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

-

EMIFROP 0.26

Markus Alpers
B.Sc. und Ausbilder f. Industriekaufleute

7. April 2016

Inhaltsverzeichnis

I Einfache Einführung in die imperative Programmierung 6

1 Das ist Programmieren (wirklich) 7

2 Nachrichtentechnik und Programmierung 8

3 Vorbereitung fürs Programmieren 9

4 Ausgewählte Programmiersprachen 10

5 Grundlagen verteilter Anwendungen 11

6 Einstieg in HTML 5 12

6.0.1 Das ist HTML . . . 13
6.0.2 Erster HTML-Quellcode . . . 14
6.0.3 Struktur eines HTML-Dokuments . . . 19
6.1 Barrierefreiheit, Internationalisierung und Lokalisierung . . 26
6.1.1 Barrierefreiheit . . . 26
6.1.2 Internationalisierung (kurz i18n) und Lokalisierung (kurz l10n) . . . 27
6.2 Der head-Container, Meta-Daten und Attribute . . . 28
6.3 Mehr Grundlagen in HTML . . . 29
6.3.1 Mehr Auslagerung von Code . . . 29
6.3.2 Verwendung von Escape-Sequenzen . . . 30
6.3.3 HTML5: Anführungszeichen sind weitestgehend optional . . . 31
6.3.4 Zusammenfassung . . . 32
6.4 Strukturen von HTML5-Dokumenten . . . 32
6.4.1 header, footer, main und aside . . . 33
6.4.2 article und section . . . 34
6.4.3 h1 bis h5 . . . 34
6.4.4 p . . . 34
6.4.5 Aufgabe . . . 35
6.5 Polyfills . . . 35
6.5.1 Einbindung von Polyfills - ¡script¿-Container . . . 36

6.6 Zusammenfassung 36
6.6.1 Hausaufgabe 37
6.7 Hyperlinks 38
6.7.1 Anker 38
6.7.2 Links 39
6.7.3 Verlinkungen als expliziter Download 40
6.7.4 Hausaufgabe: 40
6.7.5 URLs – absolute und relative Adressen 41
6.8 Formulare 42
6.8.1 Elemente eines Formulars 42
6.8.2 Das name-Attribut und das id-Attribut – Sonderfälle in Formularen 43
6.8.3 Container für Formulare 44
6.8.4 Container für die Gruppierung und Zuordnung von Eingabeelementen 52
6.8.5 Zusammenfassung 53
6.9 Multimediale Inhalte einfügen 53
6.9.1 Bilder 54
6.9.2 ¡figure¿ und ¡figcaption¿ 55
6.9.3 ¡figcaption¿ für Fortgeschrittene 55
6.10 Weitere multimediale Formate 56
6.10.1 Einbindung eigener und frei verfügbarer Videodateien 56
6.10.2 Anpassungsmöglichkeiten für den Video-Player . . . 57
6.10.3 Einbindung von geschützten Inhalten (Stichwort: DRM) 58
6.10.4 Der ¡audio¿-Container 59
6.10.5 Close Captions, Untertitel, Einbindung von Webcams usw. 59
6.10.6 Hinweis bezüglich Flash und ähnlichen Formaten . . 59
6.10.7 Hausaufgabe 60
6.11 Weitere Formatierungen und Möglichkeiten in HTML 61
6.11.1 Spoiler und andere ausklappbare Texte 61
6.11.2 Zeitangaben 62
6.11.3 Hervorhebung von Texten 63
6.11.4 Unterdrückung von Übersetzungen für Textpassagen 63
6.11.5 Aufzählungen (Ordered und Unordered Lists) 65
6.11.6 Glossare (Description Lists) 65
6.11.7 Tabellen (table) 66
6.11.8 Microdata 66
6.11.9 Validator für Microdata 70
6.12 Zusammenfassung 70
6.12.1 Mehr Text braucht die Welt . . . und Zeilenumbrüche 71
6.12.2 Hyperlinks – Verbindungen zwischen Elementen . . 73
6.12.3 Entitys - Umlaute und andere Sonderzeichen 73
6.12.4 Deutsche Umlaute ohne Entities 74

6.12.5 Hervorhebungen 75
6.12.6 Formulare 76
6.12.7 Universalattribute 81
6.12.8 Labels – Beschriftungen für input-Container 81
6.12.9 CSS – Cascading Style Sheets 82
6.12.10 CSS und unterschiedliche Displayformate 84
6.12.11 Schriftsätze und –farben – CSS/font 84

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.

Teil I

Einfache Einführung in die imperative Programmierung

Kapitel 1

Typische Irrtümer darüber, was Programmieren ist.

Kapitel 2

Von der Nachrichtentechnik zur Programmierung

Kapitel 3

Vorbereitung fürs Programmieren

Kapitel 4

Ausgewählte Programmiersprachen

Kapitel 5

Grundlagen der Entwicklung verteilter Anwendungen

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 deut-

lich, 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.

6.3 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 Escape-Sequenzen 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:

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

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

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

dann müssen Sie das so einprogrammieren:

```
Die Zeichenfolge &lt;p&gt; öffnet einen Absatz-Container in HTML.
```

Unter HTML4.01 hätten Sie zusätzlich für das ö von öffnet die Escape-Sequenz `&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 wer-

den 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 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. Der 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 In
</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 effiz
</p>
<p>Programmieren sind Menschen, die Konzepte und Modelle in eine Programm
</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, Pro
<section>
<h2>Programmierung von Webanwendung und Webapplicationen</h2>
<p>Die bekannteste Markup Language ist HTML, die HyperText Markup Languag
<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 kenne
<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 unters
  ```

- 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>Kohlroula
  ```

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 G
```

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. Ei-

ne 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 Elemente 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 dadurch 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 aber redundant: `<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 erreiche
</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-/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 Leistungsnachweis „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"`

- 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 furchtbarst
<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 Videodat

</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 nutzen müssen.

Sprich: Sie dürfen zwar innerhalb eines `<summary>` mehrere `<details>`-Container programmieren, aber keinen weiteren `<summary>`. Den müssten

Sie dann wieder innerhalb eines der `<details>` programmieren. Umgekehrt gilt das selbe.

Natürlich müssen Sie auch bei diesen Containern prüfen, ob Sie inzwischen in allen Webbrowsern unterstützt werden und ggf. ein passendes Polyfill einbinden.

6.11.2 Zeitangaben

Wichtig:

Bitte beachten Sie, dass es sich bei dem gleich vorgestellten Container um einen Container handelt, den Sie im Gegensatz zu Formularcontainern wie `<input type=datetime>` und ähnlichen `input`-Containern an beliebigen Stellen innerhalb eines `<body>`-Containers nutzen können. (Zur Erinnerung: `input`-Container bieten NutzerInnen die Möglichkeit, Eingaben durchzuführen.)

Bitte beachten Sie ebenfalls, dass der `<time>`-Container keine Ausgabe im Browser erzeugt, sondern einzig dafür sorgt, dass ein Teil des Dokuments einen Zeitstempel erhält. Das bedeutet, dass ein Browser hier direkt erkennen kann, dass ein bestimmter Bereich etwas mit einem bestimmten Zeitpunkt zu tun hat. Wenn also die EntwicklerInnen des Browsers den `<time>`-Container richtig auswerten lassen, dann können Nutzer sich einen Termin direkt aus einer Webanwendung in den eigenen Terminkalender eintragen lassen, ohne dass Sie als EntwicklerIn der Webanwendung dafür etwas programmieren müssten.

Mit dem `<time>`-Container können Sie also Zeitangaben und Zeiträume im Sinne des semantic Web definieren. Leider ist es nicht möglich, dass damit alle möglichen Zeitangaben darstellbar wären, denn im Kern wird hierdurch ein Element definiert, dass einen Zeitpunkt oder einen in Sekunden messbaren Zeitraum festlegt. Genau wie bei den entsprechenden `input`-Containern haben Sie also keine Möglichkeit, einen Zeitraum mithilfe eines einzelnen Containers zu programmieren.

Zur Erklärung:

Da eine Zeitangabe wie die vom 20. Februar bis zum 3. März nicht eindeutig ist (denken Sie an Schaltjahre, bei denen es einen 29. Februar gibt), gibt es auch keinen einzelnen `<time>`-Container, mit dem Sie diesen Zeitraum zusammen fassen können. Aus dem gleichen Grund können Sie Zeiträume nicht in Monaten oder Jahren festlegen; in solchen Fällen müssen Sie zwei `<time>`-Container programmieren, den einen für den Anfang, den ande-

ren 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 einprogrammieren, 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 nicht-semantischen 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¿ Quellcode 2.18: Verhinderung von automatischen Übersetzungen durch Browser.

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¿ Quellcode 2.19: Vererbung von Attributen
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¿ Quellcode 2.20: Überschreiben eines vererbten Attributs

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¿ Quellcode 2.21: Geordnete und Ungeordnete Liste

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 einzupro-

grammieren:

¡dl¿ ¡dt¿Kaffee, schwarz¡/dt¿ ¡dd¿Heißes Getränk, koffeinhaltig, häufig im Becher von Herrn Alpers anzutreffen.¡/dd¿ ¡dt¿Bohnesuppe¡/dt¿ ¡dd¿Kohlenhydrathaltiges Gericht, häufig in Italo-Western von Bud Spencer konsumiert.¡/dd¿ ¡/dl¿
Quellcode 2.22: Glossare / descriptive list

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.

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¿ 10.00¡/td¿¡td¿PRG¡/td¿...¡/tr¿...¡/table¿ Quellcode 2.23: Ausgesprochen unübersichtliches Beispiel für eine Tabelle

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 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 durch eine Programmiersprache folgt in der Einführung in JavaScript (also nur für MediaSystems Studierende) später.
- 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¿ Quellcode 2.24: Definition eines Absatzes als Microdata für die Anschrift einer Person beinhalten soll.

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.

Anmerkung: Diejenigen von Ihnen, die diese Veranstaltung durch eine Klausur abschließen, sollten das selbst gemacht haben, weil eine Aufgabe in Ihrer Klausur darin besteht, eine passende itemprop für einen itemtype aus einer Liste auszuwählen.

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¿ Quellcode 2.25: Microdata, die den Namen einer Person enthält.

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= ... ¿Twitter¡/a¿¡/li¿ ¡li¿¡a href=http://www.blogger.com/ihrBlogAccount itemprop= ... ¿Webblog¡/a¿¡/li¿ ¡/ul¿ ¡/section¿ Quellcode 2.26: Umfangreiche Microdata als Aufgabe zur Recherche

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.

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.

6.12.1 Mehr Text braucht die Welt ... und Zeilenumbrüche

Ab hier finden Sie die Abschnitte des ersten HTML-Skripts, das ich erstellt habe. Dieser Teil ist weitgehend veraltet, da er das Vorgehen bei der Programmierung in HTML 4.01 beinhaltet. Er ist zurzeit noch enthalten, da ich prüfe, welche Abschnitte ggf. noch aktuell sind.

Nun gut, die Überschrift mag nicht ganz der Wahrheit entsprechen, aber tun wir einmal so, als wenn sie zuträfe. Erweitern Sie also einfach den body-Container um einige Sätze. Es ist egal, ob diese Sinn machen oder reines Buchstabenchaos sind, da es hier darum geht, dass Sie sehen, welche Anzeige aus Ihrem Quellcode erzeugt wird. Speichern Sie das Ergebnis dann als seite02.html im bekannten Verzeichnis ab, wechseln Sie im Browser auf localhost bzw. aktualisieren Sie die Seite und öffnen sie dann die neue Webanwendung.

Einschub: .htm und .html

Wenn Sie sich etwas intensiver mit der Entwicklung von Webanwendung beschäftigen, dann werden Sie sehen, dass manche HTML-Skripte unter Dateibezeichnungen gespeichert werden, die nicht auf .html enden, sondern auf .htm. Streckenweise werden Sie auch Kursunterlagen finden, in denen gesagt wird, dass Sie die Skripte in Dateien speichern sollen, deren Namen mit .htm enden. Neugierige Naturen fragen nun: Wo ist der Unterschied? Kurzantwort: Es gibt keinen. Lange Antwort: Bis in die 90er Jahre war es wichtig, so effizient wie möglich zu programmieren. Deshalb gab es z.B. bei damaligen Betriebssystemen die Einschränkung, dass Dateitypen durch eine Endung festgelegt wurden, die nur drei Buchstaben haben durfte. Als immer mehr Hauptspeicher in Rechnern zur Verfügung stand war diese Beschränkung nicht mehr nötig. Wo also früher .htm stand, wurde irgendwann .html eingegeben. Das ist allerdings nicht bei allen Dateitypen so simpel, Sie dürfen es also nur bei HTML-Skripten voraussetzen.

Wichtig: Auch wenn Sie beide Endungen nutzen dürfen gilt: seite.htm ist eine andere Datei als seite.html

Wieder zurück zu unserer ersten Webanwendung:

Und was sehen Sie? Etwas sehr unschönes: Der gesamte Text zieht sich in der ersten Zeile der Webanwendung hin und wird dann auf der nächsten Zeile von links nach rechts fortgesetzt, ohne dass Zeilenumbrüche eingefügt werden.

Was ein Zeilenumbruch ist? Immer wenn Sie bei einer Textverarbeitung die Enter- bzw. Return-Taste drücken, wird die aktuelle Zeile beendet und der Text in der nächsten Zeile fortgesetzt. Bei Microsoft Word erhalten Sie dagegen einen Zeilenumbruch durch die Tastenkombination Shift & Return.

Aber Sie können einen Zeilenumbruch in HTML nicht generieren, indem Sie wie bei einer Textverarbeitung einfach die Enter-Taste drücken, sondern Sie müssen wieder ein Tag setzen:

¡br /¿

Dieses Tag fügt einen Zeilenumbruch in die Webanwendung ein. Nun fragen Sie sich vielleicht, was denn der Slash / in diesem Tag soll. Das ist ganz einfach: Da jedes Tag einen Container öffnet oder schließt, ein Zeilenumbruch aber gleich abgeschlossen ist, würde es wenig Sinn machen, ein öffnendes und ein schließendes br-Tag zu programmieren. Als Kurzschreibweise können Sie deshalb diese Form nutzen. Und damit handelt es sich hier also nicht einfach nur um einen Tag, sondern um einen vollständigen Container.

Das selbe gilt für alle Tags, die keinen eigenen Container brauchen. Hier wäre ein Beispiel: Wenn Sie eine waagerechte Linie in Ihre Seite einfügen wollen, nutzen Sie einfach das folgende Tag:

¡hr /¿

Aufgaben:

(1) Probieren Sie das gleich mal aus: Fügen Sie wenigstens einen br-Container und einen hr-Container in Ihr Dokument ein. Speichern Sie die Datei und lassen Sie sie sich im Browser anzeigen.
(2) Und probieren Sie jetzt einmal aus, was mit einem Text passiert, den Sie in einen hr-Container einfügen. (Sprich: Sie programmieren eine öffnendes hr-Tag vor diesem Text und ein schließendes hr-Tag dahinter.)

6.12.2 Hyperlinks – Verbindungen zwischen Elementen

Mittlerweile haben Sie mehrere Seiten gespeichert. Und da ist es unschön, dass wir noch keine Links haben, mit denen wir zwischen den Seiten wechseln können. Sonst hat ja jede Webanwendung Hyperlinks, über die wir genau das tun können, also sollten wir als nächstes Hyperlinks in unsere Seite einfügen.

Das Skript für einen einfachen Link sieht so aus:

¡a href=URL¿Ein wenig Text¡/a¿

Bei einem konkreten Link müssen wir jetzt also noch die URL des Ziels einfügen. Beispiel: Wir wollen in unser Skript für die erste Seite einen Link auf die zweite Seite einfügen. Das sähe dann so aus:

¡a href=seite02.html¿Hier geht's zur zweiten Seite¡/a¿

6.12.3 Entitys - Umlaute und andere Sonderzeichen

All das erklärt aber immer noch nicht, wie Sie Umlaute oder andere Sonderzeichen (die im englischen Alphabet nicht vorkommen) in Ihre Webanwendung einpflegen. Leider müssen wir hier auf eine etwas umständliche Programmierung zurückgreifen.

So müssen Sie beispielsweise anstelle des Buchstaben ü das Skript ü in den fließenden Text eintragen. Aber das sieht nur auf den ersten Blick unübersichtlich aus: Jeder dieser Skriptbefehle 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 Ä ä ä Ö Ö ö ö Ü Ü ü ü ß ß (ß mit Ligatur) € €
& & (ampers and) ¿ > (greater than) ¡ < (less than)

Anführungszeichen unten „ Anführungszeichen oben ”
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 folgendes Skript, mit dem wir das gewünschte Zeichen einfügen können:

龹

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.

6.12.4 Deutsche Umlaute ohne Entities

Kommt es Ihnen zu kompliziert vor, wenn Sie jedes Sonderzeichen als eine Entity eingeben müssen? Da sind Sie nicht alleine. Deshalb kommen wir jetzt zu einer Zeile, die uns die ganze Arbeit abnimmt. Mit dieser Zeile teilen wir dem Webbrowser mit, dass wir unsere Zeichen mit UTF-8 codieren. Und dann versteht braucht er keine Entities mehr, sondern wir können direkt Umlaute (oder auch Schriftzeichen anderer Sprachen) direkt im Skript unterbringen.

Für Fortgeschrittene: Unter Umständen kommen Sie in die Situation, dass Sie UTF-16 oder UTF-32 nutzen wollen. Selbst bei aktuellen chinesischen Texten wird das nicht passieren, da auch diese mit dem Zeichenvorrat von UTF-8 abgedeckt sind. Hiervon rät das W3C bei Webanwendung ab. Prüfen

Sie in einer solchen Situation, welche alternativen Möglichkeiten Sie haben. Ggf. gibt es ein anderes Dokumentformat, das Ihnen in dieser Situation weiter hilft.

Hier das Skript, das Sie bitte als erste Zeile in Ihren head-Container einpflegen:

¡meta charset=utf-8¿

Um Ihre Webanwendung zu internationalisieren sollten Sie außerdem das öffnende html-Tag wie folgt erweitern:

¡html lang=de¿

Wichtig: Dennoch müssen Sie die Entities beherrschen, da Programmiersprachen wie JavaScript hiervon nicht betroffen sind. Dort müssen Sie selbst Anführungszeichen als " eintragen, was nicht nur die Fehleranfälligkeit erhöht, sondern auch die Programmierung deutlich erschwert.

6.12.5 Hervorhebungen

Bislang haben Sie zwei Möglichkeiten kennen gelernt, um Texte innerhalb eines HTML-Skripts hervorheben zu lassen: Zeilenumbrüche, mittels derer Sie Absätze erzeugen können und waagerechte Linien.

Die meisten Nutzer möchten aber auch kursiv, fett oder unterstrichen nutzen, um Textstellen hervorzuheben. Fortgeschrittene Nutzer verwenden außerdem standardisierte Einstellungen, mit denen Sie mehrere Überschriftentypen haben. Dadurch kann ein Leser am Format eines Textes erkennen, ob es sich um eine Kapitelüberschrift, eine Unterkapitelüberschrift usw. handelt.

Für die Programmierung in HTML handelt es sich jeweils wieder um Container. Sie programmieren also wieder ein öffnendes und schließendes Tag für jede Art der Hervorhebung:

Fettdruck ¡b¿ ... ¡/b¿ (bold) Kursive Schrift ¡i¿ ... ¡/i¿ (italic) Unterstrichener Text ¡u¿ ... ¡/u¿ (underline)
Kapitelüberschrift ¡h1¿ ... ¡/h1¿ (Header) Unterkapitel ¡h2¿ ... ¡/h2¿ ... ¡h3¿ ... ¡/h3¿
Aufgabe:

Warum ist das folgende HTML-Skript falsch? ¡b¿¡i¿Hallo¡/b¿¡/i¿ Welt

Kontrolle

Sie können bis jetzt Texte auf eine Webanwendung hochladen, in denen jedes Zeichen vorkommen kann, dass in irgend einer Sprache der Welt vorkommt.

Sie können außerdem einzelne Bereiche eines Textes hervorheben und eine erste Strukturierung mit Hilfe von Überschriften erzeugen.

Sie wollen jetzt endlich etwas darüber erfahren, wie Sie einzelne Elemente einer Webanwendung anordnen können? Nicht so schnell mit den jungen Pferden. Schauen wir uns doch zunächst an, wie Sie in HTML Eingaben durch den Nutzer ermöglicht werden können. Und damit sind wir im Bereich von Formularen.

6.12.6 Formulare

Der Begriff Formular umfasst alles, was bei einem HTML-Skript Eingaben durch den Nutzer ermöglicht. Grundsätzlich gilt: Für Formulare verwenden wir form-Container: ¡form¿ ... ¡/form¿

Wichtig: Ohne Unterstützung durch Sprachen wie PHP, JavaScript oder ähnliche können wir zwar Nutzereingaben in einer Webanwendung einprogrammieren, aber wir können diese Eingaben noch nicht weiter nutzen. Wir behandeln Sie aber schon jetzt, weil Sie sich zunächst überlegen sollen, welche Interaktionsmöglichkeiten ein Nutzer haben sollte, um eine bestimmte Funktionalität der Webanwendung zu realisieren. Wie die Webanwendung auf diese Eingaben reagiert werden wir dann später programmieren.

Für Fortgeschrittene: Aber auch ohne eine dieser Sprachen können wir seit HTML5 Einschränkungen Programmieren, die z.B. dafür sorgen, dass ein Nutzer bei einem Eingabefeld nur ein echtes Datum (also z.B. keinen 33. 7. 1922) eingeben kann. Wenn Sie daran interessiert sind, solche Einschränkungen gleich in HTML5 zu programmieren, nutzen Sie dazu bitte Selfhtml oder eine andere Quelle.

Nutzereingaben in HTML programmieren

Wenn Sie eine Nutzereingabe annehmen wollen, dann müssen Sie innerhalb eines form-Containers einen input-Container für jede Eingabe programmieren:

¡form¿ ¡input¿Erste Eingabe:¡/input¿ ¡input¿Zweite Eingaben:¡/input¿ ... ¡/form¿

Kontrolle:

(1) Prüfen Sie, wie dieser Quellcode auf Ihrer Seite angezeigt wird und was Sie wohl ändern müssen, damit die Abfrage sinnvoll dargestellt wird.
(2) Oh, und natürlich sollen Sie die HTML-Skripte Ihres Projekts jetzt so erweitern, dass Nutzereingaben tatsächlich möglich sind.
(3) Prüfen Sie außerdem, was mit den Eingaben passiert, wenn Sie den Quellcode ändern, die geänderte Fassung speichern und die Seite dann nochmal laden.

Überblick über Formularelemente

Mit dem input-Container können wir bislang nur Eingaben verarbeiten, die z.B. über eine Tastatur eingegeben werden. Ein Datum kann also bei dieser einfachen Form von input-Containern nicht über ein Fenster angewählt werden, obwohl das in HTML mit einem input-Container programmiert werden kann.

Wenn wir solche Eingabemöglichkeiten in HTML realisieren wollen, müssen wir den Typ eines input-Containers explizit definieren. Dazu ergänzen Sie schlicht das input-Tag wie folgt, wobei Sie anstelle der drei Punkte den Bezeichner angeben müssen, der dem jeweiligen Typ entspricht:

¡input type=“ ... “¿

Wenn Sie einige dieser Typen kennen, sollte das ausreichen. Gehen Sie auch hier schlicht danach vor, welche Typen für die Funktionalitäten Ihrer Webanwendung von Belang sind. Leider wird nicht jeder Typ von jedem Browser unterstützt. Entwickeln Sie Ihre Seite zunächst so, als wenn diese Einschränkung nicht gültig wäre. Die Anpassung einer Webanwendung für unterschiedliche Browser ist ein fortgeschrittenes Thema, das den professionellen Webdeveloper auszeichnet.

Für jeden Typen gibt es noch eine Reihe sogenannter Attribute, mit denen Sie weitere Einstellungen für die Nutzereingabe vornehmen können. Dazu müssen Sie Ihr input-Tag wie folgt erweitern:

¡input type=“ ... “ attribut1=“ ... “ attribut2=“ ... “¿

Wie sie sehen gibt es hier im Gegensatz zu vielen imperativen Sprachen kein Zeichen, das Sie zwischen die verschiedenen Einträge setzen müssen.

In C und den nachfolgenden Sprachen mussten Sie beispielsweise ein Komma setzen. So etwas gibt es in HTML nicht.

Genau wie den Typ legen Sie also ein Attribut für eine Eingabe fest, indem Sie das Attribut in das öffnende input-Tag eintragen, dazu ein Gleichzeichen und anschließend den Wert den Sie dem Attribut geben bzw. zuordnen wollen. Diesen Wert müssen Sie immer in Anführungszeichen setzen.

Zunächst einige Typen, mit denen Nutzer verschiedene Arten von Texten eingeben können:

• "text": Der Name ist selbstredend: Hier kann der Nutzer einen Text eingeben. • "search": Dieser Typ ermöglicht es dem Nutzer nach einem Begriff zu suchen. Als Entwickler können Sie optional Begriffe festlegen, die gefunden werden können. • "number": Sie können bei diesem Typen einen Zahlenbereich vorgeben. • "email" und "url" sind wenig sinnvolle Typen, da Sie mit ihnen zwar prüfen können, ob eine Eingabe eine gültige Email-Adresse bzw. eine gültige URL sein könnte, aber die Prüfung, ob sie tatsächlich existiert bzw. ob sich hinter der Angabe ein Skriptbefehl versteckt, über den ein Angriff auf Ihre Seite erfolgen soll, das kann dieser Typ nicht feststellen.

Jetzt einige Typen, die es dem Nutzer ermöglichen, Angaben aus einer Tabelle bzw. einem neuen Fenster auszuwählen:

• "tel": Damit kann der Nutzer eine Telefonnummer eingeben. • "date": Hier kann der Nutzer ein Datum anwählen. • "time": Hier kann er einen Zeitpunkt anwählen. • "datetime": Zusätzlich zum Datum (wie bei date) kann hier noch eine Uhrzeit angewählt werden. • "datetime-local": Im Gegensatz zu datetime wird hier die Zeitzone ignoriert. • "month": Hier kann der Nutzer das Jahr und den Monat als vier- bzw. zweistellige Zahl eingeben. • "week"': Dreimal dürfen Sie raten... • "color": Hier kann der Nutzer eine Farbe aus einer Farbtafel auswählen. Abschließend noch eine Möglichkeit, um Schieberegler einzufügen: • "range": Damit können Sie Schiebregler und Schalter programmieren, sodass der Nutzer keinen festen Wert eingeben muss bzw. kann. Ein solches Element kann sowohl horizontal wie vertikal programmiert werden.

Aufgabe:

Prüfen Sie, welche Typen für die Nutzereingaben in all Ihren HTML-Skripten sinnvoll sind. 30 Nutzereingaben sollten es wenigsten sein. Und nein, Sie sollen nicht auf jeder Webanwendung die gleichen Nutzereingaben programmieren; stellen Sie sich vor, Sie wären auf einer Seite unterwegs, die

Sie pausenlos dazu auffordert, die gleichen Eingaben zu machen.

Der Typ "text"

Diesen Typen kennen Sie bereits: Jeder input-Container, der keinen expliziten Typen hat wird in HTML als input vom Typ Text interpretiert.

Sie können hier die folgenden Attribute verwenden. Wie immer gilt: Prüfen Sie, welche dieser Attribute bei Ihren Eingaben Sinn machen und programmieren Sie dann entsprechend. Sie brauchen wirklich nicht jedes Attribut im Detail zu wissen; wichtig ist, dass Sie im Stande sind, Attribute zu programmieren. Professionelle Entwickler zeichnen sich nicht dadurch aus, dass Sie alles wissen, sondern dadurch, dass sie solche Details schnell nachschlagen.

Das führt auch direkt zum nächsten Hinweis: Es gehört zu den essentiellen Kenntnissen, die Sie bei der Programmierung in vielen Sprachen benötigen, dass Sie lernen, Referenzen zu nutzen. Ihre nächste Aufgabe wird darin bestehen, genau das zu üben. Und keine Sorge: Das fällt jedem anfangs schwer, weil Referenzen nicht dafür verfasst sind, lesbar zu sein. Dafür sind nämlich Lehrbücher da. Referenzen sind Nachschlagewerke für Programmierer.

• Das Universalattribut id. Dazu folgt etwas später eine Erklärung. Für den Moment ignorieren Sie es bitte. • "name" kann von JavaScript genutzt werden, um die Eingabe zu nutzen. • "size" gibt an, wie viele Zeichen sichtbar sein sollen. Wenn Sie hier einen zu großen Wert angeben, dann führt das zu Problemen mit kleinen Displays wie bei Smartphones. Ist der Wert zu klein, dann stört das den Nutzer, weil er zu wenig von seiner Eingabe sieht. • "maxlength" gibt an, wie lang die Eingabe des Nutzers maximal sein darf. Im Gegensatz zu size, das nur die Anzeige beeinflusst, schränkt maxlength also tatsächlich die Eingabemöglichkeiten des Nutzers ein. • "value" legt einen Standardinhalt fest. So könnten Sie in ein Eingabefeld für den Vor- und Nachnamen so etwas wie Mäxchen Müller einblenden lassen, das durch die Nutzereingabe überschrieben wird. • "placeholder" hat eine ähnliche Funktion wie value. Probieren Sie es doch einfach mal aus, um den Unterschied zu sehen. • "readonly" kann dazu genutzt werden, damit ein text-Feld nicht geändert werden kann. So kann zum Beispiel verhindert werden, dass ein Nutzer eine Angabe einträgt, bevor er den Nutzungsbedingungen Ihrer Seite zugestimmt hat. Schlagen Sie einmal nach, welche Werte dieses Attribut haben kann. • "required" dagegen verhindert, dass die Eingaben eines Formulars abgeschickt werden können, wenn eine entsprechende Angabe vom Nutzer noch nicht durchgeführt wurde. • "pattern" ist ein sehr mächtiges Attribut, weil Sie hierüber genau pro-

grammieren können, was für Zeichen bei einer Texteingabe erlaubt sind und in welcher Reihenfolge diese auftreten dürfen. Um das richtig nutzen zu können, sollten Sie ein bereits die Veranstaltung Theoretische Informatik besucht haben.

Aufgabe:

Programmieren Sie jetzt für jede Nutzereingabe nötige und sinnvolle Attribute mit Werten, die Sinn machen.

Das id-Attribut – Variablen in HTML

Sie haben gelernt, dass Sie in HTML keine Nutzerangaben verarbeiten können. Aber wie Sie jetzt wissen können Sie Eingabefelder programmieren, mit denen ein Nutzer Eingaben erstellen kann. Wie Sie vielleicht wissen werden solche Eingaben in Programmiersprachen mit Hilfe sogenannter Variablen gespeichert und verarbeitet. Damit wir also später die Nutzereingaben in PHP verarbeiten können, müssen wir nun jeden input-Container einer Variablen zuordnen.

Und das erledigen wir über das id-Attribut. Wir programmieren es genauso, wie wir andere Attribute eines input-Containers programmieren. Der Unterschied besteht allerdings in zwei Dingen:

(1.) Das id-Attribut ändert nichts an der Darstellung oder Verarbeitung der Nutzereingabe; es ermöglicht lediglich, dass wir diese Eingabe mit einer Sprache wie PHP oder JavaScript verwenden können.
(2.) Der Wert des Attributs darf (erst ab HTML5) alle Zeichen außer einem Leerzeichen enthalten.

Wichtig: Ein id-Attribut darf auf jeder Webanwendung nur einmal verwendet werden, da sonst der Browser nicht weiß, welches Element Sie mit diesem id-Attribut referenzieren. Sie sollten deshalb über Kommentare jedes vergebende id-Attribut und seinen Zweck am Anfang Ihrer HTML-Skripte festhalten.

Praktisch: Das id-Attribut kann außerdem dazu genutzt werden, um später auf eine Stelle in einem HTML-Skript zu verlinken. Wenn Sie später wissen, wie Sie ein Dokument mit anderen Methoden als dem br-Container strukturieren können, dann können Sie es dort einsetzen.

Kontrolle:

Haben Sie alle input-Container mit einem id-Attribut bezeichnet? Nein, na dann wissen Sie ja, was Sie jetzt tun sollten.

6.12.7 Universalattribute

Sie haben gerade das id-Attribut kennen gelernt. Dieses Attribut unterscheidet sich von allen anderen Attributen, die Sie bislang kennen gelernt haben in einem entscheidenden Punkt: An jeder Stelle eines HTML-Skriptes, wo Sie es nutzen können hat es die selbe Bedeutung.

Alle Attribute, die in HTML diese Bedingung erfüllen, also alle Attribute, deren Bedeutung immer gleich bleibt, werden als Universalattribute bezeichnet. Leider werden bei selfhtml in den Beispielen zu Formularen id-Attribute aufgeführt, aber es wird dort nicht explizit angegeben, welche Universalattribute noch verwendet werden dürfen.

6.12.8 Labels – Beschriftungen für input-Container

Bislang haben Sie für jede Abfrage manuell eine Beschriftung hinzugefügt, die aber im Grunde gänzlich unabhängig von diesem input-Container eingefügt wird. Zur Erinnerung: Vorhin haben wir eine Eingabe wie folgt abgefragt: Erste Eingabe:¡input \¿

Nun wäre es aber sinnvoll, wenn bereits im Quellcode und damit für den Browser erkennbar der Text Erste Eingabe zum input-Container zugeordnet wäre. Ursprünglich hatten wir erfolglos versucht, das zu erreichen, indem wir die Zeile wie folgt programmiert haben: ¡input¿Erste Eingabe:¡/input¿

HTML bietet uns für diese Fälle den label-Container an. Das sähe dann so aus: ¡label¿Erste Eingabe:¡input /¿¡/label¿

Nun möchten wir ja solche Eingaben immer mit einem id-Attribut versehen, damit wir die Eingabe weiter verarbeiten können. Aber dann haben wir ein Problem, denn wir dürfen die Zuordnung des id-Attributs ja nur einmal vornehmen, weil der Browser sonst nicht weiß, auf welches Element wir uns beziehen.

Wenn wir jetzt aber das id-Attribut im input-Container belassen, dann ist das label immer noch unabhängig vom input und wir bekommen bei einem Hyperlink auf die id ggf. nicht die Ansicht, die wir uns wünschen.

Deshalb gibt es das for-Attribut für label-Container. Wenn ein label sich auf einen input beziehen soll, dann verwenden wir das for-Attribut und ord-

nen diesem den selben Wert bzw. den selben Eintrag zu, den wir dem entsprechenden it-Attribut des input-Containers zugeordnet haben. Das sieht dann so aus:

¡label for=“ersteEingabe“¿Erste Eingabe:¡input id=“ersteEingabe“ /¿¡/label¿

Aufgabe:

Na kommen Sie schon. Sie wissen, welche Aufgabe jetzt ansteht.^ ^

Kontrolle:

Sie können jetzt Texte auf Webanwendung programmieren und unterschiedliche Arten von Eingaben durch Nutzer vornehmen lassen.

Bevor wir uns der Frage zuwenden, wie solche Eingaben verarbeitet werden können, kommen wir zunächst zur fortgeschrittenen Programmierung von statischen Webanwendung: Der Programmierung mit CSS.

6.12.9 CSS – Cascading Style Sheets

In der Anfangszeit von HTML wurden Elemente einer Seite ausschließlich durch Container kombiniert, wie sie Sie bis jetzt kennen gelernt haben. Diese Container bewirkten direkt, dass ein Element an einer bestimmten Stelle oder in einer bestimmten Formatierung auf der Webanwendung platziert wird.

Dabei waren viele Einstellungen durch den Browser oder durch HTML vordefiniert, sodass es Designentscheidungen gab, die nicht in HTML umgesetzt werden konnten. Außerdem war eine Planung des Seitenlayouts nur für jede Webanwendung einzeln möglich. CSS stellt deshalb Möglichkeiten zum Entwurf von Strukturen und Formaten bereit, die dann durch einen kurzen Befehl auf beliebig vielen Einzelseiten umgesetzt werden kann, sodass ein einheitliches Layout aller Webanwendung mit geringem Aufwand möglich wird.

Wichtig: Wenn etwas mit CSS möglich ist, erledigen Sie es mit CSS!

Die Möglichkeiten gehen dabei so weit, dass Sie für jedes Element auf den Bildpunkt genau definieren können, wo ein Element platziert werden soll. Außerdem bietet CSS Ihnen die Möglichkeit beispielsweise zwischen einer Ausgabe auf dem Bildschirm und auf einem Drucker zu unterscheiden.

Selbst die Ausgabe über Lautsprecher ist hierdurch möglich.

Das bedeutet auch, dass Sie teilweise durch CSS Vorgaben aus HTML-Skripten überschreiben können. Die entsprechenden Passagen des HTML-Skriptes bleiben dann aber im Quellcode unverändert erhalten. Gerade als Einsteiger können Sie hier Schwierigkeiten bekommen, denn das bedeutet, dass es mitunter keine Auswirkungen hat, wenn Sie im HTML-Skript etwas ändern. Hier hilft einzig und alleine die Erfahrung weiter, denn weder der Webbrowser noch das HTML-Skript wird Ihnen mitteilen, dass eine Passage Ihres HTML-Skriptes durch ein CSS-Skript überschrieben wurde.

CSS-Bereiche eines HTML-Skripts sind leicht erkennbar, da hier häufig geschweifte Klammern zum Einsatz kommen. Das bedeutet aber auch, dass Sie nicht mehr ausschließlich über Tags und Container Format und Position von Elementen auf Ihrer Seite programmieren, sondern eben zusätzlich über die Skripte, die für CSS verwendet werden.

Einen doctype wie in HTML gibt es nicht.

Auch bei CSS geht die Entwicklung kontinuierlich weiter. Und so wie die aktuelle Version von HTML die Nummer 5 trägt, ist bei CSS die Version 2.1 aktueller Standard. Aber im Gegensatz zu HTML dauert es bei CSS deutlich länger, bis neue Standards veröffentlicht und umgesetzt werden.

Leider können Sie bei einem CSS-Skript keine Angabe zur Version machen. Im Gegensatz zu HTML sind Sie hier also darauf angewiesen, dass der Webbrowser schon weiß, welche Version Sie verwenden. Erkennt er das nicht, dann passiert es im schlimmsten Fall, dass er Ihre Formatierungen kommentarlos ignoriert.

Für Fortgeschrittene – Präprozessoren: ACSS und SASS

Um die Arbeit weiter zu reduzieren, wurden für CSS Präprozessoren entwickelt. Ein Präprozessor ist ein Programm, das in einer bestimmten Programmiersprache häufig wiederkehrende Aufgaben vorbereitet, sodass ein Entwickler diese nicht in voller Länge selbst programmieren muss. Für die beiden Präprozessoren ACSS und SASS gilt allerdings, dass Sie zunächst CSS beherrschen müssen, um sie sinnvoll nutzen zu können. SASS bietet dabei Möglichkeiten an, die einfache Kontrollstrukturen und andere Elemente imperativer Programmiersprachen einfügt und auf der dynamisch typisierten Sprache Ruby aufsetzt. Nicht zuletzt deshalb werden wir uns in diesem Kurs nicht mit CSS-Präprozessoren beschäftigen: Diese Möglichkeiten bietet uns bereits PHP an.

6.12.10 CSS und unterschiedliche Displayformate

Sie wissen, dass mittlerweile Geräte auf dem Markt sind, die jeweils die unterschiedlichsten Formate aufweisen. Vorhin haben Sie etwas über responsive web design gehört und erfahren, dass es hier nicht um ein gestalterisches Element geht, sondern darum, u.a. solche unterschiedlichen Formate in der Programmierung zu beachten. Denn dann werden die von Ihnen eingestellten Inhalte so präsentiert, dass jeder, der Ihre Webanwendung aufruft sie in einer ansprechenden Form präsentiert bekommt.

Bei CSS 2.1 (und damit unter HTML4) müssen Sie deshalb für jedes Ausgabeformat eine CSS-Deklaration programmieren. Hier drei Beispiele, von denen das letzte explizit die Breite des Gerätes beachtet:

¡link rel=ßtylesheet”type=”text/css”href=ßtandard.css”media=ßcreen”/¿

¡link rel=ßtylesheet”type=”text/css”href=”drucker.css”media=”print”/¿

¡link rel=ßtylesheet”type=”text/css”media=ßcreen and (max-device-width: 480px)”href=ßmartphoneMittelgross.css”/¿

6.12.11 Schriftsätze und –farben – CSS/font

Sie können auch Container skripten, die einen eigenen Schriftsatz mit individueller Schriftgröße und Schriftfarbe verwenden. Allerdings ist es wichtig, dass der Webserver und die üblichen Browser sowohl die Schriftart kennen als dass bei dieser Schriftart auch die Größe definiert ist, da sonst die Standardeinstellungen greifen und Ihr Layout somit über den Haufen geworfen wird.

Der Bezeichner für diesen Container lautet font. Im Gegensatz zu den Containern, die Sie bislang kennen gelernt haben, ist ein font-Container ohne zusätzliche Angaben im Tag recht sinnlos.

((Recherche: http://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Schriftformatierung/font))

Stichwortverzeichnis

Codierung, 28
Container, 16
CSS, 13

Doctype, 22
Doctype Definition, 23

HTML, 13
    Container, 16
    Doctype, 22
    Doctype Definition, 23

Internationalisierung, 28

Lokalisierung, 28

Markup Language, 13
Meta-Daten, 28

Programmiersprache
    CSS, 13
    HTML, 13
    XHTML, 21
    XML, 21

Responsive Design, 14

SelfHTML, 20
statisch, 21

Typisierung
    streng, 21

Use Case, 15

Validator
    W3C, 22
Validierung, 22

W3Schools, 20
WWW, 14

XHTML, 21