK generiert ihn selbst.

Kontrolle

Das MDK nimmt Ihnen (wie verschiedene andere IDEs) verschiedene Programmieraufgaben ab, die für die Lauffähigkeit des Programms nötig sind. In diesem Fall geht es um die Systeminitialisierung und den C Startup Code.

Erstere nimmt grundlegende Konfigurationsschritte vor, durch die das System überhaupt erst durch ein Programm genutzt werden kann, letztere stellt gewissermaßen eine Schnittstelle zwischen Ihrem C-Programm und dem System dar.

Das bedeutet für Sie: Weil wir das MDK nutzen müssen Sie bei der Programmierung lediglich den SP und die IRQs initialisieren. Mehr können und brauchen Sie noch nicht.

11.9.2 Prozesse – So planen Sie Ihr Programm

Diesen Abschnitt können Sie vorerst überspringen; die enthaltenen Informationen werden dann wichtig, wenn Ihr Programm eine Vielzahl an Aufgaben erfüllt oder die Aufgabe(n) durch parallel arbeitende Prozesse realisiert wird. Kurz gesagt sind wir hier im Bereich Software Engineering.

Bei größeren Projekten gilt: Bevor Sie drauflos tippen, sollten Sie sich einen Überblick über die verschiedenen Prozesse verschaffen, die im Rahmen Ihres Programms aktiv sind. Dann müssen Sie planen, wie Sie diese Prozesse im Rahmen des Programmablaufs koordinieren wollen.

Polling

Mit Polling bezeichnet man die einfachste Strategie: Alle Prozesse werden nacheinander abgearbeitet. Wenn es Prozesse mit unterschiedlicher Priorität gibt, sollten Sie eine andere Strategie wählen.

Interrupt Driven

Hier lagern Sie Prozesse in sogenannte ISRs (Interrupt Service Routinen) aus und lassen Sie durch IRQs aufrufen. So können Sie Prioritäten ins Programm bringen.

Concurrent Prozesse

Diese Strategie gehört in den fortgeschrittenen Bereich: Sie lassen jeden Prozess für eine gewisse Zeit laufen, sodass der Nutzer den Eindruck erhält, alle Prozesse würden gleichzeitig laufen. Entweder Sie programmieren diese Gleichzeitigkeit selbst oder Sie nutzen ein sogenanntes RTOS (Real Time Operating System). Diese Programmierung (also ohne die Nutzung eines RTOS, das Ihnen diese Aufgabe abnimmt) ist deshalb ein fortgeschrittenes Thema, weil Sie hier z.B. sogenannte Deadlocks vermeiden müssen. Das sind Situationen, in denen ein Prozess auf die Ausgabe eines anderen Prozesses wartet, der aber wiederum auf die Ausgabe des ersten Prozesses wartet. Damit hängt das Programm fest und es passiert scheinbar nichts mehr.

Wichtig: Der Begriff Realtime System bzw. Echtzeitsystem bezeichnet im Gegensatz dazu komplexe IT Systeme, die wesentlich höhere Anforderungen an die Geschwindigkeit bei der Datenübertragung zwischen den einzelnen Komponenten haben, als das z.B. im Internet der Fall ist. Wenn wir über solche Echtzeitsysteme reden, meinen wir z.B. Systeme die dynamisch auf Änderungen der Temperatur eines Kernreaktors reagieren und scheinbar ohne Zeitverlust verschiedene Zu- und Abflüsse des Reaktors in Abhängigkeit von diesen Änderungen steuern.

Kontrolle

Dieser Abschnitt gehört in den Bereich nice to know.

11.9.3 Ein- und Ausgaben

Wie schon mehrfach angemerkt müssen Sie bei der Programmierung eines Mikrocontrollers auch für solche Prozesse Code implementieren, die Sie bei höheren Programmiersprachen als von der Programmiersprache „erledigt“ betrachten können. Ein Bereich, in dem Sie hierdurch einen wesentlich höheren Programmieraufwand bekommen sind Ein- und Ausgaben.

Um diese zu verarbeiten müssen Sie zunächst wissen, was für ein Interface genutzt wird, um die Ein- bzw. Ausgaben zu übertragen. Mit Interface können zwar auch die „Steckdosen“ gemeint sein, die Sie auf dem Entwicklerboard sehen können, aber bei der Programmierung spielt die physikalische Bauweise keine Rolle. Es folgen einige Anschlussstandards, die Sie wahrscheinlich noch nicht kennen:

- Digitale I/Os • UART • I2C • SPI • Und viele andere mehr. Die folgenden Anschlussstandards werden Sie dagegen wahrscheinlich bereits kennen: • USB • Ethernet • CAN • Graphic LCD • SD Card

Aber nochmal: Sie dürfen bei der Entwicklung von Software für Mikrocontroller nicht erwarten, einen USB-Stick in den entsprechenden Anschluss zu stecken und dann damit arbeiten zu können. Vielmehr müssen Sie einige Vorbereitungen programmieren, damit die Anschlüsse überhaupt in Ihrem Programm genutzt werden können. Die tatsächliche Datenübertragung (die ja die Nutzung eines solchen Anschlusses darstellt) ist damit aber noch nicht erfüllt, sondern muss ebenfalls eigenständig von Ihnen programmiert werden.

Anbindung von Anschlüssen an den Mikrocontroller

Sehen wir uns zunächst an, wie ein solcher Anschluss (nehmen wir hier einen SPI-Controller) mit dem Mikroprozessor verbunden ist. Sie wissen ja bereits, dass der Mikroprozessor ausschließlich über ein Interface Daten erhalten kann. Sie wissen ebenfalls, dass es nicht möglich ist, dem Mikroprozessor über dieses Interface Befehle zu erteilen, sondern dass Sie sich hier eines Interrupts bedienen müssen, der vom NVIC weitergeleitet oder verworfen wird.

Was Sie aber noch nicht wissen ist, dass Anschlüsse wie ein SPI-Controller mit der sogenannten Peripherie verbunden sind und somit nur dann genutzt werden können, wenn die Register dieser Peripherie entsprechend initialisiert sind.

Und wie konnte es anders sein: Das ist ein komplexerer Prozess als die

Initialisierung des SP und der IRQs.

Initialisierung der Peripherie

In aller Regel sind vier Schritte nötig, um eine Peripherie zu initialisieren:

1. Konfiguration der clock control circuitry: Wie Sie wissen arbeiten Rechner mit einer sogenannten Taktung, sprich sie führen eine exakte Anzahl an Operationen pro Sekunde aus. Besser gesagt: Nach einem festen Zeitintervall führen Sie genau eine Operation aus. Bislang werden Sie wahrscheinlich keine Konfiguration der Taktung für einzelne Komponenten des Systems durchgeführt haben. Doch auch das müssen Sie jetzt tun. Der Grund ist recht einfach: Jede Komponente hat eine maximale Taktung, mit der sie arbeiten kann. Aber die Tatsache, dass eine Komponente an Ihr Entwicklerboard angeschlossen werden kann (z.B. über UART) bedeutet natürlich nicht, dass sie genau dieselbe maximale Taktung hat wie Ihr Prozessor. Und da Sie eben jedes Detail selbst programmieren müssen, müssen Sie auch für jede Komponente eine passende Taktung programmieren. Eine zu hohe Taktung bedeutet zum Glück normalerweise nur, dass eine Komponente nicht auf Datenübertragungen reagiert; im Bereich der maschinennahen Programmierung gibt es dagegen durchaus Fälle, in denen Sie mit entsprechenden Befehlen bzw. Programmen Schäden bewirken können.

2. Programmierung der I/O-Konfiguration Aktuelle Mikroprozessoren in Desktoprechnern haben mehr als 1000 Kontakte bzw. Beinchen. Und jeder dieser Kontakte entspricht einem Bit, das Sie setzen oder löschen können. Zur Erinnerung: Das bedeutet schlicht, dass Sie das Eingangssignal an- oder ausschalten, indem Sie eine 0 oder eine 1 einprogrammieren. Indem Sie also bei der maschinennahen Programmierung 32-stellige Binärwerte programmieren, setzen und löschen Sie gleichzeitig 32 Eingänge des Prozessors. Nun werden Sie sich wundern, welchen Sinn es hat, dass z.B. ein 32-Bit-MCU mehr als 32 Kontakte hat; schließlich kann er ja nur 32 Bit pro Rechenschritt verarbeiten. Aber das ist recht simpel: Hier handelt es sich um die Ein- und Ausgänge des Mikrocontrollers, nicht um die Ein- und Ausgänge des Mikroprozessors. Und als zweiten Schritt nach der Taktung jeder Peripherie, die Sie nutzen wollen, müssen Sie konfigurieren, über welchen I/O-Pin (also über welches Beinchen des Controllers) Daten mit der jeweiligen Peripherie ausgetauscht werden sollen. Dabei ist es möglich, mehrere periphere Komponenten über denselben I/O-Pin kommunizieren zu lassen.

3. Konfiguration der Peripherie Erst jetzt kommt die eigentliche Konfiguration der Peripherie an die Reihe. Zur Erinnerung: Sie nutzen jede Peripherie, indem Sie auf die memory gemappten Register dieser Peripherie zu-

greifen. Also konfigurieren Sie die Peripherie, indem Sie die entsprechenden Speicherbereiche deklarieren und initialisieren.

4. Interrupt Konfiguration Warum Sie nun Interrupts für jede Peripherie konfigurieren müssen, sollte Ihnen nach den vorigen Abschnitten klar sein. Wichtig ist lediglich, dass Sie daran denken, dass die IRQ-Konfiguration der vierte Schritt bei der Konfiguration/Initialisierung jeder Peripherie bzw. jedes Anschlusses an Ihren Mikrocontroller ist. Kontrolle

Sie wissen jetzt, dass Sie für jede angeschlossene Komponente vier Bereiche programmieren müssen, egal, ob diese von Ihnen selbst am Mikrocontroller angeschlossen wurde oder ob sie Teil des Entwicklerboards ist. Erst danach können sie diese Komponente in Ihr Programm einbinden.

Damit haben Sie jetzt endlich einen zweiten Bereich kennen gelernt, den Sie (nach der zuvor eingeführten Initialisierung von SP und IRQs) programmieren können. Dumm nur, dass Ihr Programm damit immer noch nichts tut...

11.9.4 Eingebettete Systeme und Debugging im laufenden Betrieb

Dieser Abschnitt ist zwar im Grunde etwas zu früh, aber da wir uns soeben angesehen haben, wie man Anschlüsse in den Programmablauf einbinden kann, kommen wir jetzt dazu, wie man diese Anschlüsse nutzen kann, um im laufenden Betrieb Rückmeldungen z.B. über Fehler zu bekommen.

Denn wie beschrieben haben eingebettete Systeme im Regelfall kein Display und sind auch sonst quasi nicht sichtbar. Deshalb müssen Sie im Regelfall Meldungen des Systems über einen der Anschlüsse (z.B. die UART-Schnittstelle) ausgeben lassen, die Sie dann mittels eines Rechners auswerten können.

Da die Schnittstellen aber im Regelfall kein C beherrschen müssen Sie hier noch eine Übersetzung zwischenschalten, die die Fehlermeldungen (z.B. in Form von printf()-Ausgaben) in eine Form übersetzen, die über UART übertragen werden kann. Eine solche Übersetzung wird als retargeting bezeichnet.

Kontrolle

Sie wissen schon... Ein weiterer Abschnitt, den Sie aktuell unter der Kategorie nice to know ablegen können.

11.9.5 Mangelnde Portabilität und das CMSIS

Bislang haben Sie fast ausschließlich etwas darüber gelesen, wie Sie konzeptionell an das Programmieren herangehen müssen; konkrete Programmierzeilen waren Mangelware. Sie wissen bereits, dass Sie bei der Programmierung von Mikroprozessoren in Assembler und in C je nach Mikroprozessor zum Teil andere Befehle nutzen müssen, weil es hier keinen Standard gibt.

Leider gilt das nicht nur für unterschiedliche Mikroprozessoren, sondern auch für unterschiedliche IDEs und Compiler. Bei allen Abschnitten dieses Kapitels, in denen es um konkrete Programmierzeilen geht, müssen Sie sich also im Klaren sein: Wenn Sie nicht die Ausstattung unseres Labors nutzen (Olimex LPC1114 Entwicklerboard, Keil MDK Evaluationsversion), dann werden Sie den Programmcode teilweise anpassen müssen, weil entweder Ihr Mikrocontroller oder Ihre IDE bzw. Ihr Compiler und Linker andere Befehle benötigen.

Die Entwickler bei ARM haben aber vor Jahren begonnen, eine Lösung für dieses Problem zu entwickeln (wir sprechen also wieder einmal über Software Engineering) und das sogenannte CMSIS (kurz für Cortex Microcontroller Software Standard) entwickelt, das kontinuierlich erweitert wird. Es ist Teil vieler IDEs, kann aber auch kostenlos von der ARM Webpage heruntergeladen werden. In der Version 1.3 bot es den standardisierten Zugriff auf das NVIC, den SCB und den SysTick. Weiterhin enthielt es verschiedene Funktionen, mit denen unterschiedliche C und Assembler Compiler gleich genutzt werden konnten. In der Einführung in die Programmierung mit C erfahren Sie, wie Sie das CMSIS einsetzen können.

Kontrolle

Da für diesen Kurs keine Kenntnisse des Software Engineering vorausgesetzt werden und Software Engineering auch eigentlich nicht Teil dieses Kurses ist, können Sie die Vorteile des CMSIS und des im nächsten Abschnitt vorgestellten ABI noch nicht richtig einschätzen. Sie sollten sich allerdings die Abkürzungen merken und in den kommenden Wochen prüfen, welche Lösungen diese beiden Standards Ihnen bieten.

Langfristig gehört die Kenntnis dieses Bereiches zu den Kernkompetenzen, die Sie bei der Programmierung für einen Cortex-M0 benötigen.

11.9.6 Das ABI – Die Klassenbibliothek für die Cortex-Programmierung

Wenn Sie bereits mit einer höheren Programmiersprache gearbeitet haben, dann kennen Sie Bibliotheken. Das sind Sammlungen von Programmcode, den Sie für Ihre Programme nutzen können ohne sie selbst verfasst zu haben. Einen guten Programmierer zeichnet es aus, zunächst zu prüfen, ob eine Programmierlösung bereits in einer Bibliothek enthalten ist.

Bei ARM wird die entsprechende Sammlung von Funktionen aber auch von anderen hilfreichen Komponenten als ABI (Application Binary Interface) bezeichnet. Sie finden die vollständige Dokumentation des ABI unter <http://infocenter.arm.com/> im Bereich Documentation -> ARM Software Development Tools.

11.10 Wie Sie die Software in den Mikrocontroller bekommen

Zunächst müssen Sie sich voll und ganz darüber im Klaren sein, dass der Mikroprozessor bzw. die MCU, die Sie programmieren nicht Teil des Computers ist, mit dem Sie das Programm entwickeln. Das mag trivial klingen, die Konsequenzen sind es definitiv nicht!

11.10.1 Die Verbindung zwischen Entwicklungsboard und Rechner herstellen

Der erste Schritt besteht nun darin, dass Sie das Entwicklungsboard irgendwie mit Ihrem Rechner verbinden müssen. Und auch wenn auf dem Entwicklungsboard ein USB-Port vorhanden ist, können Sie (mit wenigen Ausnahmen) das Board damit nicht (!) mit dem Computer verbinden. Na gut, Sie können hier ein USB-Kabel anschließen, aber Sie werden Ihr Programm so nicht auf den Mikrocontroller übertragen können; der USB-Port dient in aller Regel ausschließlich dazu, das Entwicklerboard mit Strom zu versorgen.

Um ein Programm auf das Entwicklerboard zu übertragen (Fachbegriff: Flash programmieren), müssen Sie eine Schnittstelle nutzen, die beim Cortex-M0 JTAG heißt. Physikalisch handelt es sich dabei um einen Anschluss mit 14 Pins, der auf dem Entwicklerboard leicht erkennbar ist.

Aber Sie werden kein einfaches Kabel finden, mit dem Sie z.B. den USB-Port Ihres Rechners mit der JTAG-Schnittstelle verbinden können. Sie benötigen hierzu ein zusätzliches physikalisches Interface, für das es verschiedenen

Bezeichnungen gibt:

- ICE (In-Circuit Emulator) • In-Circuit Debugger • Debug probe • USB-JTAG-Adapter

Wenn Sie (wie in unseren Laboren) die Debugging Einheit ULink2 von Keil verwenden wollen, dann verbinden Sie sie zunächst mit dem USB-Anschluss Ihres Rechners und lassen den entsprechenden Treiber vom Betriebssystem Ihres Rechners installieren. Danach verbinden Sie den entsprechenden Anschluss des ULink2 mit dem JTAG-Port auf dem Entwicklerboard und verbinden abschließend das Entwicklerboard mit einem USB-Port Ihres Rechners, um die Stromversorgung sicher zu stellen.

Kontrolle

Sie wissen jetzt, was Sie benötigen, um ein Entwicklerboard mit einem Rechner zu verbinden, auf dem Sie Software für den Mikrocontroller entwickeln wollen.

Wie Sie den Quellcode in ein Programm umwandeln und dieses zum Mikrocontroller übertragen ist Inhalt der nächsten Abschnitte.

11.10.2 Vom Quellcode zum Programm

Wenn Sie Quellcode mit einer IDE wie dem MDK entwickelt haben, dann können Sie diesen mit dem Debugger auf Fehler kontrollieren und ihn recht komfortabel in ein sogenanntes Program Image umwandeln. Es ist dieses Program Image, das Sie anschließend in den Mikrocontroller übertragen bzw. in dessen Speicher Sie es flashen.

Ein solches Program Image beinhaltet die folgenden Abschnitte:

- Die Vektortabelle (So wird der Programmbereich genannt, der die Adressen des SP und des Exception Types enthält) • C-Startup Routine • Programm Code • C Library Code

Sie werden jetzt die Schritte kennen lernen, mittels derer aus Ihrem Quellcode ein solches Program Image erzeugt und auf den Mikroprozessor übertragen wird.

Bei der Entwicklung ist noch wichtig, dass Sie sich an bestimmte Namenskonventionen halten, denn nur durch diese zeigen Sie dem Computer, was der Quellcode ist: Dateinamen, die auf .c enden enthalten Quellcode in C, Dateien, die auf .cpp enden, enthalten Quellcode in C++ und Dateien, die

auf .s enden enthalten Quellcode in Assembler.

Für diejenigen unter Ihnen, die nicht viel über Dateiendungen wissen hier noch ein Hinweis: Sie können an einen Dateinamen eine beliebige Endung anhängen. Das führt aber nicht dazu, dass Sie den Inhalt der Datei ändern, sondern sorgt lediglich dafür, dass der Rechner davon ausgeht, dass es sich um eine entsprechende Datei handelt, so wie Sie einen Schatz an der Stelle erwarten, an der sich auf einer handgezeichneten Karte ein großes X befindet.

Beispiel: Wenn Sie an den Dateinamen die Endung .pdf anhängen und nun dem Betriebssystem z.B. durch einen Doppelklick angeben, dass es die Datei öffnen soll, dann wird es versuchen, die Datei mit dem Programm zu öffnen, mit dem Sie üblicherweise .pdf-Dateien öffnen. Nun wird dieses Programm allerdings mit dem Dateiinhalt nichts anfangen können, da Sie ja nur den Dateinamen geändert haben.

Sie wissen aus dem ersten Kapitel dieses Buches, dass ein Compiler Quellcode in ausführbaren Code umwandelt. Im Falle von C und Assembler passiert ein weiterer Schritt: Hier werden durch den Compiler aus dem Quellcode zunächst die sogenannten Object Files erzeugt. (Dateiendung .o) Diese sind zwar im Grunde bereits ausführbar, allerdings benötigen sie in aller Regel noch zusätzlichen Programmcode, der in Form sogenannter Linker Scripts (Dateiendung .ld) oder Scatter-Loading Files (Dateiendung .scat) eingebunden wird. Zusammen mit diesen Dateien wird nun aus den object files das Program Image erzeugt. Und diese Aufgabe übernimmt nicht mehr der Compiler, sondern der sogenannte Linker.

Wie Sie wissen kann ein Cortex-M0 bis zu 4 GB an Speicher verwalten, aber wie viel Speicher tatsächlich angebunden ist und wie dieser aufgeteilt ist, das muss im scatter-loading file angegeben werden. Beim Arbeiten mit dem MDK brauchen Sie sich darum nicht zu kümmern, wenn Sie im MDK eingetragen haben, welchen Mikrocontroller Sie verwenden. Denn dann nimmt das MDK die entsprechenden Einstellungen selbst vor und führt das Verlinken gemeinsam mit dem Kompilieren durch.

11.10.3 Program Image flashen

Nachdem Sie ein Program Image erzeugt haben, müssen Sie es noch in den Speicher des Mikrocontrollers übertragen. Da es sich bei diesem Speicher um Flash-Speicher handelt, redet man hier auch vom flashen. Hier gilt wieder: Ein einfaches Kopieren, wie Sie es z.B. bei USB-Sticks sonst gewöhnt sind, funktioniert nicht; Sie benötigen ein spezielles Programm, den Program Flasher. Wenn Sie das MDK nutzen (oder auch einige andere IDEs),

KAPITEL 11. EINFÜHRUNG IN DEN AUFBAU VON ARMV6-PROZESSOREN AM BEISPIEL EINER

brauchen Sie jedoch kein eigenständiges Programm zu installieren, da der Flasher dort bereits integriert ist.

Kontrolle

Sie wissen jetzt, wie Sie den Rechner und das Entwicklungsboard miteinander verbinden können und warum Sie später ein knappes Dutzend Dateien in dem Verzeichnis vorfinden werden, in dem Sie Ihren Quellcode sowie das Program Image speichern.

Sie kennen darüber hinaus die Dateiendungen, mit denen Sie es bei der Programmierung eines Mikrocontrollers zu tun haben und was diese Dateien beinhalten.

Außerdem wissen Sie, dass Sie das Program Image nicht einfach kopieren können, sondern ein Flash Programm benötigen, um es in den Speicher des Mikrocontrollers zu übertragen.

Zusammen mit den übrigen Inhalten dieses Kapitels kennen Sie damit alle Grundlagen, die Sie benötigen, um ein umfangreiches Programm für einen Cortex-M0 zu entwickeln. Damit können wir uns jetzt endlich der eigentlichen Programmierung zuwenden.

11.11 Zugriff auf die Peripherie bei einem Cortex-M0

Wie Sie bereits wissen erfolgt der Zugriff auf die Peripherie (also auch auf alle von Ihnen an den Cortex-M0 angeschlossenen Geräte), indem Sie die Memory-gemappten Register im Speicher des Cortex-M0 überschreiben.

Aber natürlich müssen Sie hierfür in Ihrem C-Programm noch einige Codezeilen einfügen, um diese Prozess zu vereinheitlichen. Im Buch von Yiu finden Sie dazu drei Codebeispiele für den Fall, dass Sie eine UART-Schnittstelle nutzen wollen:

Auf S. 62 finden Sie den Code, um ein Peripherieregister als einen Pointer zu definieren.

Auf S. 63 finden Sie den Code, um eine Datenstruktur für einen Satz von Peripherien zu definieren, in denen jede Peripherie als ein Pointer definiert ist.

Auf S. 64 finden Sie dann den Code, um gleich mehreren Instanzen Ihres Programms unabhängig voneinander den Zugriff auf eine solche Daten-

struktur zu ermöglichen.

11.12 Der CMSIS – Cortex Microcontroller Software Interface Standard

Sie kennen bislang die einfachen Grundlagen von C, mittels derer Sie einfache Programme entwickeln können. Wie Sie bereits bei der Erklärung zum ABI erfahren haben, benötigen Sie darüber hinaus noch weitergehende Programmiermöglichkeiten, um aus einem C-Programm heraus einen Cortex-M0 ausnutzen zu können.

Das CMSIS ist ein Framework, das es Ihnen ermöglicht, Cortex-Programme zu entwickeln, die Sie in unterschiedlichen IDEs weiterentwickeln können. Es wurde von ARM entwickelt und wird u.a. von den folgenden IDEs unterstützt: Keil MDK, ARM DS-5, IAR embedded Workbench, TASKING Compiler und verschiedenen GNU-GCCs.

Leider ist die Darstellung in Yius Buch veraltet: Er behandelt hier den CMSIS in der Version 1.3, während mittlerweile die Version 4.3 veröffentlicht wurde. Eine aktuelle Darstellung wird hier zu einem späteren Zeitpunkt erfolgen.

Kapitel 12

Grundlagen imperativer Programmierung

Liegt bereits vor, muss „nur“ noch in LaTeX konvertiert werden.

Teil III

Einführung in die objektorientierte Programmierung und Softwareentwicklung

Kapitel 13

Einführung in die Objektorientierung mit Java

Liegt bereits vor, muss „nur“ noch in LaTeX konvertiert werden.

13.1 Java 8 - Grundlegende Struktur

Bitte beachten Sie, dass alles, was Sie in diesem Buch zurzeit lesen können auf die Java-Version 8 bezieht. Zum Teil sind Abschnitte jedoch bewusst etwas allgemeiner gehalten, da die Feinheiten der Programmierung mit der Folgeversion geändert werden könnten. Das habe ich deshalb getan, um zu verhindern, dass Sie sich Methoden fest einprägen, die in einem oder zwei Jahren nicht mehr gelten. Außerdem sind Sie so gezwungen das zu tun, was bei Java oberste Pflicht ist: Nutzen Sie für die Details der Sprache die Java API! Ohne die Nutzung der Java API werden Sie auf eine Art programmieren, die früher oder später veraltet oder sogar ungültig ist.

Im Gegensatz zu Programmiersprachen wie C, PROLOG, Scheme und vielen anderen ist Java eine Middleware. Wie Sie also in Veranstaltungen zu Netzwerken oder Nachrichtentechnik lernen können ist Java von vornherein darauf ausgelegt, dass Sie Aspekte wie die Datenübertragung über ein Netzwerk nicht im gleichen Detailgrad programmieren müssen wie beispielsweise in einer Nicht-Middleware wie C. Die Vor- und Nachteile dieses Ansatzes sind Ihnen klar, wenn Sie beide Ansätze programmiert haben: Wenn Sie alle Details selbst programmieren, können Sie ein Höchstmaß an Effizienz erreichen, auch weil Sie die Spezifikation des jeweiligen Systems ausnutzen können. Wenn Sie eine Middleware nutzen, erhalten Sie eine in vielen Fällen noch ausreichend effiziente Lösung mit weniger Programmieraufwand. Außerdem helfen die in einer Middleware wie Java vorhandenen Standardisierungen, um Software in großen Teams zu entwickeln und neue Mitglieder in eine Team zu integrieren.

Als angehende (Medie-)informatikerInnen müssen Sie sich aber darüber bewusst sein, dass Sie in Java nicht die höchstmögliche Effizienz in diesem oder anderen Bereichen erreichen werden. Sie müssen sich insbesondere darüber bewusst sein, was das für ein in Java entwickeltes Programm bedeutet.

Wie Sie wissen, wenn Sie bereits in Java programmiert haben, handelt es sich um eine klassenbasierte objektorientierte und seit Version 8 zumindest teilweise funktionale Programmiersprache. In diesem Abschnitt werden wir uns die hierarchische Struktur von Java ansehen, die sich zum Teil identisch in der Hierarchie der Pakete und Klassen widerspiegelt.

Aufgabe

Was bedeutet es für diejenigen Ihre Programme, die Sie in Java erstellen, dass es nicht möglich ist, in Java ein Höchstmaß an Effizienz zu erreichen?

Aufgabe

Was ist in anderen Programmiersprachen nötig, um dort dieses Höchstmaß zu erreichen?

13.1.1 Die Java API

So wie Java kontinuierlich geändert wird, wird auch die API von Java kontinuierlich überarbeitet. Es ist deshalb nicht sinnvoll, die API herunterzuladen (außer wenn Sie davon ausgehen müssen, des öfteren keine Internetbindung zu haben, wenn Sie in Java programmieren wollen). Zurzeit finden Sie die Java API unter <http://docs.oracle.com/javase/8/docs/index.html>. Ältere APIs finden Sie, indem Sie schlicht die Ziffer im Link anpassen. Deshalb dürfte die API für Version 9 nach der Veröffentlichung unter <http://docs.oracle.com/javase/9/docs/index.html> zu finden sein.

Wenn Sie also in Bezug auf Java einen Link in diesem Buch finden, der nicht funktioniert, dann sollten Sie zumindest über den hier genannten Link zur entsprechenden Passage der API gelangen können.

13.1.2 JVM - Java Virtual Machine Technology

Das Fundament der Middleware Java besteht in der **JVM**, der Java Virtual Machine. Bislang haben Sie in diesem Buch lesen können, dass es die sogenannte JRE (Java Runtime Environment) ist, das dafür zuständig ist, Java-Programme zu betreiben. Die eigentliche Plattform, die Java-Programme betreibt ist jedoch die JVM, die aber Teil des JRE ist.

Eine **Virtuelle Maschine** ist ein Programm, das einen Rechner simuliert. Virtuelle Maschinen sind ein praktisches Werkzeug, wenn Sie beispielsweise Betriebssysteme und Anwendungen ausprobieren wollen, denn Sie benötigen für den Betrieb keinen zusätzlichen Rechner. Stellen Sie sich beispielsweise vor, Sie sind dafür zuständig, in einem Unternehmen die Software zu warten. Sie haben dann Nutzer mit den unterschiedlichsten Systemen und den unterschiedlichsten Installationen auf diesen Systemen. Um also Änderungen zu planen müssten Sie entweder von jedem dieser Systeme ein Exemplar in Ihrem Büro haben oder Sie nutzen eine VM, für die Sie all die verschiedenen Systeme als individuelle Datei vorliegen haben. Das genügt zwar nicht für eine vollständige Prüfung darauf, ob Änderungen reibungslos durchführbar sind, aber es ist eine sinnvolle erste Prüfung.

Die JVM sorgt also dafür, dass Java Programme auf einer Vielzahl von Systemen laufen können.

Wenn sie bereits Java-Programme entwickelt haben, dann wissen Sie, dass Sie Programme nach dem Kompilieren mit dem Befehl `java` starten können. Das bedeutet aber genauer gesagt, dass Sie mit dem Befehl `java` die JVM starten und ihr ein kompiliertes Java-Programm als Argument übergeben.

Sehen wir uns nun den Befehl `java` etwas genauer an: Es gibt ihn in den zwei Versionen **java** und **javaw**.

- **java** startet nicht nur die JVM und führt über diese das übergebene Java-Programm aus, sondern öffnet zusätzlich ein Konsolenfenster. Dieser Befehl ist also für die Entwicklung von Java-Programmen gut, weil Sie hier Rückmeldungen erhalten, die innerhalb der Anwendung nicht angezeigt werden.
- **javaw** öffnet im Gegensatz zum Befehl `java` keine Konsole. Sollte es allerdings zu Fehlern kommen, wegen denen die Anwendung nicht gestartet werden kann, dann öffnet sich ein Fenster, das eine Fehlermeldung anzeigt.

Nach `java` bzw. `javaw` können Optionen folgen, die die Ausführung der Anwendung beeinflussen. beispielsweise können Sie die JVM anweisen, die Anwendung so schnell wie möglich auszuführen. Eine andere Option ermöglicht es, dass die JVM so wenig Speicher wie möglich belegt.

Nun folgt entweder der Name der Klasse oder die Option **-jar** gefolgt vom Dateinamen. Bei der Option **-jar** folgt also keine einzelne Klasse. Vielmehr haben Sie es hier mit einer Verwendung von `java` bzw. `javaw` zu tun, die Sie kennen lernen werden, wenn Sie in Netzen ein Deployment durchführen. Damit ist gemeint, dass Sie z.B. in einem Unternehmensnetzwerk ein Paket verteilen lassen und es auf mehreren Rechnern ausführen lassen, ohne an jeden einzelnen Rechner zu gehen. Das können beispielsweise Sicherheitsupdates sein, aber denken Sie jetzt nicht, dass nur bei Sicherheitsupdates ein Deployment Sinn macht: Jede Art von Installation innerhalb eines Unternehmensnetzwerkes bzw. bei verteilten Anwendungen sollte weitgehend automatisch auf allen betroffenen Systemen durchgeführt werden. Abschließend können Sie noch ein oder mehrere Argumente eingeben, die als String-Array an die `main()`-Methode der Java-Klasse übergeben wird, die durch die `java`- bzw. `javaw`-Anweisung gestartet wird. (Auch bei den Paketen, die per **-jar** gestartet werden gibt es eine Java-Klasse, die als erstes gestartet wird.)

Mehr zur JVM, zu `java` und `javaw` finden Sie bei Java 8 unter <http://docs.oracle.com/javase/8/docs/technotes/guides/vm/index.html>. ■

13.1.3 Base Libraries

Die ersten Bibliotheken von Java werden von Oracle als **Base Libraries** bezeichnet. Diese Base Libraries sind wie alle Bibliotheken von Programmiersprachen nichts anderes als Sammlungen von Programmteilen, die bestimmte Aufgaben erfüllen.

Diese Bibliotheken haben als einzige Gemeinsamkeit, dass Sie den eigentlichen Sprachkern ausmachen. Klassen aus den Base Libraries dienen also niemals dazu, um die Anzeige Ihrer Anwendung zu ändern. Sie bieten keinerlei Unterstützung bei der Interaktion mit Nutzern. (Einzig die Eingabe und Ausgabe von Zeichenfolgen ist hier enthalten.) Auch die Audio- und Videoverarbeitung (von Eingabe oder Ausgabe ganz zu schweigen) werden mit diesen Bibliotheken nicht realisiert.

Über die einzelnen Bibliotheken werden wir sprechen, wenn wir uns die übrigen Bestandteile des JRE bzw. des JDK angesehen haben.

13.1.4 Integration Libraries

Es folgen eine handvoll Bibliotheken, mit denen Sie als fortgeschrittene Java-Entwickler zu tun bekommen: Die Integrationsbibliotheken stellen z.B. Schnittstellen zu Datenbanken her, bei nur mittels SQL-Sprachen angesprochen werden können. Hier finden Sie auch die RMI-Bibliothek, mittels derer Sie auf Java-Programmteile zugreifen können, die auf einem anderen Rechner laufen. Auch wenn Sie mit diesen Bibliotheken anfangs nichts zu tun haben stellen sie also sehr nützliche Möglichkeiten zur Verfügung.

13.1.5 User Interface Libraries

Jetzt kommen wir zu den Bibliotheken, mit denen die meisten von Ihnen die Programmierung beginnen wollen: Die UIs stellen Ihnen eine Vielzahl an Klassen zur Verfügung, mittels derer Sie z.B. Ein- und Ausgabe von Sounds, die Anzeige einer grafischen Nutzeroberfläche und vielem anderen mehr realisieren können, das für aktuelle Programme unverzichtbar ist. Wissenschaftler und Systemadministratoren mögen durchaus auf solchen „Spielkram“ verzichten, aber für nahezu alle anderen Kundengruppen gilt: Gibt es keine grafische Nutzeroberfläche mit Soundfeedback und Mausinteraktion, dann bleibt's ein Ladenhüter.

13.1.6 Deployment

Dieser Begriff wurde oben schon eingeführt. Es mag deshalb verwirren, dass JAR-Dateien nicht in diese Gruppe von Bibliotheken gehören, aber der Grund ist recht simpel: Sie können JAR-Dateien auch ohne Deployment verwenden. Die Bibliotheken des Deployment-Pakets bieten Ihnen die Möglichkeit an, von einzelnen Patches für eine Java-Anwendung bis

hin zu einer vollständigen Java-Anwendung alle denkbaren Softwarepakete automatisiert zusammenstellen, zu komprimieren und an beliebig viele Zielrechner über ein Netzwerk zu verschicken, wo sie dann nach Vorgaben ebenso automatisiert installiert werden können. Der Vollständigkeit halber sei noch erwähnt: Natürlich muss dazu am jeweiligen Rechner nach der Installation des Betriebssystems einmalig eine entsprechende Konfiguration durchgeführt werden, aber das ist auch alles, was an jedem Zielsystem getan werden muss.

13.1.7 Weitere Pakete im JDK

Es folgen nur noch Pakete, die Teil des JDK, aber nicht des JRE sind. Wenn Sie sich die entsprechende Übersicht in der Java API, genauer gesagt in den Java Docs ansehen, dann werden Sie feststellen, dass dort der Befehl `java` als ein Teil des JDK, genauer des Pakets „Tools & Tool APIs“ aufgeführt werden. Der Grund ist recht simpel: Um als Entwickler die verschiedenen Bestandteile der Middleware aktiv einsetzen zu können, sind teilweise Anpassungen nötig.

Das ist die grobe Einteilung von Java in Paket- oder Bibliotheksgruppen. Sehen wir uns jetzt die einzelnen Gruppen im Detail an.

13.2 Base Libraries im Detail

Vielleicht fragen Sie sich, warum wir nicht zunächst einen Blick auf die VM werfen: Alles was hier relevant ist wurde bereits oben erklärt. Für die weiteren Details in der aktuellen Java-Version ist es wichtig, dass Sie einen Blick in die Java API werfen.

In den folgenden Abschnitten finden Sie grundlegende Informationen dazu, was im jeweiligen package zu finden ist, bzw. warum es dieses package gibt. Beachten Sie dabei, dass viele dieser Pakete wiederum in Pakete aufgeteilt sind.

13.2.1 `lang`

Die offizielle Erklärung dieses Paketes ist etwas schwammig. Demnach handelt es sich um essentielle Funktionalitäten der Sprache. Für die Frage, was dieses Paket nützt ist diese Antwort nicht eben hilfreich. Da hätte die Auskunft „`lang` ist wichtig,“ genauso viel oder wenig ausgedrückt.

Einige Klassen dieses Pakets dienen als sogenannte Wrapper Klassen, um Variablen, die in Java ja keine eigenständigen Objekte sind, als Objekte zu verpacken.

Einige Klassen dieses Packets stellen sicher, dass in Java „Dinge“, wie `Class`, `Object`, `Throwable`, `Thread` und ähnliche mehr existieren, die wir bei der Programmierung wie selbstverständlich nutzen.

Sollten Sie sich gefragt haben, warum Sie Klassen ohne Konstruktor programmieren können, können Sie in diesem Teil der API die Antwort nachlesen: Klassen vom Typ `Class` werden automatisch durch die JVM mittels eines Aufrufs der Methode `defineClass(...)` erzeugt.

Dazu stellt es auch die Möglichkeiten sicher, um entsprechend der Hierarchie von Threads Vererbung zu realisieren.

Dann gibt es hier Klassen, die dafür sorgen, dass Prozesse (also die tatsächliche Ausführung von Programmteilen) durch das Betriebssystem durchgeführt werden.

Verkürzt ließe sich also sagen, dass `java.lang` all das beinhaltet, was nötig ist, um einfachste Java-Klassen zu programmieren, aber keine Variablen im Sinne virtueller Objekte, die direkt im Speicher abgelegt werden können und den auf ihnen definierten Operationen.

Aufgabe

Werfen Sie einen Blick in die Java-API des package `java.lang` und lösen Sie anhand der dortigen Angaben die folgenden Aufgaben:

1. Was passiert, wenn ein NaN vom Typ `Float` mit einem Objekt vom Typ `Float` verglichen wird, der eine Zahl enthält?
2. Neben dem Umgang mit NaN gibt es noch etwas überraschendes, wenn es um den Vergleich mit Zahlen geht. Es geht hier um die Zahl 0.0f. Was ist das?
3. Es gibt die Methode `sum()`, um zwei `Float`-Objekte zu addieren. Wie sieht es mit anderen arithmetischen Operationen aus?
4. Suchen Sie weiter und finden Sie alle Methoden, mittels derer Sie Operationen durchführen können, die für Variablen vom Typ `float` definiert sind.
5. Welchen Sinn kann es haben, einzelne Werte als `Float`-Objekte (oder andere Variablen z.B. boolesche Variablen) als Objekte vom Typ der entsprechenden Wrapper Klasse zu verpacken?
6. Finden Sie (anhand der Java API zu `Float`) heraus, was der größte und kleinste Wert ist, den eine `float` Variable haben kann. Geben Sie ihn als gerundete 10-er Potenz an. Was fällt Ihnen auf? (Bitte hier die beiden Felder `NEGATIVE_INFINITY` und `POSITIVE_INFINITY` ignorieren.)
7. Und jetzt das gleiche für ein Objekt vom Typ `Double`.
8. Berechnen Sie, wie viel Speicher eine Variable vom Typ `float` bzw. `double` mindestens belegt.

9. Erzeugen Sie jetzt ein Objekt von jedem der beiden Typen, jeweils mit einem maximalen Wert und prüfen Sie, wie viel Speicher tatsächlich verwendet werden.
10. Finden Sie anhand der API (egal ob für Float oder Double) heraus, wie eine Zahl in einen Text umgewandelt werden kann, der z.B. für `System.out.println()` nötig ist.
11. Sie wollen ein Programm entwickeln, bei dem es auf außerordentliche Präzision bei mathematischen Operationen ankommt. Sie müssen also beispielsweise wissen, wenn es zu einem Overflow bei einer Operation kommt. (Denken Sie hier an ein Programm, dass ein Raumschiff auf dem Flug zum Mars steuert oder an die Steuerung eines Kernkraftwerks. Für beide Fälle würden Sie in der Praxis auf andere Programmiersprachen zurück greifen, aber hier geht es darum, dass Sie die Möglichkeiten von Java kennen lernen.) Prüfen Sie, welche der beiden Klassen `Math` und `StrictMath` Sie in diesem Fall nutzen würden. Beschreiben Sie, wie die Klasse Ihnen eine höhere Sicherheit gewährleistet, bzw. welche Methoden das tun. Erklären Sie außerdem, warum es dann noch die andere Klasse gibt.
12. Sie haben gelernt, dass Sie Objekte vom Typ `String` zwar überschreiben können, aber dass Sie mit ihr nicht umgehen können, als wäre es ein `char[]` (Array von `char`-Variablen). Schauen Sie sich nun die Methode `delete()` der Klasse `StringBuffer` an. Wie wird die Datenstruktur dieser Klasse allgemein genannt?

Mit diesen Übungen haben Sie ein vertieftes Verständnis für das Package `java.lang` und damit für die Entwicklung grundlegender Java-Anwendungen gewonnen. Insbesondere können Sie jetzt mithilfe der Wrapper Klassen Aufgaben lösen, für die Sie früher eigene Klassen entworfen hätten. Dazu kommt, dass Sie je nach nötiger Präzision von mathematischen Aufgaben die passenden Methoden verwenden werden. Und Sie werden nie wieder eine eigene Variable für die Eulersche Zahl oder `Pi` verwenden.

Wird fortgesetzt.

Kapitel 14

Java 8 Basics

Mit dem Begriff Basics werden häufig einfache Grundlagen oder Einsteigerwissen bezeichnet. Hier bauen wir also die Grundlagen aus, die Sie durch das Kapitel über imperative Programmierung und objektorientierte Softwareentwicklung bereits erlernt haben.

Im Klartext: Das was jetzt kommt mag vielleicht in einzelnen anderen Sprachen ebenfalls vorkommen, aber eben längst nicht in den meisten anderen imperativen klassenbasiert objektorientierten Sprachen.

In diesen Basics werden wir uns auf den Kern der Programmierung kümmern, die Frage, wie wir Algorithmen in Quellcode umwandeln. Wir werden uns dagegen nicht mit Themen beschäftigen wie der Programmierung einer grafischen Benutzeroberfläche, also den sogenannten „Fenstern“, der Datenübertragung über Netze, Zugriff auf Dateien (wobei diese beiden Punkte nahezu identische Programmier Techniken verwenden) und andere Aufgaben für die eine Middleware wie Java prädestiniert ist. Der Grund ist schlicht der, dass wir erstmal sinnvoll imperativ programmieren müssen, bevor wir die implementierten Algorithmen so verpacken, dass sie Daten mit anderen Rechnern austauschen oder komfortabel von Menschen bedient werden können. Eingaben und Ausgaben per Tastatur werden aber Teil der Basics werden, wir verzichten hier also nicht vollständig auf Interaktion, sondern fangen direkt damit an.

14.1 Eingaben und Ausgaben

Java-Programme setzen sich aus Teilen zusammen, die als Klassen bezeichnet werden. Mit einer Klasse ist noch kein Objekt im Sinne der objektorientierten Softwareentwicklung gemeint. Vielmehr ist eine solche Klasse ein Programmteil, der als Vorlage für beliebig viele Objekte im Sinne der objektorientierten Softwareentwicklung verwendet werden können. Jede Klasse

eines Pakets hat einen Bezeichner.

14.1.1 Erzeugen eines Objekts

Bei der Programmierung erzeugen wir diese Objekte fast genau so, wie wir das in der imperativen Programmierung in statisch typisierten Sprachen mit Variablen getan haben:

```
Klassenbezeichner objektbezeichner  
= new Klassenbezeichner();
```

Schauen wir uns jeden Teil dieser Zeile an:

- **Klassenbezeichner**

ist der Bezeichner der Klasse. Stellen Sie sich das ruhig wie den Datentyp einer Variablen vor. Im Detail ist es zwar wesentlich mehr, aber für die Programmierung eines neuen Objekts genügt das.

- **objektbezeichner**

ist dann der Bezeichner für das Objekt. Sie wundern sich, warum hier zumindest scheinbar die gleichen Regeln gelten wie bei Bezeichnern von Variablen? Dann denken Sie nochmal daran zurück, was Sie über Variablen gelernt haben: Es sind Bezeichner, die auf einen Bereich im Speicher des Computers verweisen, dessen Inhalt sich ändern darf. Und Objekte sind zwar deutlich umfangreicher als einzelne Variablen, aber aus der Perspektive von Java-Entwicklern sind Sie einfach nur die nächste komfortable Zusammenfassung von virtuellen Objekten, nachdem wir einzelne Variablen in einer Datenstruktur zusammengefasst haben.

- **=**

kennen Sie bereits: Es ist der altbekannte Zuweisungsoperator.

- **new**

ist tatsächlich etwas neues, das wir bislang noch nicht besprochen haben. Es ist eine Operation, die der „Programmiersprache“ anzeigt, dass es sich bei dem zugeordneten Wert nicht um eine Variable oder eine Datenstruktur, sondern um ein Objekt handelt. Das mag etwas ungewohnt sein, da wir in der imperativen Programmierung einfach Werte eingegeben und einer Variablen zugeordnet haben, aber wenn Sie erst einige Klassen erzeugt haben, werden Sie gar nicht mehr darüber nachdenken.

- **Klassenbezeichner()**

Auf den ersten Blick sieht das wie eine Funktion aus, denn es endet ja mit einem Klammernpaar. Und vollkommen richtig: Das ist eine Funktion, allerdings eine ganz spezielle, die für die klassenbasierte Objektorientierung in Java eine Rolle spielt. Es ist der sogenannte **Konstruktor**.

Der Unterschied zu den Funktionen, die Sie bislang kennen besteht darin, dass der Konstruktor keine Variable oder Datenstruktur zurückgibt, sondern ein Objekt.

Der Konstruktor hat dabei immer den selben Bezeichner wie die Klasse, von der er ein Objekt erzeugt (Fachterminus **instanziert**).

Um Konstrukturen eindeutig von anderen Funktionen zu unterscheiden beginnen sie in Java mit einem Großbuchstaben.

Für den Moment gehen wir hier noch nicht in die Details, sondern Sie sollten sich lediglich merken, dass bis auf zwei Ausnahmen alle Klassen einen Konstruktor enthalten müssen. Denn im Konstruktor programmieren wir, wie ein Objekt einer Klasse erzeugt wird.

- ;

Das Semikolon zeigt das Ende einer Programmzeile an.

14.1.2 Benutzen von Objekten innerhalb eines Java-Programms

Variante a: Objekte analog zu Variablen verwenden

Wie Sie oben gelernt haben können wir ein Objekt in Java ähnlich einer Variablen oder Datenstruktur nutzen, nachdem wir es einem Bezeichner zugeordnet haben. Sie können hier ebenfalls Arrays und andere Datenstrukturen vom Typ einer Klasse erzeugen und brauchen in dieser Hinsicht nichts neues zu lernen.

Variante b: Objekte der Programmiersprache nutzen

Es gibt aber auch Objekte, die bereits Teil der Programmiersprache sind. Manche können wir ohne weitere Vorbereitung verwenden, andere müssen wir zuerst über die **import**-Anweisung in unser Programm integrieren. Zur import-Anweisung später mehr.

Manche dieser Objekte beginnen wie Klassennamen mit einem Großbuchstaben, manche mit einem Kleinbuchstaben. Da die Erklärung dazu verwirrend ist, wenn Sie sich noch nicht umfassend mit dem Konzept der Vererbung bei objektorientierter Softwareentwicklung beschäftigt haben, lasse

ich sie an dieser Stelle außen vor. Sie können aus genau diesem Grund den folgenden Programmcode noch nicht vollständig verstehen, obwohl es sich um das übliche Einstiegbeispiel in Programmiersprachen handelt.

14.1.3 Hello, world.

```
public class HelloWorld{
    public static void main(String[] args){
        System.out.println("Hello, world!");
    }
}
```

Wie immer reden wir hier zunächst nicht von einem Programm, sondern von einer Klasse. Sie haben oben gelesen, dass es zwei Arten von Klassen gibt, die keinen Konstruktor benötigen. Und hier haben Sie es genau mit solch einem Fall zu tun. Der Grund ist recht einfach: Unser Programm soll einzig und alleine die Zeile *Hello, world!* auf der Konsole ausgeben. Wir brauchen nicht nur keine beliebige Anzahl Objekte, damit unser Programm funktioniert, sondern vielmehr ist das Objekt-Konzept hier vollkommen überflüssig. Und netterweise nötigt Java uns nicht dazu, trotzdem einen Konstruktor zu programmieren.

Streng genommen wird im Hintergrund dennoch ein „Standard-“ Konstruktor von Java erzeugt, aber das kann uns an dieser Stelle egal sein...

Schauen wir uns nun den Code im Detail an:

- `public class HelloWorld{`

- **public**

Wenn Sie sich an die Erklärungen zu virtuellen Objekten erinnern, dann erinnern Sie sich an die **Zugriffsberechtigung**. Damit ist der Teil eines virtuellen Objekts gemeint, der definiert, wer bzw. welche Programmteile Zugriff auf das Objekt haben. In Java gibt es vier unterschiedliche **Zugriffsmodifikatoren**, wobei einer keinen Bezeichner hat. `public` ist einer dieser Modifikatoren und er gewährt quasi freien Zugriff für alle. (Das ist natürlich wieder nicht vollkommen richtig; die Details schauen wir uns an, wenn wir uns mit den Zugriffsmodifikatoren beschäftigen.)

- **class**

Dieses Schlüsselwort gibt schlicht an, dass nun eine Klasse definiert wird. Es wird Ihnen einige Zeit lang unsinnig vorkommen, dass wir eine Klasse mit diesem Schlüsselwort markieren

müssen, weil wir in Java keine Funktionen oder Variablen außerhalb von Klassen definieren bzw. nutzen dürfen. Das umso mehr, weil Sie anfangs ausschließlich Klassen programmieren, in denen Variablen und Methoden (so nennen wir Funktionen in objektorientierten Sprachen) vorkommen. Aber das wird sich später ändern: Dann werden Sie parallel zu Methoden auch Klassen innerhalb von Klassen definieren. Also ist es essentiell, dass es dieses Schlüsselwort gibt, und dass wir es immer benutzen, wenn wir eine Klasse programmieren.

– **HelloWorld**

Das ist der Bezeichner unserer Klasse. In Java ist es eine Konvention, dass jede Klasse in einer eigenen Datei gespeichert werden muss, die als Dateibezeichner Klassenbezeichner.java hat. Unsere Klasse HelloWorld muss also in einer Datei mit dem Bezeichner HelloWorld.java gespeichert werden.

– {

Das ist in Java das Symbol, mit dem wir den Anfang eines Rumpfes hervorheben. Alles, was nun folgt (bis zum entsprechenden `}`) ist also Teil unserer Klasse HelloWorld.

• **public static void main(String[] args)**

– **public**

kennen Sie jetzt schon. Da es in diesem Fall vor einer Methode steht, bedeutet das, dass alle anderen Klassen diese Methode benutzen dürfen. Richtig verstanden: Durch einen passenden Zugriffsmodifikator können wir unterbinden, dass andere Objekte auf einzelne Methoden unseres Objekts bzw. unserer Klasse zugreifen können.

– **static**

Das ist wieder etwas neues, aber im Grunde nicht weiter kompliziert. Damit zeigen wir nämlich an, dass etwas nicht geändert werden darf. Der Unterschied gegenüber statischen Variablen (also Konstanten) hat etwas mit dem Konzept der Vererbung in objektorientierter Softwareentwicklung zu tun: Hier können nämlich Teile eines Programms (also Klassen) den Inhalt anderer Teile des Programms (also anderer Klassen) übernehmen und beliebig abändern. Wenn dagegen in der Klasse ein solcher Teil als static definiert wurde, aus der ein Teil übernommen wird, dann darf er nicht mehr geändert werden.

– **void**

Damit wird definiert, dass diese Methode keinen Rückgabewert hat.

– **main()**

Das kennen Sie schon aus der imperativen Programmierung. Es gibt nur einen kleinen aber sehr wichtigen Teil, auf den wir eingehen müssen: Wenn Sie ein Java-Programm „starten“, dann „starten“ Sie streng genommen nicht das Programm als ganzes, sondern Sie rufen die `main()`-Methode einer Klasse dieses Programms auf.

Und wenn Sie dabei Argumente übergeben wollen, so wie Sie das bei einem anderen Funktionsaufruf tun würden, dann tun Sie das, indem Sie nach dem Namen der Klasse nach dem „Startbefehl“ schlicht die Argumente durch Leerstellen getrennt eingeben. Dazu folgt im Anschluss ein Beispiel. Schauen wir uns vorher den Rest unseres kleinen Beispiels an.

– {

Wie gewohnt beginnt damit der Rumpf der Methode.

• **`System.out.println("Hello, world!");`**

Hier sind wir an genau der Stelle, von der ich vorhin sprach: Um diese Zeile wirklich zu verstehen brauchen Sie ein fortgeschrittenes Verständnis für die Umsetzung des Konzepts der Vererbung in objektorientierter Softwareentwicklung. Außerdem müssten Sie etwas darüber wissen, wie Java als Middleware den Zugriff auf die Konsole, auf Dateien und andere Bereiche außerhalb des Programms ermöglicht. Das zu erklären wäre aber an dieser Stelle bei weitem zu umfangreich und würde eher zu Verwirrung führen.

Sie kennen einerseits Methoden (Funktionen in objektorientierter Softwareentwicklung) und andererseits kennen Sie Konstruktoren. Dann kennen Sie Klassen und Objekte. Von daher muss diese Zeile Sie verwirren: Da ist die Java-Klasse *System*, die offensichtlich aus der Klassenbibliothek stammt mit einem Punkt mit einem Objekt namens *out* verknüpft, das dann wieder über einen Punkt mit der Funktion *println()* verknüpft ist.

Wie gesagt erfordert es einiges an Hintergrundwissen, um das tatsächlich zu verstehen. Merken Sie sich deshalb bitte für den Moment folgende zwei Dinge:

(1) Wenn Sie eine Textausgabe auf der Konsole erzeugen wollen, dann genügt dafür leider nicht mehr ein einfaches `print()` oder `printf()`, wie Sie das z.B. aus C kennen, sondern Sie müssen dieses komplette `System.out.println()` verwenden. `System.out.print()` funktioniert ebenfalls. Der Unterschied besteht darin, dass `System.out.println()` einen Zeilenumbruch erzeugt.

(2) Nein, es ist nicht leicht, ohne entsprechendes Hintergrundwissen zu verstehen, was da passiert. Wenn wir uns mit dem Zugriff auf Dateien im Rechner oder auf Webpages beschäftigen nutzen wir genau die gleichen Grundlagen, die bei `System.out.println()` wichtig sind. Und der Grund aus dem viele Studierende an diesen fortgeschrittenen Themen scheitern ist die schlichte Tatsache, dass sie denken, `System.out.println()` „gibt doch bloß eine Zeile Text aus... Was soll daran schwer sein?“ Wenn wir dann später zu den genannten Themen kommen und sie hören, dass dort die gleichen „Dinge“ passieren wie bei der Textausgabe, schalten sie ab, weil sie denken, dass das ja ganz simpel sein müsse. Aber das ist es definitiv nicht.

Zusammenfassung

Um in Java auch nur eine einfache Textzeile auszugeben müssen wir eine Vielzahl an Details programmieren, die nur wegen der klassenbasierten objektorientierten Natur von Java nötig sind.

Setzen Sie sich mit den oben genannten Dingen intensiv auseinander, denn die sind bei der Java-Programmierung Ihr täglich Brot. Und ich meine wirklich jedes einzelne Detail.

14.1.4 Hello, (Name des Benutzers)

Kommen wir nun zu einer Variante von Hello, world! bei der jeder Nutzer individuell angesprochen werden kann:

```
public class HelloWorld{
    public static void main(String[] args){
        System.out.println("Hello, " + args[0] + ".");
    }
}
```

Mit dem, was Sie bislang gelernt haben sollten Sie vollständig im Stande sein, die Änderungen zu verstehen. Wenn Sie diese Datei unter der Bezeichnung `HelloWorld.java` gespeichert haben, sie anschließend auf der Konsole mit **javac HelloWorld** kompilieren und dann per **java HelloWorld Horst** starten, dann wird der Rechner Sie mit einem freundlichen **Hello, Horst.** begrüßen.

Aufgaben:

- Ändern Sie das Programm so ab, dass es nacheinander mehrere Nutzer begrüßen kann, auch wenn die Anzahl Nutzer im Vorfeld nicht feststeht. Sie sollen hier also eine Schleife programmieren, die über die Länge des Arrays `args` läuft.

Hinweis: Die Länge eines Array erhalten Sie per `arrayBezeichner.length`.
Im Falle von `args` also per `args.length`. Ähnlich wie bei `System.out.println()`
können Sie mit dem aktuellen Wissen noch nicht verstehen, was genau diese Verknüpfung `.length` bewirkt.)

Stichwortverzeichnis

- agil, 42
- Algorithmen und Datenstrukturen, 12, 42
- Algorithmen design, 12, 13, 42
- Algorithmik, 12, 18
- Algorithmus, 22
 - online, 13
- Android, 97
- Anker, 152
- Anwendung, 29
 - verteilt, 75, 81, 82
 - Webanwendung, 82, 83
- API, 66
- App
 - Entwicklung, 29
- ASCII, 49
- backend, 36
- Betriebssystem, 75
- Bibliothek, 26, 80
- Big Data, 13
- Bittigkeit, 56
- Blender, 80
- Bus, 52
- Client, 81, 105
- Codierung, 49, 119, 138
 - ASCII, 49
 - Hamming, R.W., 50
- Compiler, 60
- Computerprogramm, 22
- Container, 126
- Continuous Deployment, 41
- Continuous Integration, 41
- CSS, 123
 - Property, 208
- Selektor, 208
- Datenbank, 85, 111, 226
 - Datensatz, 85
 - MySQL, 83
 - relational, 85
- Datensatz, 85
- Datenschutzes, 13
- Datenstruktur, 226
- Datentyp, 50, 52, 86
 - Integer, 86
- Delta, 28, 61
- Deployment
 - support, 61
 - tools, 61
- Design Pattern, 42
- Dienst, 106
- DNS, 106
- Doctype, 132
- Doctype Definition, 133
- Dokumentation, 27
- dynamisch, 111
- Elektrotechnik, 24, 48
- Endgeräte
 - mobil, 31
 - Wearables, 31
- entryFourier-Transformation, 48
- ERP, 44
- Exception, 87
- Extreme Programming, 42
- first-class Object, 91
- Flash, 110
- formal table, 202
- Formulare, 156
- FPGA, 54

- Framework, 26
 - AJAX, 39
 - AngularJS, 39
 - Backbone, 39
 - EmberJS, 39
 - Grunt, 39
 - Gulp, 39
 - jQuery, 39
 - LESS, 39
 - requireJS, 39
 - Ruby on Rails, 83, 84
 - SASS, 39
 - Twig, 39
 - Zend, 40
- Frontend, 40
- frontend, 36
- FSF, 64
- FTP, 120
- Funktion, 90, 225
- Game Engines, 80
- Games
 - Blender, 80
 - Browsergames, 32
 - Game Engines, 80
 - MMORPG, 32
 - Steam, 79
- general purpose programming, 75
- GI, 64
- Git, 28, 71
- Hardware, 23
- HTML, 83, 110, 123, 152
 - Attribut
 - action, 157
 - allowfullscreen, 175
 - alt, 170
 - autoplay, 174
 - controls, 172
 - datetime, 180
 - disabled, 166
 - height, 170
 - loop, 174
 - method, 157
 - name, 158
 - poster, 174
 - preload, 174
 - width, 170
 - Attribute
 - itemprop, 187
 - itemtype, 187
 - Attrubute
 - itemscope, 185
 - Canvas, 170
 - Container, 126
 - audio, 175
 - dd, 184
 - details, 178
 - dl, 184
 - dt, 184
 - fieldset, 168
 - figcaption, 171
 - figure, 171
 - form, 157
 - iframe, 175
 - img, 170
 - legend, 168
 - li, 183
 - ol, 183
 - optgroup, 166
 - option, 166
 - select, 166
 - summary, 178
 - table, 184
 - td, 184
 - th, 184
 - time, 179
 - tr, 184
 - ul, 183
 - video, 172
 - Description List, 184
 - Doctype, 132
 - Doctype Definition, 133
 - ordered list, 183
 - Tabelle, 184
 - unordered list, 183
- HTML5, 84
- HTTP, 97, 119

- HTTPS, 120
- Hyperlink, 119, 152
- Hypertext, 119, 151, 152
- id, 153
- IDE, 26, 61
- IDEs
 - Eclipse, 68
 - Visual Studio, 63
 - XCode, 63
- IEEE, 64
- Informatik, 12, 21, 24, 51, 52, 76, 85, 86, 89
 - Praktische Inf., 10, 21, 28, 33, 42, 89
 - Technische Inf., 10, 14, 24, 34, 50, 52, 53, 56, 86
 - Theoretische Inf., 33
- instanziert, 310
- Interface, 49
- Internationalisierung, 138
- Internet, 95
- Interpreter, 60
- iOS, 97
- IP
 - IP-Adresse, 105
 - IPv4, 107
 - IPv6, 107
- ISO 10646, 141
- IT-Sicherheit, 60
- ITU-T, 64
- JavaScript, 148
- Klasse, 66, 77
- Klassenbibliothek, 66
- Klausel, 25
- Kommunikationstechnik, 52, 95, 106
- Kommunikationswissenschaften, 53
- Kompression, 173
- Konsole, 67
- Konstruktor, 310
- Kontrollstruktur, 225
- Lambda-Kalkül, 90
- LaTeX
 - formal table, 202
 - Inhaltsverzeichnis, 198
 - Präambel, 193
 - Referenz, 206
 - tabular matter, 202
 - Umgebung, 198
- Linux, 28, 97
- localhost, 105
- Logistik, 14
- Lokalisierung, 138
- machine-readable content, 185
- Markup Language, 83, 95, 123
- Mathematik, 54, 55
 - boolesche Algebra, 54
 - Fourier-Transformation, 48
- MDK, 62
- Media Systems, 15, 17, 48
- Mediendesign, 45
- Medientechnik, 14, 17, 49
- Message Sending, 77
- Meta-Daten, 138
- Microdata, 38, 185
- Middleware, 26
- Mikroprozessor, 52
- MinGW, 63
- Mnemonic, 58
- MVC, 42
 - View, 207
- MySQL, 83
- Nachrichtentechnik, 24, 48, 51, 53, 76, 81, 95, 106, 119
- Netzwerk, 81
- Objektorientierung, 22, 79
 - Klasse, 77
 - Message Sending, 77
 - Portabilität, 79
- Operation, 225
- Paradigma, 77
- Patch, 41
- PHP, 83

- Funktion, 225
- Polyfill, 147
- Portabilität, 79
- Präambel, 193
- Prämisse, 24
- Programmieren, 21
 - Paradigma, 20, 86
- Programmierschnittstelle, 66
- Programmiersprache
 - ActionScript, 110
 - Assembler, 57
 - C, 57, 75, 86, 88
 - C++, 57
 - CSS, 110, 123
 - Erlang, 77
 - Flash, 176
 - HTML, 70, 83, 110, 123, 152
 - HTML5, 84
 - Java, 57, 77, 78, 86, 176
 - JavaScript, 82, 84, 148, 176
 - Maschinensprache, 56
 - MySQL, 71, 83, 112
 - PHP, 70, 82, 83, 111
 - PROLOG, 75
 - Python, 86, 89
 - Ruby, 83, 86
 - Ruby on Rails, 83, 84
 - UML, 96
 - XHTML, 131
 - XML, 131
 - Zweck einer Sprache, 79
- Programmiersprachen
 - .NET, 37
 - ActionScript, 40
 - AS, 40
 - C, 37
 - C++, 37
 - C#, 37
 - CSS, 37
 - CSS3, 38
 - ECMAScript, 39
 - Flash, 40
 - HTML, 37
 - HTML5, 38
 - Java, 26
 - JavaScript, 37, 39
 - Objective-C, 37
 - PHP, 37, 39
 - PROLOG, 25
- Programmierung, 11, 12, 33, 53
 - Bibliothek, 80
 - deklarativ, 24, 90
 - funktional, 89
 - Game Engines, 80
 - general purpose programming, 75
 - IDE, 26
 - imperativ, 20, 22, 52, 57, 90
 - Java 8
 - API, 302
 - Lambda-Kalkül, 90
 - logisch, 24
 - Markup Language, 83
 - maschinennah, 50, 56, 57
 - objektorientiert, 22, 76
 - Paradigma, 77
 - parallel, 11
 - prozedural, 21
 - Script, 84
 - strukturiert, 21
 - systemnah, 11
- Projekt
 - management, 29
- Property, 208
- Protokoll, 11, 97, 119
 - FTP, 120
 - HTTP, 97
 - HTTPS, 120
 - SSL, 120
- Prozedur, 225
- Pseudocode, 23
- Quellcode, 60
- Refactoring, 61
- Referenz, 206
- Rekursion, 226
- Repository, 28

- Responsive Design, 41, 124
- Ruby, 83
- Ruby on Rails, 84
- SAP, 44
- Schemata, 187
- Schleife, 226
- Schnittstelle, 49
- SCM, 28
- Script, 84
- SCRUM, 42
- SDK, 61
- SDKs
 - MDK, 62
- search engine optimization, 185
- Selektor, 208
- SelfHTML, 130
- semantic web, 185
- Semantik, 114
 - Microdata, 38
- semantisches Web, 114
- SEO, 185
- Server, 81, 105
- shell, 67
- Sicherheit
 - IT-Sicherheit, 60
- Skalierbarkeit, 42, 83
- Software, 23
- Software Engineering, 11, 28
 - agil, 29
 - Agile Softwareentwicklung, 42
 - Design Pattern, 42
 - Extreme Programming, 42
 - MVC, 42
 - SCRUM, 42
 - Skalierbarkeit, 83
 - TDD, 42
 - V-Modell, 29
 - Wasserfallmodell, 29
 - YAGNI, 42
- SSL, 120
- statisch, 132
- Subroutine, 57
- Subversion, 28, 71
- SVN, 28, 71
- Syntax, 114
- Syntaxerkennung, 68
- System
 - FPGA, 37
 - Mikroprozessor, 52
- Systeme
 - FPGA, 10
 - SPS, 10, 40
- tabular matter, 202
- TDD, 42
- Terminal, 67
- Test Driven Development, 42
- tool, 60
- Toolchain, 61
- Typecasting, 88
- Typisierung
 - dynamisch, 86, 87, 159
 - schwach, 86
 - stark, 86
 - statisch, 86, 159
 - streng, 132
 - Typecasting, 88
- Umgebung, 198
- UML, 96
- Unicode, 141
- URI, 106
- URL, 106, 155
 - absolut, 155
 - relativ, 155, 156
- Usability, 41
- Use Case, 125
- Validator
 - W3C, 132
- Validierung, 132
- Variable, 52, 225
- Versionskontrolle, 28
- Verteilte Anwendungen, 93
- Virtuelle Maschine, 302
- Visual Studio, 63
- VLSI, 51

W3Schools, 130
Werkzeug, 60
Wichtige Institutionen
 MIT, 76
Wichtige Personen
 Kay, Allen, 76
 Kernighan, 75
 Ritchie, 75
 Ritchie, Dennis, 75
Wichtige Unternehmen
 Apple, 79
 Microsoft, 79
 Steam, 79
 Sun, 78
Windows, 97
World Wide Web, 95
WWW, 124

XCode, 63
XHTML, 39, 131

YAGNI, 42

Zugriffsmodifikator, 311