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

-

EMIFROP 0.28

Markus Alpers
B.Sc. und Ausbilder f. Industriekaufleute

7. April 2016

Inhaltsverzeichnis

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

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 Belege/Beweis kann zum einen eine andere wissenschaftliche Arbeit sind, 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 protoypbasiert 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 Bereich als **Technische Informatik** bezeichnet. Das schließt beispielsweise die Programmierung von Steuer- und Regelsystemen, also insbesondere **SPS**e und **FPGA**s 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 **Protokolle**n 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 gehören[1]. Umgekehrt enthält jeder brauchbare Kurs zum Software Engineering Teile zur Programmierung[2].

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

[2] Weshalb 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 Aufga-

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 eine 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. prozdurale 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 **Paradigma**s 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 effiziente Lösung für ein Problem, über das Sie bei Ihrer aktuellen Programmieraufgabe stolpern. Wenn Sie die Aufgabe in einer bestimmte 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 all der 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 **Klausel**n. 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 **Middleware**, **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 **IDE**s (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 (**Delta**s) gespeichert, sodass 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 Softwareßusammenfassen, 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 **agil**e 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 entwickln 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 sytemabhängig; 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 hereinfallen 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 Jobbs & 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 angeschlosse ist, ist die Technik und Technologie, durch die insbesondere kabllose 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 Sytem 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 Stuttgard 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 Bereich 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 mehrerer 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 Doktortitel 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 **FPGA**s 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 **Desktoprechner**n und **Smartphone**s 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 Plaß 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, das 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 Kombinaion 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 **SPS**en gibt es eine Programmiersprache namens Ablaufsprache, die ebenfalls mit **AS** abgekürzt wird. Im Englischen wird diese Sprache als SFC / Sequencial 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 (**Patch**en) 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 „**Algorithmendesign**" (**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 veraltert 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 (KIT[1]) 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 können[2]. 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-pro-gramm/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 **ASCI**-Code gehört. Der ASCI-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: **Interface**s bzw. **Schnittstelle**n. Ein Interface ist etwas, das die Verbindung zwischen zwei „Dingen"herstellt. Wenn wir über den Datenaustauch 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 Programmiereinsteiger sehr schnell zu tun, auch wenn wir dann von **Datentyp**en 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

[3] Hamming ist gewissermaßen der Godfather der Nachrichtentechnik.

geschrieben: Das ist nicht der Fehler der Rechners, 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 zankenden Geschwister

Wie beschrieben beschäftigen sich **Nachrichtentechnik**erInnen 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. **Informatik**erInnen 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 unvoreingenommen[4] 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 **Informatik**erInnen 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 zustanden, 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-

[4]Mit 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 schieflaufen 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 **FPGA**s, den Field-Programmable Gate Arrays direkt programmieren[5]. 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 festlegt 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 bewiesenïn 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 bewei-

[5]Diese 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 „warïst 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 wortwörlich 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 addieren[6]. 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 **Subroutine**n 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 veröffentlich 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 auszugweise 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 ein 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 **Delta**s 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 µVision 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 IAx86-Architektur[1] 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

[1] Das 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 wählten Sie 2010 im XCode
`Menü -> Preferences -> Downloads` den Eintrag „Command Line Toolsünd 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 setzten 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 **Package**s 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 für Dateien im .exe-Format das gleiche, was für alle Programm 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 „Kachelansichtßchlicht 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 webbasierte 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 Datenbankenäusführlich 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 fortgeschrittenene 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 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 Dilletantentum 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 Rechnerän 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 **Betriebssystem**e zu entwickeln und dabei möglichst wenig maschinennah 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, das 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 **Informatik**erInnen und **Nachrichtentechnik**erInnen 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 **Klasse**n 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 **Netzwerk**e und **Nachrichtentechnik** belegen. 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 eine Netzwerk nur nutzten, um Nachrichten darüber auszutauschen. **Webanwendung**en 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 **Script**en. 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 **Informatik**erInnen 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. **Datenbank**en 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 **Informatiker**Innen 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-

grammieren 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 Umwandelung 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:

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

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 $3 * 10^{-1}$ 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 **Informatik**erInnen 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 **Algorithmik** 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"Typecasting 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-Kalkül**s, 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 Programmablaufs 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 Programmablaufs 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 ProgrammiererInnen 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 dilletantischen Quereinsteiger und einem ernstzunehmenden Softwareentwickler.

Danach haben Sie etwas über Konzepte erfahren, die C-, C++- und JavaProgrammiererInnen 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 Psys 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 m^2, 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 Literaturnobellpreisträger, der in seinen Romane die zweite dieser Vorgaben umgesetzt hat: Ernest Hemmingway)
   - 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 Hochschulniveau 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 Netzwerk[1] 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 Webservers 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

[1] oder 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 dementsprechnd 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 einschalten[2]. 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.

[2]Wie 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 aufwänden.

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 ein: `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 hassen 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 dir 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 Küstler. 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, dass 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 **Hyperlink**s) 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 Protokol 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 Bildergallerien 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 wir 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 könenn 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 mittags).)

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 **Containter**.

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 das 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_begruessung_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 ungerechtigtes 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_begruessung_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>`-Conatainer.)
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 internationaliziation 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 **Wurzelelement** 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 gibt es 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` 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_begruessung_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, das 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 `\&uuml;` 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	`&Auml;`	
ä	`&auml;`	
Ö	`&Ouml;`	
ö	`&ouml;`	
Ü	`&Uuml;`	
ü	`&uuml;`	
ß	`&szlig;`	(ß mit Ligatur)
€	`&euro;`	
`&`	`&`	(ampers and)
>	`>`	(greater than)
<	`<`	(less than)
Anführungszeichen unten	`&bdquo;`	
Anführungszeichen oben	`&rdquo;`	

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 `&szlig;` programmieren.

Weiterhin gab es noch die Möglichkeiten, Escape-Sequenzen mit der Nummer der Unicode-Tabelle zu programmieren. Beispielsweise steht `&#x9fb9;`

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:

`&`	&	(kaufmännisches Und)
`<`	<	(less than)
`>`	>	(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 &lt;p&gt; öffnet
einen Absatz-Container in HTML.
```

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

```
Die Zeichenfolge &lt;p&gt; &ouml;ffnet
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 nichtstandardisierte Attribute zuordneten. Die Folge bestand darin, das 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 NutzerInnern durch die Seite scrollen. Der Main-Bereich würde dann rund zwei Drittel der Breite und der Aside-Bereich ein Drittel des Bildschirms einehmen.

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, dass 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 das `<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 einprogr
</form>
```

Der `<form>`-Container ist das Wurzelelement für Formulare, so wie der ¡html¿-Container das Wurzelelement 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 verwenden 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 Verwechselungen 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 sagts 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=importancy 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.

- Checkboxen:
  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 Checkboxen, 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 Checkboxen 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>`-Containerns 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 dass, 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 Eingabeelementen

Um mehrere Elemente zu gruppieren müssen diese lediglich in einem `<fieldset>`-Container zusammengefasst und durch einen `<br />`-Container getrennt werden. `<br>`-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 `<img>`-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 `<img>`-Container. `<img>`-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 `<img>`-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 `<img>`-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 anforderen. 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 `prelaod=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, Mikrophonen 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 nutze
n 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-

grammieren, 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 ein 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` (Hiweis: 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 nichtsemantischen Container lauten schlicht `<b>` oder `<strong>` 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 Conatainers 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 `<span>`-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 latainischen 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 `<span>`-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 `<span>`-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 `<ul>` 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 `<ol>` 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 `<li>`-Container innerhalb eines `<ul>`- oder `<ol>`-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>
<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 `<ul>` und `<ol>` 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 Zeilenumrüche (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 Programmsprache 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 all das Programmieren, was anders als beim Standard-Dokument ist.

Dieser Kurs ist eine knappe Einführung. Umfangreichere Kurs 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 geschweifter 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 mehrer Kapitel umfassen können.
- `article` ist für Artikel gedacht, die z.B. in Zeitschiften 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 sogenante 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 `\(` und `\)` 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 `^`. 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 **Umgebung**en 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 eigefügt 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 Authoren.
- `\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 aktualisert.

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 Unterstichworter 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 formular 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 geschweifter 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{graphix}`

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/coumnwidth` 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

Kapitel 9

Funktionalität mit PHP 5.6

Kapitel 10

Langfristige Datenspeicherung mit MySQL 5.6

Stichwortverzeichnis