Bachelor-Thesis

Konzeption und Realisation einer responsiven Single-Page-Applikation nach dem Model-View-Controller-Ansatz unter Verwendung von Frameworks

Eine wissenschaftliche Arbeit von

Sascha Schnetz
Schumannstraße 5
77654 Offenburg

Betreuer

Prof. Dr. Roland Riempp

Zweitbetreuer

Fabian Fakir, B. Sc.

Fachhochschule Offenburg

Badstraße 21
77652 Offenburg

Sommersemester 2014

Hochschule Offenburg
University of Applied Sciences

M+I Medien und Informationswesen

I Inhaltsverzeichnis

„Zeiten der Ordnung sind die Atempausen des Chaos.“ - Walter Hilsbecher

II Über diese Bachelor-Thesis

„Die bis zum Erreichen des Ziels verbleibende Arbeit steigert sich mit dem Herannahen des Abgabetermins.“ - Edward A. Murphy

1 Vorwort

Im Rahmen meines Bachelor-Studiums Medien- und Informationswesen an der Hochschule Offenburg, habe ich einen Einblick in viele Bereiche der medialen Welt bekommen. Viele davon haben mich dabei sehr begeistert. Besonders der Bereich der Informatik und der Webentwicklung haben mich sehr fasziniert. Auf der Suche, nach einem für mich passenden und aktuellen Thema, wollte ich mich in einem Gebiet, in dem ich später auch beruflich Fuß fassen möchte, festigen.

Seit meinem 12. Lebensjahr erstelle ich Webseiten. Damals noch für die Musik-Band meiner Schule, später auch für Vereine und Narrenzünfte. Heute entwerfe ich Webseiten für kleine und mittelständische Unternehmen. Dabei habe ich seither immer statische oder schwerfällig selbst programmierte dynamische Seiten erstellt. Die Verwendung von Frameworks und das Programmieren von Single-Page-Applikationen hatte ich bisher gemieden. Nun ist es an der Zeit mich an diese Technologien heranzuwagen.

In dieser Bachelor-Thesis findet sich vieles, was mir in meinem bisherigen Webentwickler-Leben unbekannt war. Von der Entwicklung einer Single-Page-Applikation bis hin zur Implementierung aktueller Frameworks.

2 Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese wissenschaftliche Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Diese finden sich im Quellenverzeichnis im Anhang dieser Arbeit. Alle genutzten Bilder, die nicht im Quellenverzeichnis aufgelistet sind, wurden selbst erstellt.

Sascha Schnetz

III Das Projekt

„Die Virenproblematik ist nur ein temporäres Phänomen und wird in ein paar Jahren verschwunden sein.“ - John McAfee - Gründer von McAfee, 1988

3 Einleitung

Diese wissenschaftliche Ausarbeitung behandelt das Thema *Konzeption und Realisation einer responsiven Single-Page-Applikation nach dem Model-View-Controller-Ansatz unter Verwendung von Frameworks*. Dabei werden die aktuellsten und gängigsten Frameworks unter die Lupe genommen und deren Vorteile sowie Nachteile erörtert. Zu jedem Framework wird zusätzlich, soweit sinnvoll, ein Prototyp entworfen um die Funktionen direkt zu testen und sich ein Bild von den Funktionen machen zu können. Zum Schluss werden alle Frameworks miteinander verglichen und ein Fazit gezogen.

Anhand dieser Vergleiche wird anschließend eine Webseite für ein Unternehmen entworfen, welche ein ansprechendes Layout und eine zeitgemäße Technik verwendet sowie die Vorteile der vorgestellten Frameworks nutzt.

Der Hauptteil dieser Ausarbeitung stellt die Programmierung dar. Hier wird das konzipierte Projekt realisiert und im Anschluss auf verschiedenen Endgeräten getestet.

4 Ziel der Thesis

Das Ziel der Thesis ist es, einen Leitfaden zum Erstellen moderner Webseiten zu fertigen. Dieser Leitfaden soll sowohl für Anfänger als auch für alteingesessene Web-Gurus ein Nachschlagewerk bilden. Es ist jedoch leichter und besser verständlich, wenn man die Grundlagen der HTML-, PHP- und JavaScript-Programmierung kennt, da diese zu Umfangreich wären, um hier ausführlich beschrieben zu werden. Das übersichtliche Inhaltsverzeichnis dient der schnellen Themenfindung. Zur besseren Übersicht wurde der Inhalt in große Teilabschnitte gegliedert. Der Quellcode der Webseite findet sich auf der, im Anhang befindlichen, CD. Es gibt eine Online- (mit PHP) und eine Offline-Version (ohne Blog).

IV Frameworks

„In der Informatik geht es genau so wenig um Computer, wie in der Astronomie um Teleskope." - Edsger Wybe Dijkstra

5 Was sind Frameworks?

Frameworks sind von Informatikern zusammengestellte Sammlungen nützlicher Befehle, welche sich in das Projekt einbinden und z.B. über Klassen und Identifier ansprechen lassen.[1] Frameworks finden heutzutage in fast allen Webauftritten Verwendung, da es sehr mühsam wäre, alles selbst zu programmieren. Hier wird auf eine bereits erstellte Bibliothek zurückgegriffen um sich - wie sollte es anders sein - Arbeit, Zeit und Geld zu sparen.

Allerdings gibt es viele Webentwickler, welche Frameworks aus dem Weg gehen. Ein Grund dafür ist zum Beispiel die Größe. Es wird weitestgehend darauf geachtet, dass sie nicht zu „schwergewichtig" sind, allerdings kommen trotzdem an die 100-200 Kilobyte zusammen. Das klingt zwar nach wenig, aber in Zeiten des mobilen Internet, in welcher ständig unterwegs „gesurft" wird, sind 100 Kilobyte mit einer Edge oder 3G Verbindung eine bedenkliche Datenmenge. Wenn nur ein kleiner Teil eines Frameworks Verwendung findet lohnt sich die komplette Implementierung nicht.

Im Rahmen des zu erstellenden Projekts werden die zur Zeit aktuellen Frameworks Verwendung finden. Da es im Internet eine Überflut an Frameworks gibt, werden hier nur die gängigsten betrachtet (auf http://www.Github.com lassen sich aktuelle Trends verfolgen). Um auch hier an die mobile Generation zu denken, müssen diese nicht nur nach ihrem Können sondern auch nach ihrem Gewicht gewählt werden. Das Ziel ist es, nur die Frameworks zu verwenden, die wirklich benötigt werden und aus denen am meisten Nutzen gezogen werden können. So kann es durchaus vorkommen, dass man auf spezielle Features verzichten muss, da es sich nicht lohnen würde ein ganzes Framework, wegen beispielsweise eines besonderen Hover-Effekts[2], zu implementieren.

1 Vgl. http://www.itwissen.info/definition/lexikon/Framework-framework.html (Stand: 21.07.2014)

2 Ein Hover-Effekt bezeichnet den Effekt, welcher Auftritt, wenn man mit der Maus über ein bestimmtes Element, zum Beispiel einen Button, fährt.

So gut wie alle Frameworks laufen unter der MIT-Lizenz. Das ist eine gängige Lizenz unter Programmierern und Kreativen, welche die Verwendung des Produkts definiert. Die MIT-Lizenz sagt aus, dass die Software (hier die Frameworks) frei für alle Projekte verwendet werden dürfen, egal ob der Quellcode des Projekts frei einsehbar ist oder nicht. Software, welche der MIT-Lizenz unterliegen, dürfen kopiert, geändert, fusioniert, verlegt, verbreitet, unterlizenziert und/oder verkauft werden. Das macht die Lizenz für Software-Entwickler sehr attraktiv. Das erlaubt das Einbinden der Frameworks ohne Angaben der Herkunft und laut Lizenz auch deren uneingeschränke Verwendung und Abänderung. Hier die Lizenz im englischen Original[3]:

MIT-Lizenz ▶
Die MIT-Lizenz im englischen original

Copyright (c) <year> <copyright holders>
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the „Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED „AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

6 JavaScript-Frameworks

Welche Frameworks man schlussendlich verwendet sei jedem selbst überlassen. Die Frameworks, die hier vorgestellt werden, sind allerdings nicht willkürlich gewählt. Es wurde darauf geachtet, dass es in der Informatik

3 Auszug aus http://opensource.org/licenses/MIT (Stand: 07.05.2014)

etablierte und gern verwendete Frameworks sind und vor allem, dass die Frameworks über eine gute API[4] verfügen. Das ermöglicht den schnellen und unproblematischen Einsatz. Allerdings ist immer eine gewisse Grundkenntnis in JavaScript und HTML vorausgesetzt.

6.1 AngularJS

AngularJS ist ein sehr beliebtes und weit verbreitetes Framework aus dem Hause Google Inc. Es handelt sich hierbei um eine Umfangreiche Zusammenstellung von JavaScript-Befehlen, welche das clientseitige, dynamische Nachladen von Inhalten erlaubt.[5] So kann man in ein Textfeld einen Namen eintippen, und bei jedem Tastaturanschlag wird die

◀ Abbildung 1
Die offizielle Webseite von AngularJS

Ausgabe aktualisiert. Wozu ist das nötig? Das ist sehr von Vorteil, denn so kann man zum Beispiel eine Liste mit Einträgen filtern und bei jedem Buchstaben wird die Liste aktualisiert. Google verwendet diese Technik (oft „auto-complete“ genannt) bei der eigenen Suchmaschine oder der Videoplattform YouTube. So werden mögliche Suchbegriffe bei jedem neuen Buchstaben aktualisiert. Das ist aber nur eine von vielen nützlichen Funktionen.

Weitergehend ist AngularJS wie gemacht für eine Single-Page-Applikation, da AngularJS nach dem Model-View-Controller-Konzept program-

4 API (Application Programming Interface) bezeichnet die vom Hersteller zur Verfügung gestellte Dokumentation einer Software.

5 Vgl. https://cloud.google.com/developers/articles/angularjs-cloud-endpoints-recipe-for-building-modern-web-applications?hl=de (Stand 21.07.2014)

miert wurde und sich auf einfache Weise dynamische Inhalte nachladen lassen. Im folgenden Test wird ein Blick darauf geworfen, ob es möglich ist, eine dynamische Startseite zu bauen und den Inhalt mittels AngularJS nachzuladen.

6.1.1 Die Installation

AngularJS stellt bei der Installation keine Hürden auf. Wie viele andere Frameworks wird die Bibliothek im HEAD-Bereich des HTML-Dokuments entweder lokal oder über einen Web-Link eingebunden. Das wird oft gemacht, kann aber zu Problemen führen, sollte der andere Server zeitweise nicht erreichbar sein. Daher ist die lokale Verknüpfung auf dem eigenen Server zu Empfehlen.

Zuerst muss das JavaScript-Paket von der offiziellen Webseite[6] heruntergeladen werden. Wenn der Download gestartet wird, kann man zwischen drei verschiedenen Angular JS Versionen und Arten wählen und diese herunterladen. Die *Minified* (sehr kleine Dateigröße, da alle Leerzeichen und Umbrüche entfernt wurden), die *Uncompressed* (Für Entwickler; sehr Übersichtlich, hat aber eine höhere Dateigröße) und das *ZIP-Archiv*, in dem alles enthalten ist, was man brauchen könnte. Wir laden hier das ZIP-Archiv herunter. Nach dem Download entpacken wir das Archiv und erhalten eine regelrechte Flut an JavaScript-Dateien. Uns interessiert allerdings nur die oberste angular.js, die Anderen sind Erweiterungen (Plugins), welche wir eventuell noch zu einem späteren Zeitpunkt gebrauchen können.

Wenn wir nun AngularJS in unserem Prototyp-Projekt verwenden möchten, dann müssen wir die Datei aus dem ZIP-Archiv in unseren Projektordner laden und über den HEAD-Eintrag `<script src="lib/js/angular.js"></script>` verknüpfen. In diesem Fall muss die angular.js Datei im Unterordner `lib/js/` relativ zum HTML-Dokument liegen. „Relativ" bedeutet, dass der Pfad ausgehend von unserer HTML-Datei angegeben wird.

6.1.2 Die Funktionsweise

Das Framework arbeitet mit sogenannten Scopes. Ein Scope ist ein Gültigkeitsbereich des Models, welcher von AngularJS modifiziert werden kann.[7] Darüber hinaus kommt hier der Model-View-Controller-Ansatz zum tragen, denn die Programmlogik wird bei AngularJS in einem Controller ausgelagert. Das HTML-Dokument dient als Model (Gerüst) und der Webbrowser als View (Anzeige der fertigen Seite).

6 http://angularjs.org/ (Stand: 19.05.2014)

7 Vgl. https://docs.angularjs.org/guide/scope (Stand: 21.07.2014)

Wir legen nun im Ordner init (oft von Programmierern genutzt; kurz für initialisieren) ein JavaScript-Dokument an, welches unseren Controller darstellt und nennen es controller.js. Nun müssen wir dieses Dokument, wie die AngularJS-Datei, über das Script-Tag `<script src="init/crontroller.js"></script>` einbinden. Danach fangen wir mit der Programmierung des Prototypen an. Sollte der Überblick verloren gehen, so befindet sich auf der CD, im Anhang, der komplette Quelltext des Prototypen.

6.1.3 Der Prototyp

Wir verfügen nun über eine index.html, in welcher mehrere JavaScript-Dokumente eingebunden sind. Unser Prototyp soll schlicht und Übersichtlich sein und nur die Hauptfunktionen aufzeigen um nicht zu Umfangreich zu werden. Unser HTML-Grundgerüst index.html sieht nun wie folgt aus:

◀ Quelltext 1
Grundgerüst des Prototyps
Auf CD: /frameworks/angularJS/index.html

```
1  <!DOCTYPE HTML>
2  <HTML ng-app='app'>
3     <HEAD>
4           <meta charset='utf-8' />
5          <title>Angular JS</title>
6     </HEAD>
7
8     <BODY style='margin: 0;'>
9           <div ng-controller='Controller'>
10
11             <a href='#/home'>Home</a> |
12             <a href='#/apps'>APPs</a>
13
14             <div ng-view>
15                <!-- Hier wird der Inhalt geladen -->
16             </div>
17
18          </div>
19
20          <script src='lib/js/angular.js'></script>
21          <script src='lib/js/angular-route.js'></script>
22          <script src='init/module.js'></script>
23          <script src='init/controller.js'></script>
24          <script src='init/routing.js'></script>
25
26       </BODY>
27 </HTML>
```

Neben einem weitestgehend normalen Grundgerüst kommen hier zwei ungewöhnlichere Befehle vor. Der Code wird im Folgenden kurz erläutert. Danach werden die eingebundenen Skripte durchgegangen wodurch der Code besser verständlich werden sollte.

Wir haben in der ersten Zeile einen HTML5-Doctype gefolgt von dem HTML-Tag. Dieses HTML-Tag hat ein AngularJS-Attribut `ng-app`. Das ist nötig, um das erste Modul einer AngularJS Anwendung zu laden. Hier heißt unser zu beginn geladenes Modul einfach „app“. Nun folgt ein Standard-Header. Diesen könnte man dynamisch machen, aber um Codezeilen zu sparen kümmern wir uns hier nur um den Body-Bereich. In der späteren Webseite wird aber auch der Head-Bereich dynamisch sein.

Abbildung 2 ▶
Der AngularJS Prototyp erlaubt das Filtern in Echtzeit

Nachdem der Body-Bereich beginnt wird direkt ein DIV-Container geöffnet, welcher das Attribut `ng-Controller` enthält. Dieser Controller wird von uns geladen und ist durch die vorher angesprochene `ng-app` initialisiert worden. Der Controller enthält die Programmlogik und lässt uns alles innerhalb des Gültigkeitsbereichs, sofern wir das wollen, dynamisch ändern. Nach einer kurzen Navigation gelangen wir zum Hauptteil, welcher im Grundgerüst allerdings sehr mager aussieht. Der DIV-Container mit dem Attribut `ng-view` bezeichnet einen Bereich, welcher über das sogenannte Routing nachgeladen werden kann. Routing wird das Verweisen auf Templates bezeichnet.[8] Sprich, wenn wir `/apps` in die Adresszeile eingeben, soll die App-Ansicht geladen werden. Wenn wir `/home` oder nichts eingeben, soll die Startseite geladen werden. Da es sich hier um eine Single-Page-Applikation handelt, können wir nicht auf andere eigenständige HTML-Dateien verweisen, sondern das geschieht hier über das Routing. Dazu werden Templates genutzt, welche nichts anderes als externer HTML-Code sind, welcher hier eingefügt wird. Am Ende des Body-Tags werden nun noch alle benötigten Skripte eingebunden.

8 Vgl. https://docs.angularjs.org/tutorial/step_07 (Stand: 21.07.2014)

Das erste Skript ist das Angular.js-Skript, welches hier natürlich unverzichtbar ist, da es die Angular-JS-Befehle verwaltet. Das angular-route.js (ebenfalls im ZIP-Archiv) wird benötigt um das Routing zu ermöglichen. Danach werden die Module geladen. Diese werden oft genutzt um bestimmte Programmteile, welche thematisch nicht in den Controller passen, auszulagern. Prinzipiell ist es nicht nötig, es ist eher eine Philosophiefrage. Unser Modul macht nichts weiter, als die `ng-app` zu starten bzw. zu initialisieren. Das vorletzte Skript ist der Controller, er beinhaltet die Programmlogik zu jedem Untermenü und verarbeitet die Benutzerinteraktionen. Unser Controller wird im Beispiel verwendet, um die Templates zu verwalten und bestimmte Arrays oder Variablen für die entsprechenden Unterseiten zu definieren. In der Datei sind Controller für jede der Unterseiten definiert. So gibt es eine Funktion `homeCtrl` und eine `appsCtrl`. In der letzten Skriptdatei wird das Routing von uns festgelegt.

Es ist möglich, dass die Funktionen der Skripte schwer nachvolziehbar sind. Deshalb schauen wir uns die Dateien nun genauer an.

6.1.4 Module.js

```
1  var app = angular.module('app', ['app.ctrl', 'ngRoute']);
```

◀ Quelltext 2
Initialisieren unserer Applikation
Auf CD: /frameworks/angularJS/init/module.js

Das ist der ganze Inhalt der JavaScript-Datei. Hier wird ein neues Modul erstellt, welches im Prinzip unsere Webseite repräsentiert. Wenn wir in unserer index.html das HTML-Tag-Attribut `ng-app="app"` verwenden, wird exakt dieses Attribut angesprochen. In den Klammern wird zuerst der Name (‚app') vergeben und in den eckigen Klammern die zu ladenden Module. `app.ctrl` lädt unseren Controller, welcher unsere Programmlogik verwaltet. `ngRoute` ist ein von angular-route.js zur Verfügung gestelltes Plugin, welches für das Routing benötigt wird. Es ist Bestandteil der Datei angular-route.js.

6.1.5 Controller.js

Die noch überschaubare controller.js beinhaltet unsere Controller. Jede Unterseite besitzt einen eigenen Controller, der die Inhalte verwaltet. So gibt es hier Beispielsweise zwei Controller, einen für die Home-Seite (`homeCtrl`) und einen für die Unterseite der Apps (`apps-Ctrl`). Der `homeCtrl` hat hier nicht viel zu tun, da die Homepage in diesem Beispiel statisch ist. Interessant wird es bei dem App-Controller. Dieser beinhaltet ein Array, welches später ausgegeben werden kann. So lassen sich Projekte in Arrays speichern und diese werden einfach ausgegeben und lassen sich sogar mit AngularJS durchsuchen und filtern.

Quelltext 3 ▶
Der Home- und APPs-Controller
Auf CD: /frameworks/angularJS/init/controller.js

```
function homeCtrl($scope) {
}

function appsCtrl($scope) {
    $scope.showreels = [
        {
            name: 'Crazy Balloon',
            text: 'Bist du bereit für den verrücktesten
            Flug aller Zeiten?',
            plattform: 'Android'
        },{
            name: 'Aenigma',
            text: 'Lateinische Zitate spielerisch lernen.
            ',
            plattform: 'Android'
        },{
            name: 'Perfection',
            text: 'Weil Schönheit vergänglich ist.',
            plattform: 'WEB'
        }
    ];
}

var ctrl = angular.module('app.ctrl',[]).controller(
'Controller', appsCtrl);
```

6.1.6 Routing.js

Quelltext 4 ▶
Hier wird das Routing initialisiert
Auf CD: /frameworks/angularJS/init/routing.js

```
app.config(function($routeProvider, $locationProvider) {
  $routeProvider
        .when('/home', {
              templateUrl: 'templates/home.html',
              controller: 'homeCtrl'
        })
        .when('/apps', {
        templateUrl: 'templates/apps.html',
        controller: 'appsCtrl'
        })
        .otherwise({redirectTo: '/home'})
});
```

Wir sprechen hier wieder unsere `app` an und konfigurieren sie. Dabei wird eine Funktion aufgerufen welche den `routeProvider` abhört. Das ist der Bereich, der in der Adresszeile angehängt wird, wie zum Beispiel `http://www.meineSeite.de/rootProvider`. Hier wird abgehört, wenn (when) ein Ereignis eintritt, welches `/home` ist, dann lade die templateURL `templates/home.html` und lade den entsprechenden Controller `homeCtrl`; wenn (when) `/apps` übergeben wird, dann lade entsprechend die templateURL `templates/apps.html` und lade den Controller `appsCtrl`. Danach wird noch ein `otherwise` (anderenfalls) definiert, welcher wie ein aus der Informatik bekanntes `ELSE` fungiert. Wenn keines der Ereignisse zutrifft, dann muss man es so handhaben als wäre `/home` eingegeben worden.

Im Folgenden werden noch die zwei geladenen Templates beschrieben.

6.1.7 Home.html

```
1 <h1>Herzlich willkommen </h1>
2 <p>Ich bin ein Musterabsatz!</p>
```

◀ Quelltext 5
Simpler Inhalt, welcher via AngularJS geladen wird
Auf CD: /frameworks/angularJS/templates/home.html

Die Home.html ist wenig spektakulär. Sie dient dazu die Startseite anzuzeigen und beweist, dass das Routing auf die Startseite einwandfrei funktioniert.

6.1.8 Apps.html

```
1  <p style='margin: 30px 0 0 40px;'>Search: <input ng-
2  model='query'></p>
3  <ul ng-repeat='showreel in showreels | filter:query'
4  style='list-style: none;'>
5      <li>
6            <h1 style='margin-bottom: -10px;'>{{showreel.
7            name}}</h1>
8            <p style='margin-bottom: -15px;'>{{showreel.
9            text}}</p>
10           <p style='font-style: italic;'>{{showreel.
11           plattform}}</p>
12     </li>
13 </ul>
```

◀ Quelltext 6
Der Inhalt der Sektion APPs zeigt ein Showreel an.
Auf CD: /frameworks/angularJS/templates/apps.html

Hier wird es nun interessant, da wir hier wieder mit AngularJS arbeiten. An sich haben wir ein HTML-Gerüst, welches einen Paragraphen mit einer Suchmaske enthält und eine ungeordnete Liste ausgibt. Doch wenn wir genau hinschauen, sehen wir, dass hier wieder diese ng-Tags auftauchen.

Bei der Suche ist ein Input definiert, welches das Attribut `ng-model="query"` beinhaltet. Dieses erzeugt ein Suchfeld, welches wir mit unserer Liste verknüpfen.

Die Liste enthält im `ul-Tag` ein Attribut namens `ng-repeat`. Dieses Attribut wird den umschlossenen Inhalt so oft ausgeben, wie Elemente vorhanden sind. Die vorhandenen Elemente lesen wir mittels `showreel in showreels` aus unserem im Controller definierten Array. Danach folg ein Filter, welcher mit unserer Such-Maske (`query`) verknüpft ist. Wenn wir beliebige Eingaben in die Suchmaske eintippen, wird in Echtzeit nach jedem Buchstaben die Ausgabe aktualisiert. Das Listenelement enthält nun AngularJS-Variablen. Diese sind an den geschweiften Klammern `{{...}}` zu erkennen. Alles was zwischen doppelten geschweiften Klammern steht wird von AngularJS als Variable interpretiert und verarbeitet. Wir sprechen hier in unserem Showreel das jeweils aktuelle Element an und lassen uns den entsprechenden `.name`, `.text` und die dazugehörige `.plattform` ausgeben. So lassen sich sehr leicht Listen erzeugen, welche man über Filter schnell und einfach durchsuchen kann.

6.1.9 Fazit

Es braucht ein bisschen Zeit, bis man mit AngularJS und dessen Funktion zurecht kommt, dafür bietet es aber eine sehr große Palette an nützlichen Funktionen. Gerade das Laden der Inhalte über das Routing, das Verwenden von Variablen und das Filtern von Listen sind sehr nützliche Funktionen, welche dem späteren Web-Projekt sicher zu gute kommen könnten.

6.2 JQuery

JQuery ist das wohl bekannteste Framework und darf in dieser Liste nicht fehlen. Es handelt sich hierbei, wie auch AngularJS, um ein JavaScript-Framework. Es wird auch oft als AJAX-Framework[9] bezeichnet, da es die Kommunikation mit dem Server über AJAX sehr vereinfacht. Allerdings wäre es etwas überzeichnet, wenn man JQuery nur als AJAX-Framework bezeichnen würde, da es im Vergleich zu den vorhandenen Funktionen nur eine geringe Rolle spielt. JQuery wurde von vielen sehr engagierten Informatikern, über Jahre hinweg, programmiert und stetig erweitert. So ist JQuery eine Bibliothek (oder Sammlung), welche viele Funktionen bereitstellt. Eine weitere Stärke von JQuery ist die große Community, welche die Funktionen von JQuery nutzen um daraus sehr mächtige Plugins zu programmieren.

9 AJAX ist eine JavaScript-Technologie, welche das dynamische clientseitige Nachladen von Inhalten ermöglicht.

6.2.1 Die Installation

Wie schon AngularJS ist dieses Framework sehr einfach zu implementieren. Es reicht lediglich die JQuery-Dateien, welche sich über die offizielle Webseite **http://www.jquery.com/download** herunterladen lassen, in dem HTML-Dokument zu verknüpfen. Über die Zeile `<script src="lib/js/jquery-1.11.0.min.js"></script>` lässt sich das Framework einfach einbinden, sofern es in dem entsprechenden Verzeichnis abgelegt wurde. Danach kann es verwendet werden.

6.2.2 Die Funktionsweise

Es macht wenig Sinn, bei einer so umfangreichen Funktionsweise, wie sie JQuery bietet, einen Prototypen zu entwerfen. Stattdessen werden die Hauptfunktionen kurz aufgezeigt.

Möchte man mit JQuery arbeiten, und hat es erfolgreich in sein Projekt eingefügt, so braucht man erstmal ein Skriptbereich, in dem man mit JQuery arbeiten kann. Dieser kann sich direkt in der Seite befinden oder ausgelagert werden. Wird ein Skript in einer externen Datei ausgelagert, muss man es wie auch JQuery selbst über `<script src="meinScript.js"> </script>` einbinden.

Wie schon mehrmals erwähnt stellt JQuery viele nützliche Funktionen bereit, welche pures JavaScript auch bieten würde, aber viel aufwändiger zu programmieren wäre. So kann man zum Beispiel ein HTML-Element mit der ID `home` im Dokument über gewohntes JavaScript `document.getElementByID("Home");` oder über die JQuery-Funktion `$("#home");` ansprechen.

Das ist ein sehr triviales Beispiel. Es zeigt aber, dass JQuery schreibarbeit abnimmt und häufige Funktionen in eine kürzere Schreibweise transportiert. So gibt es dank JQuery sehr nützliche Funktionen wie `.val()`, welches den aktuellen Wert eines Objektes auslesen oder `.html()`, welches den HTML-Code austauschen kann. Besonders aufregend wird es dann aber bei CSS-Manupulationen, wie zum Beispiel die Funktion `animate()`. Diese Funktion ermöglicht es uns über einfache Parameter Objekte zu animieren. Um zu erkennen, wie mächtig JQuery dabei wirklich ist, hier ein kurzes Beispiel zu `animate();`

```
1 $('#myButton').click(function(){
2   $('#myImg').animate({
3     opacity: 0.5,
4     height: 80%
```

◀ Quelltext 7
Simples Beispiel für die animate()-Funktion

```
5    }, 5000);
6 });
```

Hier wird ein Button mit der ID `myButton` abgehört. Wenn dieser Button geklickt wird, soll das Bild mit der ID `myImg` animiert werden. Das Bild wird in seiner Deckkraft auf 50% gesetzt und die Höhe ändert sich auf 80%. Die Dauer der Animation wird in Millisekunden angegeben, d.h. der Wert 5000 gibt eine Länge von 5 Sekunden an.

6.2.3 Plugins

Plugins sind von der Community (Fangemeinde) erstellte Skripte. Diese enthalten Funktionen welche bestimmte Aufgaben erfüllen. Im Prinzip sind sie nichts anderes als eine sinnvolle Abarbeitung von JQuery-Funktionen.[10] Hat man nun ein Plugin richtig eingebunden, so kann man zum Beispiel ein Objekt mit dem Funktionsaufruf `.draggable();` so definieren, dass es mit der Maus bewegt werden kann. JQuery stellt dafür die benötigten Funktionen bereit, allerdings müsste man mehrere Zeilen Code dafür programmieren. Ein Programmierer hat hier ein Plugin mit der Funktion `draggable(){…}` erstellt, wodurch wir nur noch die Funktion aufrufen brauchen. Nach diesem Schema gibt es reichlich Plugins und wir können diese online testen, die besten herunterladen und nach dem Einbinden ohne Probleme verwenden. Man sollte allerdings darauf achten, dass man nicht zu übermütig wird sondern nur die Plugins einbindet, die man auch wirklich verwenden möchte. Denn jede JavaScript-Datei hat eine gewisse Dateigröße, die gerne vergessen wird.

Im Folgenden werden nun kurz die möglicherweise für unser Projekt relevanten und auch sehr beliebten Plugins vorgestellt.

JQuery UI Tooltip

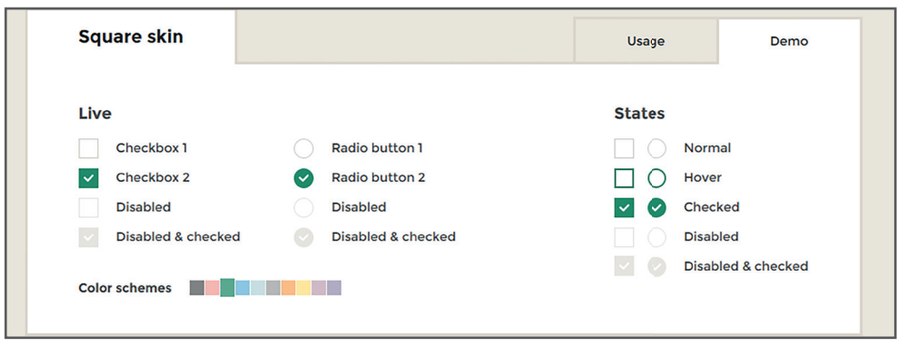
Das JQuery UI Tooltip ermöglicht es eigene Tooltips zu erstellen.[11] Das kann gerade bei einem Feedbackformular sehr sinnvoll sein, da es dem Nutzer beim darüberfahren nützliche Informationen bietet. Man muss allerdings beachten, dass so ein Hover-Effekt nur am Computer sichtbar ist und an mobilen Geräten nicht zum tragen kommt. Dieses Plugin steht unter der MIT-Lizenz und kann problemlos heruntergeladen und verwendet werden. Dazu muss die heruntergeladene JavaScript-Bibliothek einfach in das HTML-Dokument nach JQuery eingebunden werden. Über ein paar einfache Skriptbefehle lässt sich das Plugin verwenden.

10 Vgl. http://learn.jquery.com/plugins/ (Stand: 21.07.2014)

11 Vgl. http://plugins.jquery.com/ui.tooltip/ (Stand 21.07.2014)

iCheck

iCheck ist ein Plugin, welches uns gestattet die Checkboxen anzupassen. Dazu gibt es vorgefertigte Templates, welche man aber auch anpassen kann.[12] Diese Checkboxen haben einen sehr edlen Look und sehen auf allen Endgeräten gut aus. Es ist ein sehr kleines Plugin, welches aber gerade ein Kontaktformular oder eine Anmeldung optisch ansprechender gestalten lässt.



◀ Abbildung 3
Die iCheck-Demoseite bietet viele konfigurationsmöglichkeiten

Naver

Bei Naver handelt es sich um ein Plugin, welches eine einfache Navigation ermöglicht. Auf kleineren Bildschirmen fährt sie sich zusammen.[13] So lässt sich eine ansehnliche responsive Navigation erstellen. Ob dieses Plugin Verwendung findet lässt sich noch nicht sagen, aber es ist nicht verkehrt, wenn man es sich im Hinterkopf behält. Allerdings ist so eine Funktion schon in anderen Frameworks, wie Bootstrap, integriert.

Scroller

Scroller ist, wie der Name schon vermuten lässt, eine Erweiterung zum anpassen von Scrollleisten. So lässt sich Farbe und Form der Scrollleisten ändern und auf das eigene Design anpassen.[14]

Wallpaper

Dieses Plugin hilft beim Einbinden von Hintergründen. Der Vorteil ist, dass man Videos (zum Beispiel YouTube-Videos) in den Hintergrund einbinden kann, welche ohne Interface automatisch in Schleife abgespielt werden.[15] Dadurch lassen sich schöne, animierte Hintergründe erstellen, welche optisch sehr ansprechend sind. Der Nachteil ist, dass die Videos, je nach Bildschirmgröße, verpixelt sein können und auf mobilen Endgeräten Darstellungsfehler auftreten können. Auch muss das Video erst gepuffert werden, bevor es abspielt wird. Ist YouTube mal

12 Vgl. http://plugins.jquery.com/icheck/ (Stand: 21.07.2014)
13 Vgl. http://plugins.jquery.com/naver/ (Stand: 21.07.2014)
14 Vgl. http://plugins.jquery.com/scroller/ (Stand: 21.07.2014)
15 Vgl. http://plugins.jquery.com/wallpaper/ (Stand: 21.07.2014)

aus irgendwelchen Gründen nicht erreichbar oder langsam, so kann es durchaus vorkommen, dass kein Hintergrund angezeigt wird.

Accordion Image Menu
Das According Image Menu fasziniert von der ersten Nutzung an. Es handelt sich dabei um eine Navigation, welche über Bilder realisiert ist, welche sich beim darüberfahren aufziehen.[16] Viele kennen diese Art der Navigation. Es ist optisch sehr ansprechend, hat aber seine Nachteile.

Abbildung 4 ►
Das Plugin dient nicht nur als Navigation sondern gerne auch als Showreel

So ist die Navigation auf mobilen Endgeräten unübersichtlich. Man kann dieses Problem umgehen, indem man auf kleinen Geräten diese Navigation ausblendet und eine andere Einbindet. Man sollte dabei auch immer an die Performance denken, da die ausgeblendete Navigation trotzdem geladen wird.

Parallax Image Scroll
Seit einigen Monaten ist das parallaxe Scrollen in aller Munde. Es handelt sich dabei um einen optischen Effekt bei dem sich der Hintergrund in einer anderen Geschwindigkeit bewegt als der Vordergrund. Dieses JQuery-Plugin fügt genau diese Funktionalität hinzu.[17]

Sly (Slider)
Sly ist ein Slider, welcher Bilder in hunderten von Varianten darstellen kann.[18] So kann man mit diesem Slider die Neuigkeiten im Header oder eine Galerie in einem Showreel realisieren.

SlipHover (Image Hover Effect)
SlipHover Image ist ein sehr beliebtes und häufig gesehenes Plugin. Es handelt sich dabei um einen Hovereffekt. Dabei fährt ein leicht transparenter Balken genau von da über das Bild, wo man mit der Maus über

16 Vgl. http://plugins.jquery.com/accordionImageMenu/ (Stand: 21.07.2014)
17 Vgl. http://plugins.jquery.com/parallax-imageScroll/ (Stand: 21.07.2014)
18 Vgl. http://plugins.jquery.com/sly/ (Stand: 21.07.2014)

den Bildrand fährt, und blendet die Beschriftung ein. Die Bilder dienen dabei nicht als Galerie sondern als Links zu Unterseiten. Das Plugin kann zum Beispiel bei einem Portfolio genutzt werden um über Vorschaubilder ein Projekt anzuzeigen. Wenn man nun über ein Projekt fährt wird der Titel eingeblendet. [19]

6.2.4 Fazit

JQuery bietet alles, was das Programmierer-Herz benötigt. Diese Bibliothek ist fast schon ein Must-Have bei heutigen Webseiten. Bei dem breiten Angebot nimmt man auch gerne die recht große Datenmenge in Kauf. Gerade die zusätzlichen Plugins der Community sind einfach einzubinden und erweitern die Homepage um snaprechende und ansehnliche Funktionen.

7 CSS-Frameworks

Neben JavaScript-Frameworks gibt es auch CSS-Frameworks. CSS-Frameworks sind Stylesheets, welche Klassen bereitstellen, die wir in unser Dokument einbauen können. Diese Klassen sind dann in der CSS-Datei, z.B. von Bootstrap, definiert und haben somit bestimmte Eigenschaften.[20] Wenn wir mit dem Look dann nicht zufrieden sind, so können wir in einer eigenen CSS-Datei diese Eigenschaften nochmals überschreiben (kaskadierend). Dadurch werden CSS-Frameworks sehr attraktiv. Ein weiterer großer Vorteil ist, dass diese Frameworks heute großen Wert auf responsives Webdesign legen. So ist es einfacher eine Seite durch ein Framework für mehrere Bildschirmgrößen und Endgeräte zu optimieren.

7.1 Bootstrap

Bootstrap ist ein nicht ganz reines CSS-Framework und bietet viele nützliche Funktionen, welche die optische Erscheinung des Webauftritts beeinflussen. Dafür wird neben einer CSS-Datei auch eine JavaScript-Datei angeboten. Diese kann man zwar ignorieren dadurch verliert man aber viele nützliche Funktionen und Animationen, welche Bootstrap bietet.

Bootstrap ist nicht gerade ein Leichtgewicht. Die CSS-Datei ist allein schon 100 Kilobyte groß, die JavaScript-Datei ist dagegen mit ihren 30 Kilobyte relativ klein. Zusammen sind es dennoch 130 Kilobyte, die jedes Mal geladen werden müssen.

---

19 Vgl. http://plugins.jquery.com/sliphover/ (Stand: 21.07.2014)

20 Vgl. http://kritzelblog.de/erste-schritte-mit-bootstrap-3-0/ (Stand: 21.07.2014)

7.1.1 Die Installation

Die Installation von Bootstrap verläuft genau so einfach wie von bisher gewohnt. Man muss zuerst die CSS-Datei herunterladen und dann über `<link href="lib/css/bootstrap.css" rel="stylesheet" />` einbinden. Die JavaScript-Datei muss danach ebenfalls wie gewohnt über `<script src="lib/js/bootstrap.js"> <script>` eingebunden werden. Die Installation ist damit abgeschlossen. Die Frameworks können jetzt in vollem Umfang verwendet werden.

7.1.2 Die Funktionen

Bootstrap ist ein nützliches Framework, um sein Layout mit vorgefertigten Stilen für Schaltflächen, Listen, Navigationen und Ähnlichem zu versehen. Außerdem verfügt es über ein gutes Grid-System (Spaltenlaoyut), welches sich leicht einbinden lässt und womit einfach ein Grid-Layout realisiert werden kann. Über entsprechende Klassen kann man auch bestimmte HTML-Elemente bei bestimmten Bildschirmgrößen ein- und ausblenden. So kann man unwichtige Dinge bei schmalen Bildschirmen, wie zum Beispiel Smartphones, ausblenden, um den zur Verfügung stehenden Platz voll auszunutzen. Was zudem mit in das Framework eingebettet worden ist, ist eine Schriftart, welche alle möglichen Icons enthält. Diese lassen sich dann einfach über einen entsprechenden Befehl, zum Beispiel in Buttons, einbetten.

Neben dem nützlichen CSS-Teil ist auch ein JavaScript-Bereich in Bootstrap eingepflegt worden, welcher die Funktionalität nochmals erweitert. Allerdings reicht es meist nicht, nur das JavaScript von Bootstrap einzubinden. Man muss, je nachdem welche weiter Funktionalität man hinzufügen möchte, noch weitere JavaScript-Dateien einbinden. So lassen sich durch carousel.js wunderschöne Slider, oder mit collapse.js sehr ansehnliche Akkordeon-Effekte hinzufügen.

7.1.3 Der Prototyp

In diesem Abschnitt werden wir einen kleinen Prototyp entwerfen, welcher die wichtigsten Funktionen von Bootstrap kurz aufzeigt. Dabei sollen nicht alle Funktionen gezeigt werden, da der Quellcode einfach zu umfangreich werden würde. Es soll allerdings gezeigt werden, wie einfach es ist mit Bootstrap ein Grid-System zu bauen und wie die JavaScript-Funktionen von Bootstrap verwendet werden können. Außerdem zeigt das Beispiel, wie man einfach die vorgebauten Elemente wie Navigationen benutzt.

Den Anfang bildet ein HTML5 Grundgerüst. Wir brauchen hierzu die index.html, in welcher wir arbeiten. Dadurch wird es etwas übersichtlicher. Im Kopfbereich der HTML5-Datei befindet sich nichts außer Titel, Charset und der Verlinkung zu unserer Bootstrap JavaScrip-Datei über `<link href="lib/css/bootstrap.css" rel="stylesheet" />`. Spannender wird es hingegen im Bodybereich. Unser Prototyp soll über eine voll responsive Kopfnavigation und ein Formular verfügen.

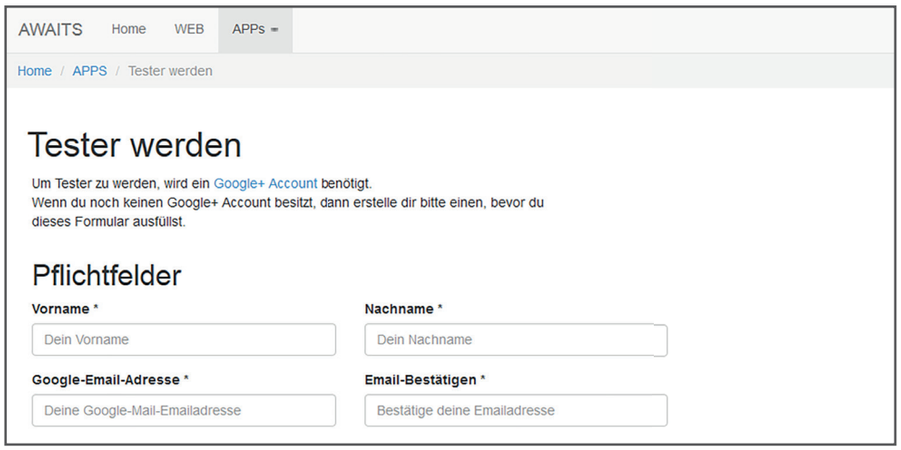
```
<nav class='navbar navbar-default' role='navigation'>
  <div class='container-fluid'>
        <div class='navbar-header'>
              <button type='button' class='navbar-
              toggle' data-toggle='collapse' data-
              target='#navbar-collapse-1'>
                    <span class='sr-only'>Toggle
                    navigation</span>
                    <span class='icon-bar'></span>
                    <span class='icon-bar'></span>
                    <span class='icon-bar'></span>
              </button>
              <a class='navbar-brand' href='index.html'>
                    AWAITS
              </a>
        </div>
        <div class='collapse navbar-collapse'
        id='navbar-collapse-1'>
              <ul class='nav navbar-nav'>
                    <li><a href='#'>Home</a></li>
                    <li><a href='#'>WEB</a></li>
                    <li class='dropdown active'>
                    <a href='#' class='dropdown-
                    toggle' data-toggle='dropdown'>
                    APPs <b class='caret'></b></a>
                    <ul class='dropdown-menu'>
                          <li><a href='#'>Übersicht</a>
                          </li>
                          <li class='divider'></li>
                          <li><a href='#'>Aenigma</a></li>
                          <li><a href='#'>Crazy Balloon</a>
                          </li>
                          <li class='divider'></li>
                          <li class='active'><a
                          href='#'>Tester werden</a></li>
```

◀ Quelltext 8
Ein Beispiel für die Bootstrap-Navbar. Auf CD: /frameworks/bootstrap/index.html

```
36                      </ul>
37                      </li>
38                  </ul>
39          </div>  <!-- /navbar-collapse -->
40    </div>  <!-- /container-fluid -->
41 </nav>
```

Grundlegend sehen wir hier eine Navigation, wie man sie bestimmt schon aus vielen anderen Projekten kennt. Bootstrap funktioniert über Klassen, welche dem Element vorgeschriebene Stil-Eigenschaften verleihen. So wird ohne viel CSS schon eine ansehnliche Navigation realisiert. Diese Navigation bietet mehr als man auf den ersten Blick sehen kann. Wird das Browserfenster schmäler, so wird aus dieser Navigation ein Smartphone-typisches Menü, welches man über einen Button ausklappen kann.

Abbildung 5 ▶
So soll unser Bootstrap Prototyp am Ende aussehen



Zuerst wird das nav-Element geöffnet und bekommt die Klassen `navbar` und `navbar-default`. Damit sagen wir dem Element, dass es sich um eine Navigation handelt und das Standard-Navigations-Layout verwenden soll. Durch `role="navigation"` sagen wir dem Element eindeutig, dass es eine Navigation ist. Danach können wir anfangen unsere Navigation innerhalb dieses Tags zu erstellen.

Nun kommt ein erster Abschnitt, mit dem wir den Fall der Smartphone-Nutzung behandeln. Hier wird durch einen DIV und die Klasse `container-fuid` ein Haupt-Knoten erstellt. Dann wird ein Header definiert, welcher bei der Smartphoneversion genutzt wird. Er ist durch den `navbar-header` (der nächste Knoten) definiert und enthält zum einen den Button, welcher die eigentliche Navigation aufklappt und zum Anderen das Logo, welches auf die Startseite verlinkt. Es fällt auf, dass hier ein But-

ton drei Mal den gleichen `span` enthält. Das liegt daran, dass die Typografie nur einen einzelnen horizontalen Strich im Sortiment hat. Wir sind aber gewohnt, dass der Button, um eine Navigation auszuklappen, 3 Striche untereinander anzeigt. Deshalb wird hier etwas getrickst. Wenn der Bildschirm groß genug ist, dann ist dieser DIV-Knoten nicht sichtbar, sondern nur der eigentliche Navigationsbereich, welcher sich im folgenden Knoten `<div class="collapse navbar-collapse">` befindet. Bootstrap macht das alles vollautomatisch und weiß zu jedem Zeitpunkt, wann es die Navigation normal und wann in der Smartphone-Variante anzeigen soll, ohne dass wir hier extra Klassen definieren müssen. In diesem Knoten befindet sich jetzt die eigentliche Navigation als Listenelement. Durch die Klasse und Eigenschaft `dropdown` lässt sich auch relativ einfach ein optisch ansprechendes und responsives Drop-Down-Untermenü erstellen. Auch weitere Hilfsklassen wie die Klasse `divider` (Trennstriche zwischen Unterpunkten) helfen uns bei der Individualisierung und Gestaltung unserer Navigation. Über die Klasse `active` können wir problemlos den aktuellen Menüpunkt auf „Aktiv" setzen, wodurch er etwas hervorgehoben wird.

Es kann vorkommen, dass man bei dieser Masse an Klassen und Verschachtelungen den Überblick verlieren kann. Hier hilft einem die offizielle API[21] weiter. Diese bietet neben einer genauen Anleitung auch viele Beispiele und Tutorials in denen gezeigt werden, wie das alles nochmals genau funktioniert.

Nach unserer Navigation wollen wir noch einen kurzen Breadcrumb (dt. Brotkrumen) erstellen. Ein Breadcrumb ist eine Pfadangabe, welche dem Nutzer zeigt, wo er sich gerade befindet. Der Name ist dabei eine Anlehnung an das Märchen „Hänsel und Gretel", welche sich eine Spur aus Brotkrumen gelegt hatten um zurückzufinden.

```
1 <ol class='breadcrumb' style='position: relative;
2 top:-20px;'>
3               <li><a href='#'>Home</a></li>
4               <li><a href='#'>APPS</a></li>
5               <li class='active'>Tester werden</li>
6 </ol>
```

◀ Quelltext 9
Beispiel einer einfachen Breadcrumb. Auf CD: /frameworks/bootstrap/index.html

Die Breadcrumb ist sehr übersichtlich und einfach gehalten. In einer gerodneten liste `<ol>` mit der Klasse `breadcrumb` wird der Pfad angegeben. Die aktuelle Position wird wieder mit der Klasse `active` hervorgehoben. Wie gewohnt einfach, schnell und effektiv zu verwenden.

[21] API findet sich auf http://getbootstrap.com/ (Stand: 21.05.2014)

Als nächstes müssen wir uns noch dem Formular widmen, damit unser Prototyp aus mehr als einer Navigation und einer Breadcrumb besteht. Das Formular ist in eine Section eingeschlossen. Das ist zwar nicht zwingend notwendig, ist aber gutes Programmierer-Jargon. Da das Formular sehr lang ist und unnötig Seiten füllen würde, sind hier nur die ersten zwei Eingabefelder als Beispiel zu sehen.

Quelltext 10 ▶
Die ersten zwei Eingabefelder des Formulars
Auf CD: /frameworks/bootstrap/index.html

```
<form role='form' style='max-width: 700px;'>
                  <div class='col-md-12'>
                    <h2>Pflichtfelder</h2>
                  </div>
                  <div class='form-group  col-md-6'>
                    <label>Vorname *</label>
                    <input type='email' class='form-
                    control'  placeholder='Dein
                    Vorname'>
                  </div>
                  <div class='form-group col-md-6'>
                    <label>Nachname *</label>
                    <input type='email' class='form-
                    control'  placeholder='Dein
                    Nachname'>
  </div>
   <!--Hier wurden die restlichen Formularfelder
   rausgekürzt  -->
  <div class='form-group col-md-12' style='margin-top:
  20px;'>
                  <button type='submit' class='btn btn-
                  default'>Abschicken</button>
                  </div>
</form>
```

Das Formular ist von einem Form-Tag umschlossen und sendet in diesem Beispiel keine Daten, sondern soll nur optisch implementiert werden. Zuerst kommt die Überschrift „Pflichtfelder“, gefolgt von dem ersten Eingabefeld. Jedes Eingabefeld bekommt die Klasse `form-group` und `col-md-6`. Dabei sagt `form-group` aus, dass es sich um ein Gruppenelement handelt. `col-md-6` erschafft ein Grid-Layout, wobei die Zahl `6` die Breite definiert. Bei Bootstrap hat man ein Grid-System aus 12 Spalten. Mit `col-md-6` sagen wir dem Element, dass es 6 Spalten breit sein soll. Immer relativ zum Elternelement. Da unsere Form maximal 700 Pixel breit ist, ist das Eingabefeld mit `col-md-6` maximal halb so breit (350 Pixel). Durch das Grid-System wird die Seite dazu voll responsiv.

Wird der Bildschirm kleiner rutschen die Eingabefelder automatisch untereinander. Jedes Element einer Form-Group wird durch ein Label, welches die Bezeichnung enthält, und das Eingabefeld (`input`) definiert. Am Schluss des Formulars wird dann noch ein Button zum verschicken der Daten eingefügt. Hier wurden wieder Bootstrap-Klassen verwendet, um den Stil anzupassen.

7.1.4 Fazit

In diesem Beispiel haben wir einen funktionsfähigen, optisch ansprechenden und sogar responsiven Bootstrap-Prototypen erstellt. Man darf nicht vergessen, am Schluss noch die nötigen JavaScript-Dateien einzufügen. Unser Projekt braucht die Bootstrap- sowie die JQuery-JavaScript-Datei. Hier lässt sich schon erahnen, wie eng Bootstrap und JQuery zusammenarbeiten. Viele Plugins und Funktionen benötigen beide Frameworks, da sie sich sehr gut ergänzen.

7.2 Pure

Pure ist, wie auch Bootstrap, ein responsives CSS Framework. Es hat allerdings den großen Vorteil, dass es eine sehr geringe Datengröße hat. Die Pure.css-Datei ist nur knappe 5 Kilobyte groß und ist somit ein sehr kleines CSS-Framework.[22] Es ist eine eindeutige Kampfansage an die Konkurrenz, welche mit dem 10-20-fachen an Datenmenge zurechtkommen muss. Es verfügt zwar nicht über besonders viele Spielereien, dafür bietet es dennoch ein responsives Grid-System und einige Styles für Buttons und Formulare. Sozusagen die wichtigsten Eigenschaften, die ein CSS-Framework benötigt, auf den Punkt gebracht.

7.2.1 Die Installation

Die Installation von Pure geschieht wieder über das Einbinden der CSS-Dateien. Diese werden über die Homepage http://purecss.io heruntergeladen und mittels `<link href="lib/css/pure.css" rel="stylesheet" />` eingebunden. Dabei gilt zu beachten, dass das zur Verfügung gestellte ZIP-Archiv keine „gesamte" CSS-Datei enthält, sondern nur einzelne CSS-Dateien für Grundlagen, Grid-Layout, Formulare, Buttons, Tabellen und Menüs beinhaltet. Man muss im späteren verlauf die benötigten Stylesheets entsprechend laden. Der Sinn dahinter ist, dass man sich weitere Kilobyte spart indem nur die benötigten Elemente geladen werden. Wenn man dennoch alle CSS-Dateien braucht, so lässt sich über die Verlinkung `<link rel="stylesheet" href="http://yui.yahooapis.com/pure/0.4.2/pure-min.css">` eine Sammlung aller Pure-CSS-Dateien einbinden. Das spart Codezeilen.

22 Vgl. http://purecss.io/ (Stand: 21.07.2014)

7.2.2 Die Funktionsweise

Pure funktioniert ähnlich wie die anderen CSS-Frameworks. Durch das Angeben von Klassen werden bestimmte, im Stylesheet definierte, Formatvorlagen angewandt. Dazu ist die offizielle Homepage (http://purecss.io) ein erster Anlaufpunkt. Man wählt dort in der Navigation das Element, welches man einbinden möchte - zum Beispiel eine Schaltfläche. Danach sucht man sich den Stil aus und kopiert die dort angegebenen Codezeilen oder übernimmt direkt nur die Klasse. Pure bietet dabei sogar mehrere Arten der Einbindung an. So kann man entweder das Button-Tag direkt anpassen oder ein Link-Tag als Schaltfläche definieren. Der optische Effekt ist dann der gleiche. Allerdings hat man mehr Freiheiten.

Variante 1 (Link):

```
<a class="pure-button" href="#">Pure Button</a>
```

Variante 2 (Schaltfläche):

```
<button class="pure-button">Pure Button</button>
```

7.2.3 Der Prototyp

Würden wir an dieser Stelle einen Prototypen entwerfen, so wäre er weitestgehend analog zu dem Prototypen von Bootstrap, weshalb hier nur ein kurzer Ausschnitt aus dem Pure-Prototypen vorgestellt werden soll. Der Rest ist altbekannt.

Quelltext 11 ▶
Menü des Pure-Prototyps
Auf CD: /frameworks/pure/index.html

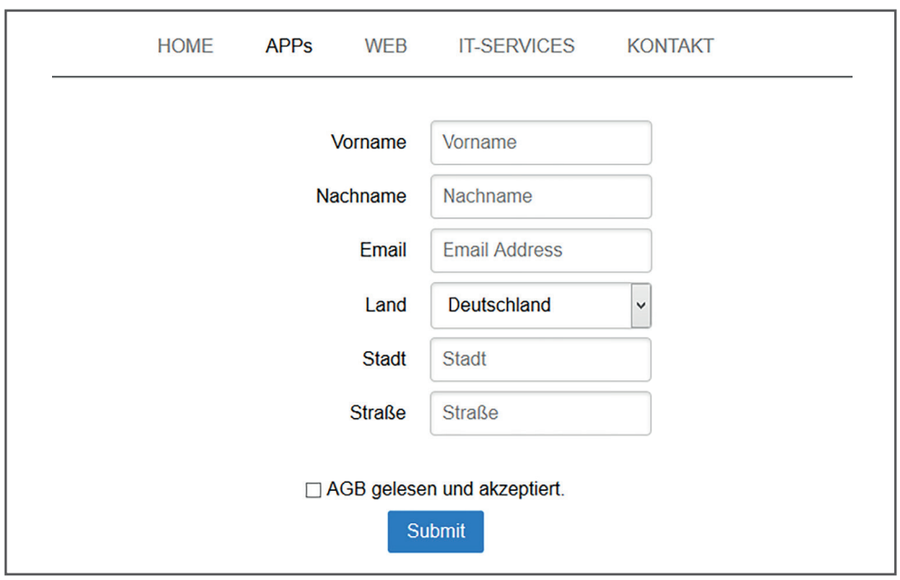
```
<div class='pure-menu pure-menu-open pure-menu-horizontal' style='text-align: center;'>
            <ul>
                <li><a href='#'>HOME</a></li>
                <li class='pure-menu-selected'><a href='#'>APPs</a></li>
                <li><a href='#'>WEB</a></li>
                <li><a href='#'>IT-SERVICES</a></li>
                <li><a href='#'>KONTAKT</a></li>
            </ul>
</div>
<hr/>
```

Hier sieht man, wie einfach es ist, ein Menü mithilfe von Pure zu realisieren. Man muss nur eine Navi nach bekanntem Listen-System bauen und die entsprechende Klasse in den umgebenden Container einbauen. Die Navigation ist zwar nicht spektakulär, bietet aber alle Grundfunktionen und ist responsiv.

Bei den Formularen ist es sehr angenehm mit Pure zu arbeiten, da es nur die Klassenangabe im umschließenden Form-Tag benötigt. Wenn das Framework weiß, dass es ein Formular ist, so werden alle enthaltenen Elemente automatisch nach dem neuen Stil formatiert, ohne jedes Mal anzugeben, dass es sich um ein Pure-Eingabefeld handelt. Wenn man das ganze responsiv möchte, muss man die Eingabefelder in Container mit entsprechenden Klassen einbetten.

```
<form class='pure-form pure-form-aligned' style='text-
align: center; margin-top: 30px;'>
            <fieldset>
                <div class='pure-control-group'>
                    <label for='vorname'>Vorname</label>
                    <input id='vorname' type='text'
                    placeholder='Vorname'>
                </div>

                <div class='pure-control-group'>
                    <label for='nachname'>Nachname
                    </label>
                    <input id='nachname' type='text'
                    placeholder='Nachname'>
                </div>
            <!--Hier wurde Code entfernt -->
            <button type='submit' class='pure-button
            pure-button-primary'>Submit</button>
        </fieldset>
</form>
```

◀ Quelltext 12
Pure macht es einfach ein Formular oder Container responsive zu machen Auf CD: /frameworks/pure/index.html



◀ Abbildung 6
Unser Prototyp sollte nun so aussehen

7.2.4 Fazit

Pure ist ein Leichtgewicht, das ist unumstritten. Doch leider wurde diese Leichtigkeit mit fehlender Funktionalität erkauft. Es wurde darauf geachtet, dass nur das Nötigste eingebunden wird. Wenn man eine kleine Webseite bauen möchte, die genau den Pure-Look haben soll, dann ist es bestimmt Sinnvoll, Pure zu verwenden. Auch die Möglichkeit nur das Menü-CSS-Stylesheet einzubauen und somit mit unter 5 Kilobyte eine responsive Navigation einzubauen, ist sicherlich ein großer Pluspunkt für das Framework. Wenn die Seite jedoch umfangreicher werden soll, stößt man mit Pure schnell an die Grenzen des realisierbaren.

7.3 Topcoat

Topcoat ist eine sehr umfangreiche von Adobe veröffentlichte Sammlung verschiedener Web-Elemente. Es wurde dabei besonders darauf geachtet, dass die Ressourcen nicht strapaziert werden, was das Framework sehr sinnvoll erscheinen lässt.

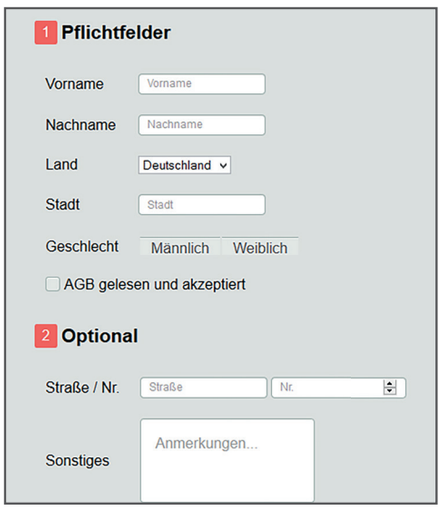
Topcoat bietet ansehnliche CSS-Layouts für wenig Datenmenge. Allerdings hat es nichts zu bieten, was es nicht schon in anderen Frameworks gibt. Adobe hat hier besonders auf die Performance geachtet und nur das in ihr Framework eingebettet, was für sinnvoll erachtet wurde. Dennoch misst die Datei in komprimierter Variante für Desktop und Mobil gute 68 Kilobyte. Adobe wirbt auch mit der großen Palette an Webschriften auf der Homepage von Topcoat – allerdings müssen diese extra eingebettet werden und setzen Topcoat nicht voraus. Als Alternative bietet auch Google mit Google-Fonts eine beachtliche Sammlung an Web-Schriften an.

7.3.1 Die Installation

Die Installation ist ähnlich einfach wie schon bei den anderen Frameworks. Das Zip-Archiv kann unter http://topcoat.io heruntergeladen werden und wird in einen entsprechenden Ordner in unserem Projektverzeichnis abgelegt. Bevor wir Topcoat einbinden, müssen wir uns für eine der Style-Dateien entscheiden. Es gibt 4 verschiedene Dateien, je eines für eine helle und eine dunkle Oberfläche und noch jeweils für Mobil oder Desktop Anwendung. Wenn man nur eines braucht, macht das die Sache sehr performant, wenn man aber eine multi-Plattform Anwendung schreiben möchte, muss man mehrere Stile einbinden, was auch wieder mehrere Codezeilen und höhere Ladezeiten bedeutet. Über die Verlinkung `<link href="lib/css/topcoat-desktop-light.css" rel="stylesheet" />` lässt sich dann zum Beispiel die Desktop-Version mit heller Oberfläche einbetten.

7.3.2 Die Funktionsweise

Das Framework funktioniert ähnlich wie die anderen Frameworks über Klassen-Bezeichner. So kann man einen Button mit der Klasse `topcoat-button--large--cta` versehen, um einen schönen Adobe-Button zu kreieren. Nachteil ist die nicht ganz so schlüssige Bezeichnung der Klassen. So lassen sich jedoch, wie in den anderen Beispielen, verschiedene Layouts realisieren.



◀ Abbildung 7
Das Framework Topcoat bietet uns die Möglichkeit mit CSS-Klassen einen „Adobe"-Look zu erzeugen

7.3.3 Webschriften

Die Webschriften sind nicht wirklich ein Feature von Topcoat, da sie auch ohne das Framework eingebunden werden können. Dennoch soll hier kurz erläutert werden, wie man Webschriften importiert.

Eine Webschrift sucht man sich entweder auf Adobes Web-Plattform „Adobe Edge Web Fonts" oder bei „Google Fonts" aus. Das Einbinden der Schriften funktioniert unabhängig davon auf die gleiche Weise.

Nachdem man sich eine Schrift mit dem übersichtlichen Nutzerinterface ausgesucht hat, muss man sich entscheiden welche Schriftschnitte man braucht. Dabei gibt es verschiedene Versionen wie Regular, Italic, Bold, Medium, Denim, Book, Light, usw. Im Anschluss wird eine JavaScript- und eine CSS-Zeile angezeigt. Diese müssen beide in das Webprojekt eingebunden werden. Die JavaScript-Zeile lädt die Schrift und die CSS-Zeile bindet diese dann in das Projekt ein. Das geschieht über das Nachladen der Schriften mittels Font-Face. Bei einigen Browsern kann man das Laden von Schriften mittels Font-Face blocken, daher sollte man mögliche Alternativ-Schriften angeben, um im Notfall eine gute Darstellung zu gewährleisten.

7.3.4 Fazit

Dieses Framework ist nicht besonders groß, bietet aber auch nicht besonders viel. Es glänz mit seiner Leistung, aber wegen des geringen Funktionsumfangs traut man sich fast nicht, das Framework einzubinden. Die Bezeichner der Klassen sind nicht immer schlüssig und müssen ständig nachgelesen werden. Ein Vorteil ist hier aber die Zusammenarbeit mit den Adobe-Produkten. So sind im Dreamweaver alle Klassen aufgelistet und man kann sich so einen besseren Überblick verschaffen.

Das ist auch nötig, denn eine wirkliche API mit allen Funktionen gibt es nicht. Es gibt nur eine Online-Demo, welche aber nicht wirklich viel Einblick in die Möglichkeiten bietet. Leider ist das Layout nicht responsiv sondern statisch, da es nicht über ein Grid-Layout oder ähnliches verfügt. Das müsste man selbst programmieren oder durch andere Frameworks realisieren. Schlussendlich bekommt man also ein Framework, welches schnell und nicht all zu groß ist. Es bietet dafür aber nur wenig verschiedene Stile und ist nicht responsiv.

8 Weitere Frameworks

Es gibt noch etliche weitere Frameworks, allerdings ist hier weder Zeit noch Platz um alle zu durchleuchten. Zum Beispiel gibt es neben JavaScript- und CSS-Frameworks auch sogenannte PHP-Frameworks. Dabei handelt es sich um mächtige serverseitige Frameworks, welche die Handhabung mit PHP-Typischen Funktionen wie einem Login, einem Gästebuch oder Ähnlichem erleichtern. Meistens ist die Handhabung und Installation aber sehr aufwendig und umfangreich, wodurch es oft keinen Sinn macht, ein PHP-Framework zu implementieren, wenn man es nicht wirklich benötigt. In unserem Fall werden wir kein PHP-Framework benötigen, da wir den nötigen PHP-Code selbst schreiben werden. Dennoch soll im folgenden Abschnitt kurz das bekannteste PHP-Framework Zend vorgestellt werden.

8.1 Zend Framework

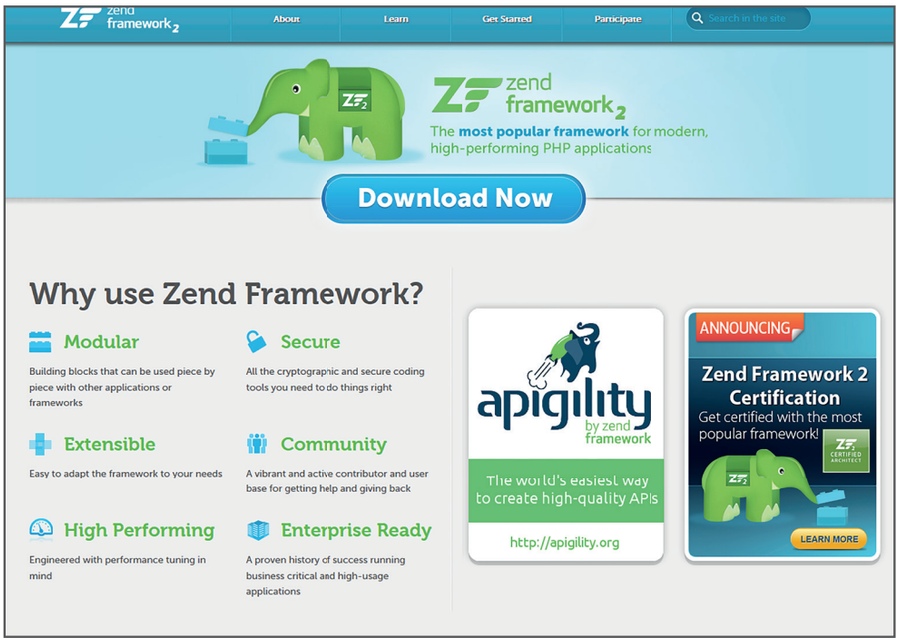
Zend Framework ist das wohl bekannteste und populärste PHP-Framework, welches sich zurzeit im *World Wide Web* finden lässt. Es handelt sich um ein Framework, welches auf Performance setzt und sich durch seine Modularität und Sicherheitsmodule einen Namen gemacht hat.[23] Auch eine engagierte Community hat sich um das beliebte Framework gesammelt, welche mit reichlich Tutorials und Hilfe-Foren auf fragende Programmierer wartet.

Auf http://framework.zend.com findet sich neben vielen interessanten Informationen und Tutorials auch eine rund 1700 Seiten umfassende Dokumentation.

Doch was ist Zend und wofür braucht man es? Das PHP eine mächtige serverseitige Sprache ist, das wissen die meisten Programmierer. Mit PHP lassen sich Logins, Blogs und vieles mehr realisieren. Da es ser-

23 http://framework.zend.com (Stand: 25.05.2014)

verseitig arbeitet bietet es darüber hinaus auch ein gewisses Maß an Sicherheit. Doch diese ganzen Funktionen selbst zu programmieren kann einem Informatiker manchmal neben dem letzten Nerv auch die letzte freie Minute rauben. Hier kommt Zend ins Spiel. Dieses mächtige Framework stellt eine Sammlung von Funktionen bereit, welche helfen, komplexere und mächtigere PHP-Anwendungen zu realisieren. Sei es die Kommunikation mit Datenbanken oder das Anlegen verschiedener Module.



◀ Abbildung 8
Der offizielle Webauftritt des ZEND-Frameworks

Zend ist ein objektorientiertes Framework und basiert auf dem Model-View-Controller-Konzept, welches unserem Projekt zugute kommen würde, da wir diesen Ansatz als Programmiergrundlage gewählt haben.

Da wir in unserem Projekt allerdings kein PHP-Framework brauchen werden, wird dieser Bereich nicht weiter bearbeitet. Der Nutzen rechtfertigt hier nicht den Aufwand der Einrichtung und die komplizierte Verknüpfung der verschiedenen Frameworks mit einem PHP-Framework.

9 Das richtige Framework finden

Wir haben nun mehrere CSS- und JavaScript-Frameworks vorgestellt. Darunter sehr bekannte Frameworks wie Bootstrap und JQuery, aber auch Neulinge und kleinere Frameworks wie Pure oder Topcoat.

Nun geht es darum, geeignete Frameworks zu finden, um unser Projekt so funktionieren und erscheinen zu lassen, wie wir uns das vorstellen. Dazu werden nochmal kurz die Highlights und der mögliche Nutzen der einzelnen Frameworks aufgezeigt.

9.1 JavaScript-Frameworks auf den Punkt gebracht

Wir haben zwei sehr bekannte und beliebte JavaScript-Frameworks vorgestellt und deren Funktionen und Nutzen ermittelt. Das war zum einen das sehr funktionale AngularJS-Framework und das ebenfalls sehr funktionale und beliebte Framework JQuery. Beides sind JavaScript-Frameworks, allerdings unterscheiden sie sich sehr in ihrer Funktionalität. JQuery nutzt eine rießige Bibliothek an Funktionen um einem Programmierer Arbeit abzunehmen. Andere Programmierer haben das erkannt und Plugins für JQuery geschrieben, welche sich einfach einbinden lassen und großen Nutzen bringen. AngularJS hingegen hat das Ziel eine Seite dynamischer zu machen. Es erlaubt das Nachladen von Inhalten, ohne dass die Seite neu geladen werden muss. Dabei wird der Model-View-Controller-Ansatz verwendet, um eine View zu erstellen, den Inhalt aber über Controller zu laden. Das Zusammenspiel mit AngularJS und JQuery könnte hier nicht besser sein. Das eine Framework ergänz das Andere, wodurch man hier eine sehr mächtige Komposition aus JavaScript-Funktionen erhält. Es wird darauf hinaus laufen, dass wir beide Frameworks implementieren werden. So erhalten wir durch AngularJS den mächtigen One-Pager nach dem Model-View-Controller-Konzept und dank JQuery viele Funktionen, um die Seite nach unseren wünschen zu gestalten. Animierte Navigationen, Formulare und viele aufwendige Funktionen werden Besucher der Seite faszinieren.

9.2 CSS-Frameworks auf den Punkt gebracht

CSS-Frameworks findet man im Internet wie Sand am Meer. Mit der Zeit haben sich aber sehr beliebte und begehrte Frameworks herauskristallisiert. Ein sehr bekanntes, nämlich Bootstrap, wurde hier vorgestellt. Außerdem haben wir uns den sehr leichtgewichtigen Konkurrenten Pure und das noch eher unbekannte Framework Topcoat angeschaut. Drei Frameworks, drei verschiedene Stärken. Bootstrap ist das All-In-One-Paket, welches allerdings einige Kilobyte auf die Waage bringt. Daneben wirkt das perfomante Pure wie eine Feder – allerdings auf Kosten von Funktionalitäten. Wer es sehr klein und übersichtlich möchte, der ist bei Adobes Topcoat gut aufgehoben. Dieses Framework bietet eine sehr überschaubare Palette an Funktionen, in der man sich schneller zurecht findet. Darüber hinaus ist eine gute Adobe Dreamweaver Unterstützung gegeben.

Es fällt schwer, hier ein geeignetes Framework zu finden. Da wir aber auf kurz oder lang die Plugins von JQuery verwenden werden (sei es für eine Bildergallerie oder die responsiven Funktionalitäten), werden wir um dieses Framework nicht herumkommen. Die anderen Frameworks würden uns keinen zusätzlichen Nutzen bieten. Pure und Topcoat sind zwar nützliche Frameworks, sind aber eher für kleine Webauftritte wie Online-Visitenkarten oder Blogs zu empfehlen. Ein Internetauftritt für ein Unternehmen sollte aber an Seriosität und Funktionalität nicht sparen.

V Konzeption

„Hardware nennt man die Teile eines Computers die man treten kann."
- Jeff Pesis

Bei dieser Thesis soll ein Webauftritt für ein Kleinunternehmen erstellt werden. Exemplarisch dient hier das Unternehmen Awaits (kurz für **A**dvanced **W**eb **A**nd **IT-S**ervices). Für dieses werden im Folgenden ein kurzes Corporate Design mit Logo und Gestaltungsrichtlinien sowie die benötigten Funktionen des Webauftritts festgelegt.

10 Das Kleinunternehmen

Das Unternehmen soll sich, wie sich schon erahnen lässt, mit der Erbringung von Computer-Dienstleistungen (IT-Services) sowie der Erstellung von Webseiten und Apps als Unternehmen definieren. Hierfür ist vor allem ein guter und informativer Webauftritt wichtig, welcher auch als Portfolio dienen soll. Da dieses Unternehmen ganz frisch in den Startlöchern steht, gibt es noch keine Gestaltungsrichtlinien (Corporate Design). Diese werden wir nun kurz erstellen, damit wir bei der Gestaltung der Webseite einen gestalterischen Leitfaden haben, an dem wir uns entlanghangeln können.

11 Corporate Design

Das Corporate Design dient zur Festlegung von Gestaltungselementen aller Art für ein Unternehmen. Dabei wird nicht nur das Logo und der Schriftzug beachtet, sondern auch Kleiderordnung, Visitenkarten, Rechnungslayouts, Formularvorlagen, etc. Hier soll allerdings nur das Wichtigste festgelegt werden.[24]

11.1 Der Name

Awaits – Advanced Web And IT-Services

Bei dem Firmennamen und dem Claim sind die ersten Buchstaben immer großzuschreiben. Der Name Awaits darf auch ohne Claim geschrieben werden, wenn es aus gestalterischen Gründen notwendig ist.

24 Vgl. http://www.typolexikon.de/c/corporate-design.html (Stand: 21.07.2014)

11.2 Die Farben

Die Farbgestaltung dient als Wiedererkennungsmerkmal und ist ein elementarer Bestandteil eines jeden Unternehmens. Auch beim Unternehmen Awaits sind die Farben fest vorgegeben.

Awaits-Cyan

Gestaltungs- und Schmuckelemente

50% Cyan | RGB (131/208/245) | CMYK (50/0/0/0) | Hex (#83D0F5)

Alternativen

Tabellen, Hintergründe, Überschriften

Alternative 1
Hex (# 626F75)

Alternative 2
Hex (# 3F6475)

Alternative 3
Hex(# 68A5C2)

Alternative 4
Hex (# CEE9F7)

Grau

Gestaltungs- und Schmuckelemente, Überschriften, Fließtext

80% Schwarz | RGB (87, 87, 86) | CMYK(0/0/0/80) | Hex(# 575756)

Abstufungen des Grau

Tabellen, Hintergründe

Abstufung 1
80% Farbton

Abstufung 2
60% Farbton

Abstufung 3
40% Farbton

Abstufung 4
20% Farbton

11.3 Die Schriften

Folgende Schriftformatierungen dürfen für Print- und Screen-Medien verwendet werden:

Hauptschrift

Lato

Hairline (Gestaltungs- und Schmuckelement)
Light (Fließtext)
Regular (Fließtext/Überschrift)
Bold (Überschrift)

Alternative Schriftart

Arial

Regular (Fließtext)
Bold (Überschriften)

Schriftfarbe

Die Standard-Schriftfarbe ist das im Bereich Farben definierte Grau (# 575756).

Schriftgrößen

Die Standardschriftgröße in Fließtexten ist 10 oder 12 Pt.

Die Standardschriftgröße in Bildunterschriften, Hinweisen, Infoboxen und Zitaten beträgt 8, 10 oder 12 Pt.

11.4 Das Logo

Das Logo sollte ein einmaliges Wiedererkennungsmerkmal eines jeden Unternehmens sein. Es ist explizit darauf zu achten, dass das Logo richtig platziert ist und die Gestaltungsrichtlinien nicht verletzt wurden. Es stehen 3 verschiedene Logo-Varianten zur Verfügung.

Logo ohne Claim

Das Logo ohne Claim darf verwendet werden, falls der Claim unangebracht ist oder sich mit einer nahen Schrift beißen würde.

Logo mit Claim

Das Logo mit Claim soll, falls möglich, immer verwendet werden. Der Claim ist in Schreibweise und Schriftart fest vorgegeben.

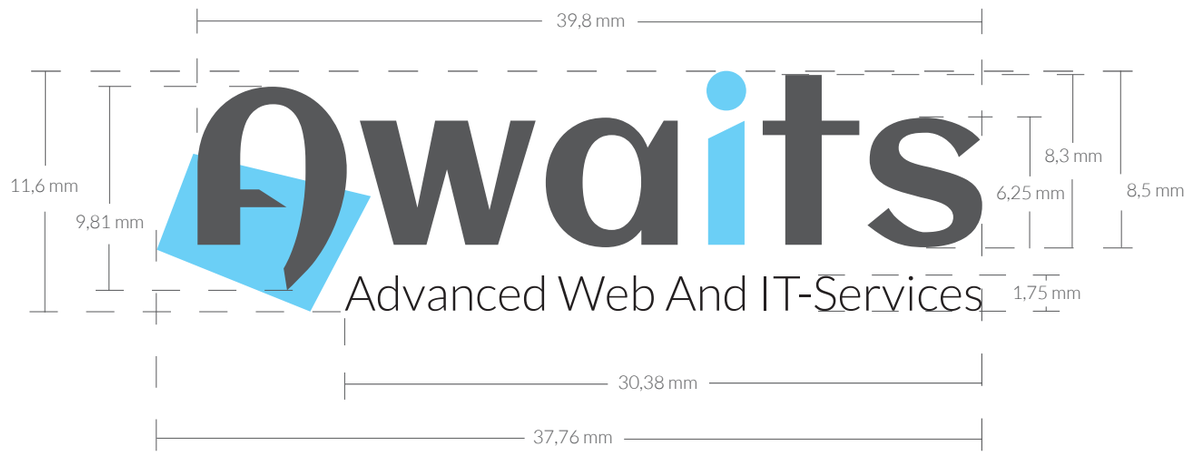
Logo Kurzform

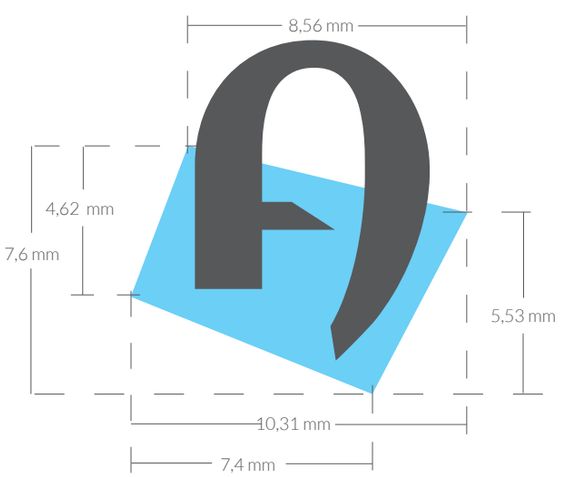
Das Logo in Kurzform dient neben den ausgeschriebenen Logos als Alternative in einem anderen Format. Das Logo wird hauptsächlich auf Rückseiten von Flyern, auf Visitenkarten oder ähnlichem Verwendung finden.

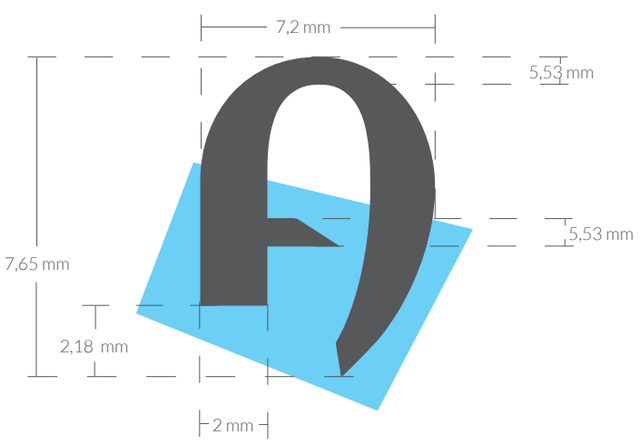
Das Logo in Kurzform sollte dabei nur verwendet werden, wenn die anderen Logos unangepasst wären.

11.5 Die Maße des Logos

Hier sind die genauen Maße und Verhältnismäßigkeiten aufgezeigt, nach denen die Logos zu gestalten sind.







12 Inhalte der Webseite

Die Webseite soll der Firmenauftritt eines Kleinunternehmens sein, welches Web- und IT-Dienstleistungen erbringt. Auch App-Programmierung soll zum Leistungsumfang des Unternehmens zählen. Die Webseite soll die bisherigen Projekte als Showreel darbieten und dem Kunden auch Informationen wie Preise und Kontaktdaten an die Hand geben.

Die Homepage sollte über die Bereiche HOME, WEB, APPs, IT-SERVICE, BLOG und ein IMPRESSUM verfügen.

HOME soll den Besucher willkommen heißen und ihm kurz aufzeigen, wer der Unternehmer ist und was für Qualifikationen dieser besitzt. Außerdem soll auf die Unterseiten verwiesen werden. Des Weiteren befindet sich in dieser Sektion ein Kontaktformular, damit Kunden direkt Kontakt aufnehmen können. Am Ende der Seite sollen Links zu Partnern untergebracht werden.

Der Abschnitt WEB verfügt über mehrere Unterpunkte. Wenn man auf den Bereich WEB klickt wird ein zuerst ein Showreel angezeigt. Jedes Projekt wird hier direkt verlinkt. In einem Dropdownmenü findet sich dann neben den Projekten ein Kalkulator, welcher den potentiellen Kunden die Kostenermittlung ihrer Homepage erleichtert. Der Kalkulator soll verschiedene Auswahlmöglichkeiten zum personalisieren des Angebotes bieten und den Preis immer in Echtzeit anzeigen.

APPs dient ähnlich wie WEB vor allem zum aufzeigen der bisherigen Projekte. Hier gibt es kein Kalkulator, da die Preise einfach zu unterschiedlich und projektbezogen sind. Dafür soll es einen Infobereich geben, in dem der Kunde die verschiedenen Informationen zur Preisabgabe einsehen kann..

IT-SERVICES wird kein Showreel bieten, da es hier hauptsächlich um Wartung, Beratung und Reperaturen geht.

Der BLOG ist der Magnet, welcher Besucher auf die Seite zieht. Er wird so aufgebaut sein, dass jeder Artikel eine Überschrift und ein Datum enthält. Danach wird ein einleitender Absatz in den Artikel (eventuell mit Bild) zu sehen sein. Über den Link „Weiterlesen“ wird man zum kompletten Artikel weitergeleitet. Es soll auch eine Möglichkeit zur Filterung der Einträge nach Erscheinungsjahr geben.

Zusammenfassend soll die Navigation der Seite wie folgt aufgebaut sein:

HOME (Dropdown)

Über Mich	Kurze Vorstellung des Unternehmers
Kenntnisse	Balkenanzeige meines Wissensstandes
Kooperation	Zusammenarbeit mit anderen Unternehmen
Kontakt	Kontaktformular für Anfragen
Links	Verlinkungen zu Partnern und Wissenswertem

WEB (Dropdown)

Showreel	Zeigt verschiedene Projekte
Workflow	Zeigt den Verlauf eines Web-Projektes
Kalkulator	Lässt einen unverbindlichen Preis kalkulieren

APPs (Dropdown)

Showreel	Zeigt unterschiedliche Projekte
Workflow	Zeigt den Verlauf eines App-Projektes
Preise	Zeigt auf, wo man sich über Preise informieren kann
Betatester	Anmeldeformular für Betatester

IT-SERVICE (Dropdown)

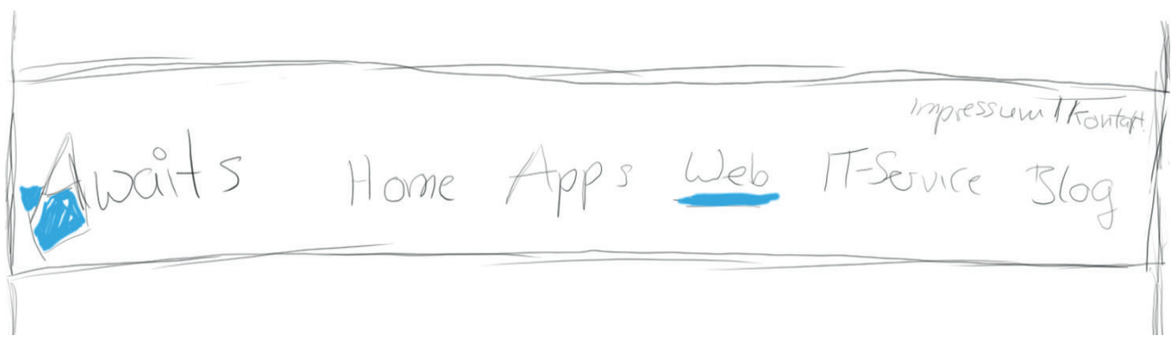
Information	Information über das IT-Service-Angebot
Leistungen	Leistungen die ich erbringen kann
Beratung	(Kauf-)Beratung bei Hard- und Software

BLOG	Artikel über Neuheiten der Web- und IT-Welt

13 Layout

Die Webseite soll über einen fixierten Header verfügen. Dieser soll die oben definierten Menüpunkte enthalten.

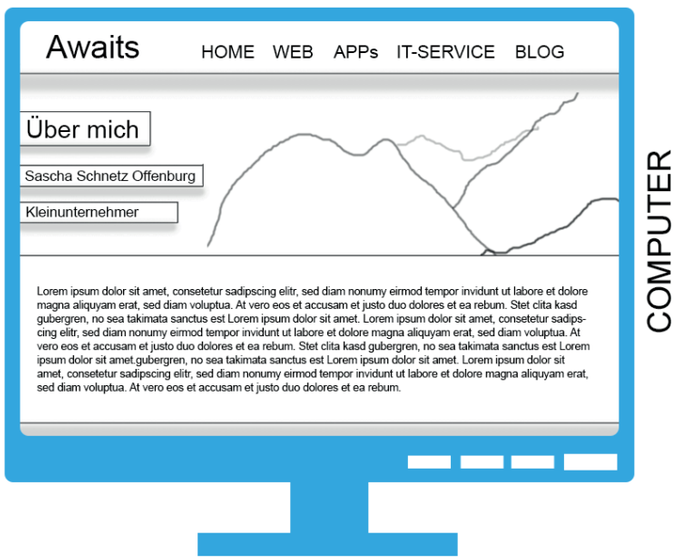
▼ Abbildung 9
Scribble des Headers der späteren Webseite



Wenn ein Abschnitt geöffnet ist, dann soll jeder Unterpunkt einen eigenen Kopf-Bereich besitzen, in dem die Überschrift des Unterpunktes (z.B. Über mich) steht. Außerdem sollen im Kopf-Bereich auch zwei kurze Sätze zum aktuellen Menüpunkt stehen. Der Inhalts-Bereich soll dann zweispaltig sein. Auf der einen Seite soll eine kurze Karrikatur zum Unterpunkt zu sehen sein, auf der anderen der Text. Dadurch wirkt dieser nicht so statisch und das visuelle Erscheinungsbild wird aufgewertet.

Der Inhalt (content) sollte scrollbar sein und sich beim scrollen unter den an der Oberkante der Seite fixierten Header schieben.

Abbildung 10 ▶
Entwurf der Webseite auf einem Computer



14 Wichtige Zusatz-Funktionen

Eine Webseite soll nicht nur Informationen bieten, sondern auch das Können des Programmierers aufzeigen sowie einen guten Eindruck vermitteln. Gerade visuell ansprechende Zusatzfunktionen hinterlassen oft einen guten Eindruck. Mögliche Zusatzfunktionen sind eine JQuery Galerie, der Preiskalkulator, die Filterfunktion der Blog-Artikel, ein responsiver Aufbau, optisch ansprechender Aufbau des Showreels und die erstklassigen Tooltips.

Nicht alle Funktionen werden ihren Weg in die fertige Version der Webseite schaffen. Diese werden aber eventuell zu einem späteren Zeitpunkt nachgerüstet.

VI Programmierung

„Würden Architekten Häuser bauen, wie Programmierer ihre Programme, dann könnte ein Specht die ganze Zivilisation zerstören.“ - Edward A. Murphy

In dieser Bachelor-Thesis ist die Erstellung und die Realisierung einer responsiven Single-Page-Applikation nach dem Model-View-Controller-Ansatz das Thema. Nachdem wir im vorherigen Kapitel den One-Pager konzipiert haben, wird er nun realisiert. Damit allerdings klar ist, um was es sich dabei genau handelt, sind die in der Themendefinition verwendeten Begrifflichkeiten im Folgenden nochmal genauer erklärt.

15 Einen Single-Page-Applikation erstellen

Ein Single-Page-Applikation bezeichnet einen Webauftritt, welche nur eine einzige Index-Seite hat, auf dem die komplette Webseite realisiert wird. Es wird hierbei nicht wie früher für jede Unterseite eine eigene HTML-Datei angelegt, sondern der Inhalt wird dynamisch über eine Programmlogik verwaltet.[25] Klickt der Nutzer zum Beispiel auf eine Unterseite „Über uns“, so wird nicht eine neue Seite „about.html“ geladen, sondern nur der Inhalt ausgetauscht. Das Grundgerüst bleibt unangetastet. Der große Vorteil hierbei ist, dass nicht bei jedem Klick das komplette Gerüst mit Bildern geladen werden muss, sondern wirklich nur der sich ändernde Inhalt ersetzt und neu geladen wird.

Es gibt auch andere Arten von Single-Page-Applikationen. So wird eine Webseite als One-Pager bezeichnet, bei der die komplette Webseite auf einer Seite aufgebaut ist und anstatt über Menüpunkte nur durch Scrollen fungiert. Wo man bei einem klassischen Menü über eine Navigation bewusst auf ein Untermenü zugreift, so scrollt man sich auf einem One-Pager nach unten. Dabei wird oft – wie bei einer Geschichte – eine Sinnvolle Anordnung und gute Übergänge zu den Unterpunkten gewährt. Das soll ein besseres Leseerlebnis gewährleisten und den Lesern eine Art roten Faden oder im besten Fall eine Geschichte mit Anfang und Ende geben[26]. Über Ankerpunkte und eine fixierte Menüleiste bleibt weiterhin ein schnelles Springen zu bestimmten Unterpunkten möglich.

25 http://devangelist.de/spa-webapi-knockout/ (Stand: 21.07.2014)

26 http://www.onepager.de (Stand: 15.06.2014)

Es gibt viele Mittel und Wege einen One-Pager oder Single-Page-Application zu erstellen. Man kann zum Beispiel AJAX benutzen, um mittels JavaScript Inhalte neu zu laden, oder auf ein Framework wie AngularJS zurückgreifen. Es ist dabei wichtig, dass man sich vorher Gedanken darüber macht, welcher Inhalt ausgetauscht werden muss und welcher konstant bleibt. Dynamischer Inhalt muss durch Platzhalter bzw. Variablen definiert werden.

Der Nutzen einer Single-Page-Applikation hat sich schnell herumgesprochen und an Beliebtheit gewonnen. Es gibt etliche Beispiele wie Facebook, Google, Spotify oder Twitter.[27] All diese Seiten verfügen über eine mächtige Programmierung und sind auf nur einer Seite realisiert. Dabei werden beide angesprochenen Arten verwendet. Das Unterbringen aller Inhalte auf einer Seite (Spotify) oder das dynamische Austauschen von Inhalt über z.B. Controller (Facebook).

16 Der Model-View-Controller

Bei der Programmierung wird der Model-View-Controller-Ansatz (MVC) verwendet. Hierbei handelt es sich um einen sehr populären Programmieransatz[28], der in den letzten Jahren sehr an Attraktivität

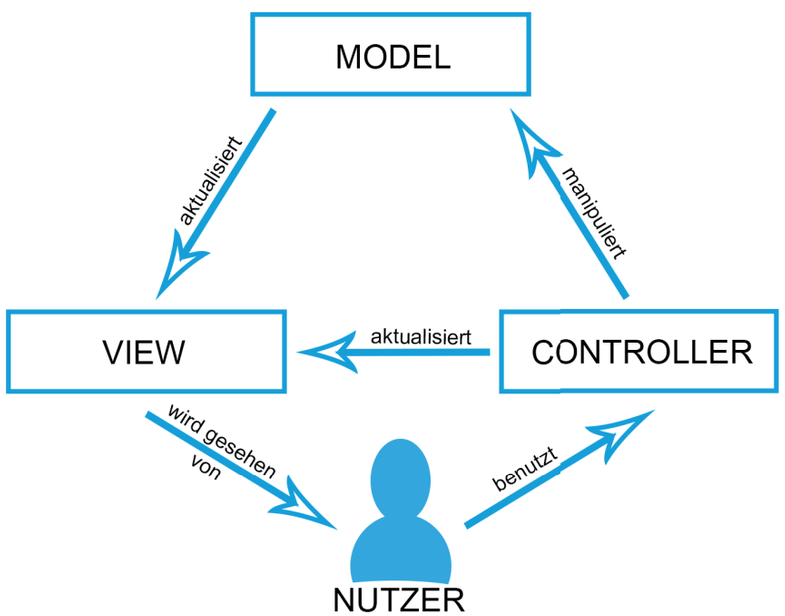


Abbildung 11 ▶
Eines von vielen möglichen Modellen des MVC-Ansatzes

27 Vgl. http://techcrunch.com/2012/11/30/why-enterprise-apps-are-moving-to-single-page-design/ (Stand: 21.07.2014)

28 MAURICE, Florence: PHP 5.4 & MySQL 5.5 - Der Einstieg in die Programmierung dynamischer Webseites. 1. Aufl. München : Addison-Wesley Verlag, 2012, S. 393

gewonnen hat. Der Model-View-Controller ist ein Entwurfsschema (oft Pattern genannt), welches auch häufig als Programmiergrundlage verwendet wird, da es sich gerade im Web sehr gut nutzen lässt.

Das Entwurfsmuster sieht eine Aufteilung in Datenmodell (Model), Präsentation(View) und Systemsteuerung (Controller) vor. Dabei ist das Model meistens für die Verarbeitung der Daten verantwortlich. Es kümmert sich darum, dass Daten und Benutzereingaben temporär oder permanent (zum Beispiel durch eine Datenbank) gespeichert werden. Die View ist für die Darstellung der Seite verantwortlich. Sie stellt die vom Model zur Verfügung gestellten Daten dar (zum Beispiel in einem Web-Browser). Der Controller als Steuereinheit überwacht und verarbeitet die Benutzerinteraktionen. Dadurch sind Model, View und Controller stark voneinander abhängig und bauen aufeinander auf. Die Komponenten sind dabei meistens in verschiedenen Dateien untergebracht. So lässt sich die Seite später besser warten und bleibt auch übersichtlicher. Soll eine Änderung am Datensatz vorgenommen werden, muss nur das Model bearbeitet werden, die anderen Komponenten bleiben unberührt. Viele Frameworks und Anwendungen bauen auf diesem Schema auf. Wenn Controller und Model sehr eng miteinander verknüpft sind, kann es vorkommen, dass sie in einer Datei untergebracht werden. Diese nennt sich Model Controller. Genauso kann das bei View und Controller der Fall sein. Dann spricht man von einem View Controller[29].

Der MVC ist für eine Realisierung von Single-Page-Applikationen ebenfalls sehr geeignet. Denn es gibt in dem Fall nur eine View (die „One-Page“) und alle Daten (Inhalte der Unterseiten) werden in Modellen untergebracht. Der (oder die) Controller managen dann die Funktionen und Interaktionen.

17 Responsive Design

Responsive (Web-)Design ist ein Begriff, der erst vor wenigen Jahren mit dem Boom der Smartphones und Tablets entstand. Das erste Mal wurde der Begriff Responsive Webdesign 2010 vom Designer Ethan Marcotte beschrieben[30]. Der Begriff bezeichnet die Anpassung einer Webapplikation an verschiedene Bildschirmgrößen. Dazu werden verschiedene Stylesheets definiert, welche dann je nach Auflösung über Media-Queries

29 STÄUBLE, Markus: Programmieren für iPhone und iPad. 4. Aufl. Heidelberg : dpunkt.verlag GmbH, 2012, S. 69

30 FRANKE, Florian; IPPEN, Johannes: Apps mit HTML5 und CSS3 für iPad, iPhone und Android. 2. Aufl. Bonn : Galileo Press, 2013, S. 411

geladen werden. Ein Media-Query ist ein Befehl, welcher ein Stylesheet nur lädt, wenn bestimmte Voraussetzungen erfüllt wurden.

Quelltext 13 ▶
Media-Queries im HTML-Dokument einbetten

```
1 <link href='css/klein.css' rel='Stylesheet'
2           media='screen and (max-width: 799px)' />
3 <link href='css/gross.css' rel='Stylesheet'
4           media='screen and (min-width: 800px)' />
```

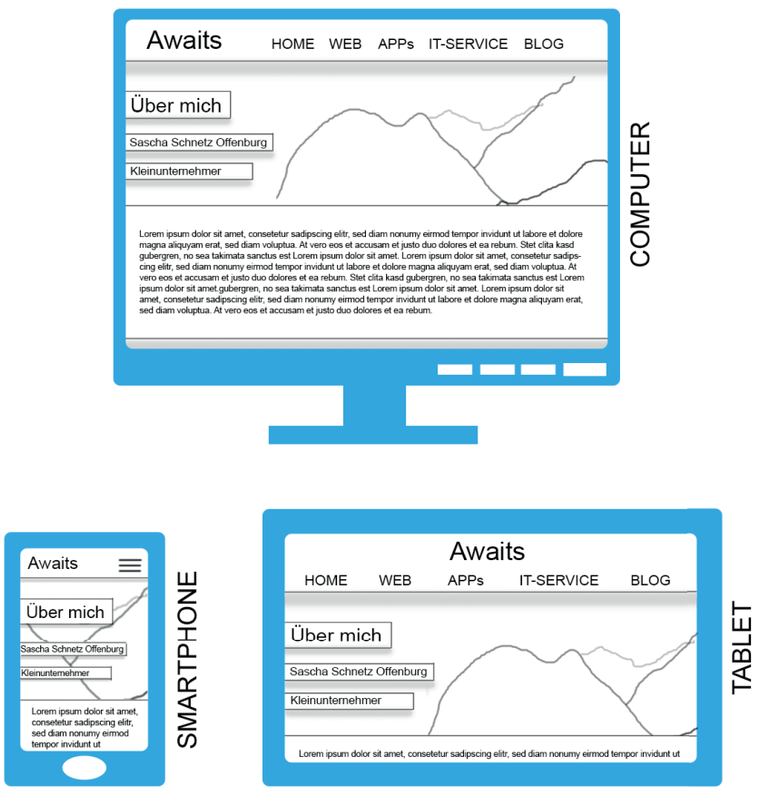
Dieser Befehl lädt bis zu einer Fensterbreite von 799 Pixel die CSS-Datei klein.css. Ist die Fensterbreite größer als 799 Pixel, wird die Datei gross.css geladen. Man kann auch nur ein einziges Stylesheet einbinden und die Media Queries in dem Stylesheet definieren. Das hat allerdings den Nachteil, dass das Stylesheet sehr groß wird und immer die komplette Datei geladen werden muss. Ein Media Query direkt im Stylesheet wird wie folg definiert:

Quelltext 14 ▶
Media-Queries direkt im Stylesheet definieren

```
1 @media screen and (max-width: 799px){...};
2 @media screen and (min-width: 800px){...};
```

Hier ist das gleiche Beispiel wie oben verwendet worden. Allerdings werden diese Befehle direkt im Stylesheet untergebracht. Trifft ein Media-Query zu, so wird der Inhalt in den geschweiften Klammern angewendet.

Abbildung 12 ▶
Hier sieht man den Entwurf unserer Homepage auf unterschiedlich großen Geräten



Somit lassen sich sehr viele Parameter abfragen um immer ein geeignetes Stylesheet zu laden. Man kann mit Media Queries auch die Orientierung (Portrait oder Landscape), die Displaybreite (anders als die Fensterbreite ist diese immer Geräteabhängig und kann nicht geändert werden) oder das Verhältnis (Pixel Ratio) erfragen und entsprechend reagieren.

Da es aber sehr aufwändig ist, für alle Endgeräte ein eigenes Stylesheet zu schreiben und diese auch noch sinnvoll einzubinden, gibt es Frameworks wie Bootstrap, die uns an dieser stelle Unterstützen.

18 Entwicklungsumgebung

Als Entwicklungsumgebung bieten sich heutzutage viele verschiedene Programme und Plattformen an. Als erstes wird eine geeignete Plattform in Gestalt eines Betriebssystems gesucht. Apple und Linux werden zwar immer beliebter, doch hier bietet sich meiner Meinung am besten die Programmierung mit Windows an. Gründe hierfür sind zum einen, dass das Adobe-Paket, mit welchem hier gearbeitet werden soll, mit Photoshop, Dreamweaver und Illustrator nicht oder nur bedingt auf Linux laufen. Windowssysteme sind darüber hinaus schnell auf jedem Heimrechner installiert. Wenn man keinen Apple-Rechner zur Verfügung hat, ist es hingegen fast unmöglich das Apple-Betriebssystem auf einem normalen Heimcomputer zu installieren, da Apple bestimmte Hardware voraussetzt. Da uns Windows von der Hochschule zur Verfügung gestellt wird, macht es die Verwendung nochmals attraktiver. Dennoch ist die Wahl eines Betriebssystems reine Geschmackssache, denn man kann Webseiten auch ohne das Adobe-Paket mit einfachen Texteditoren und Grafikprogrammen erstellen.

Nachdem ein passendes Betriebssystem gefunden wurde (hier Windows), muss man sich noch für passende Software entscheiden. Wie schon angesprochen sollen hier professionelle Softwarelösungen von Adobe © zum Einsatz kommen. Als HTML-Editor dient uns der Adobe Dreamweaver CC (natürlich geht auch jedes andere Textverarbeitungsprogramm wie zum Beispiel Notepad++). Zur Bearbeitung von Bilddateien nutzen wir Adobe Photoshop CC. Es gilt ebenfalls, dass man auf alternative Bildbearbeitungsprogramme zurückgreifen kann. Es gibt im Internet auch sehr gute und teilweise sogar kostenlose Alternativen wie GIMP.

Nachdem wir die zwei wichtigsten Programme gefunden haben brauchen wir noch einen lokalen Server. Normalerweise braucht man für eine

Homepage keinen lokalen Server. Man kann die HTML-Dateien direkt im Browser öffnen und ansehen. Da wir aber auf unserer Webseite einen eigenen Blog mit PHP-Programmieren, wird zwingend ein lokaler Server mit Datenbank benötigt. Ohne diesen würden wir keine Interpretation unseres PHP-Codes sehen, da PHP Serverseitig verarbeitet wird. Wenn wir keinen Server haben, kann unser Code auch nicht verarbeitet werden.

Als lokaler Server dient uns hier WAMP. WAMP ist eine Abkürzung für **W**indows **A**pache **M**ySql **P**HP und ist ein kleiner lokaler Server, welcher nach dem Download und wenigen Mausklicks installiert ist. Nach der Installation können wir Dateien in das Verzeichnis `C:/WAMP/www/` im Installationsordner ablegen und im Browser über die Adresse `localhost` abrufen. Auf der geöffneten Übersichtsseite können wir unsere Seite anwählen oder auch zu MyPHPAdmin wechseln, welcher später noch für die Administration der benötigten Datenbanken wichtig ist.

19 Die Verzeichnisstruktur

Nachdem wir unseren Server installiert haben müssen wir uns für eine Verzeichnisstruktur entscheiden, das unser Webprojekt übersichtlich hält. Zuallererst brauchen wir eine Index-Datei. Diese enthält unser Grundgerüst. Wir legen wegen des späteren Blogs die Datei gleich als .php und nicht als .html an. Nun brauchen wir mehrere Ordner. Der bekannteste und am häufigsten verwendete unterordner ist der Bilder-Ordner. Wir legen also einen Ordner mit dem Namen „images“ an. Hier werden alle Grafiken (ggfs. auch in weitere Unterordner) abgelegt. Neben dem Bilder-Ordner brauchen wir zusätzlich auch ein Verzeichnis für unsere ganzen Skripte und Stylesheets. Deshalb erstellen wir einen Ordner mit dem Namen `lib`. Die Abkürzung steht für Library (Sammlung). Hier kommen weitere 3 Unterordner rein. Und zwar `CSS` für Stylesheets, `JS` für JavaScript-Dateien und `fonts` für Schriftarten, die wir nachladen können. Außerdem wird noch ein „init“-Verzeichnis benötigt, in welches Dateien zur Initialisierung der Webseite hineinkommen, wie z.B. die PHP-Dateien, Controller und Modelle. Auch die Datei mit den Login-Daten für die Datenbank befindet sich an dieser Stelle. Das Einzige, das uns jetzt noch fehlt, ist ein Verzeichnis für Templates. In das „templates“-Verzeichnis werden die einzelnen Seiten wie Home, APPs, WEB, usw. abgelegt. Im Prinzip befindet sich hier der Inhalt, welcher in das Grundgerüst reingeladen werden soll.

20 Das Grundgerüst

Eine Single-Page-Applikation ist definiert durch die Tatsache, dass es nur eine einzige Index-Datei gibt und jeglicher Inhalt dynamisch geladen wird. Hierfür wird die bereits angelegte leere index.php geöffnet. Wie jede Webseite müssen wir hier als erstes das klassische HTML-Gerüst, bestehend aus dem Doctype, den HTML, HEAD und BODY Tags, erstellen. Auch den Titel der Seite und die META-Angaben im Header (wie das Charset oder den Viewport für Smartphone-Skalierung) können wir schon einfügen. Im Bodybereich gibt es einen HTML5 HEADER-Container. Dieser beherbergt nachher unsere Navigation. Außerdem brauchen wir hier einen Content-Bereich, der unsere Inhalte laden soll. Hierzu wird ein DIV-Container erstellt, welcher die Id „content“ enthält. Außerdem beinhaltet er einen Weiteren DIV-Container, welcher AngularJS erlaubt eine View zu laden.

◀ Quelltext 15
Das Grundgerüst unserer Webseite. Auf CD: /homepage/grundgeruest.php

```
1  <!DOCTYPE HTML> <!-- HTML5 Doctype -->
2  <HTML>
3    <HEAD>
4          <meta charset='utf-8' />
5          <meta name = 'viewport'  content = 'initial-
6          scale = 1.0, user-scalable = no'>
7          <title>Awaits</title>
8    </HEAD>
9
10   <BODY>
11         <header>
12               <!-- Hier kommt die Navigation rein -->
13         </header>
14
15               <div id='content'>
16               <div>
17                     <!-- Hier wird der Inhalt geladen -->
18               </div>
19          </div>
20 </BODY>
21 </HTML>
```

21 Stylesheets, Frameworks und Skripte einbetten

Nachdem das Grundgerüst steht müssen wir noch unsere benötigten Plugins, Frameworks und Stylesheets laden. Zunächst fügen wir also

Verlinkungen zu CSS-Dateien in den Kopf-Bereich unserer index.php (nach Zeile 7 im Grundgerüst) ein.

Quelltext 16 ▶
Einbinden der verschiedenen Stylesheets
Auf CD: /homepage/index.php

```
1 <link href="lib/css/reset.css" rel="stylesheet" />
2 <link href="lib/css/bootstrap.css" rel="stylesheet" />
3 <link href="lib/css/style.css" rel="stylesheet" />
```

An erster Stelle wird das Stylesheet reset.css geladen, welches Browserspezifische Vordefinitionen überschreibt und so eine einheitliche Ausgangslage schafft. Danach wird das Stylesheet von Bootstrap geladen. Hier wird noch das große unkomprimierte Stylesheet geladen, damit wir ggfs. noch Änderungen vornehmen können. Ist das Projekt fertig wird diese Datei komprimiert. Das bedeutet, dass alle Leerzeichen und Umbrüche entfernt werden, die nur unnötige Kilobyte verbrauchen. Als drittes wird die selbst erstellte style.css geladen. Hierbei handelt es sich um ein leeres Stylesheet, in welchem wir eigene Anpassungen vornehmen können.

Cascading-Stylesheets (CSS) sind, wie der Name schon sagt, kaskadierend. Das bedeutet, dass sie der Reihenfolge nach abgearbeitet werden und vorherige Definitionen überschrieben werden. Aus diesem Grund ist auch unser eigenes Stylesheet das Letzte. Somit können wir sichtbare Änderungen vornehmen und diese werden nicht von anderen wieder negiert.

Nachdem alle Stylesheets eingebunden sind, müssen noch die Skripte und Frameworks eingebunden werden. Das geschieht am Ende der Seite, über den Body-Tag. Da es sich hierbei um JavaScript-Bibliotheken handelt, werden alle Skripte am Ende über Skript-Tags geladen:

Quelltext 17 ▶
Skripte in unser HTML-Dokument einbinden

```
1 <script src='lib/js/jquery-2.1.1.min.js'></script>
2 <script src='lib/js/bootstrap.js'></script>
3 <script src='lib/js/angular.min.js'></script>
4 <script src='lib/js/angular-route.js'></script>
5 <script src='init/module.js'></script>
6 <script src='init/controller.js'></script>
7 <script src='init/routing.js'></script>
8 <script src='lib/js/jquery.sliphover.min.js'></script>
```

Neben den Frameworks werden hier auch die wichtigen Dateien controller.js, module.js, und routing.js (mit dem benötigten Angular.js-Plugin angular-route.js) für unseren Model-View-Controller An-

satz eingebunden. Wie bei den Stylesheets wird auch hier am Ende noch ein eigenes Script und ein kleines JQuery Plugin eingebunden.

22 Die Navigation

Nachdem das Grundgerüst steht und alle Komponenten eingebettet sind, kümmern wir uns um die Navigation. Zuerst wird die Navigation erstellt, dann mit einem Stylesheet angepasst und zu guter Letzt noch die Funktionen programmiert.

Bootstrap stellt uns in seiner Online-API eine vorprogrammierte sehr gute und relativ passende Navigation (Navbar) zur Verfügung, die wir zuerst einbetten und dann modifizieren:

```
1  <nav class='navbar navbar-default' role='navigation'>
2    <div class='container-fluid'>
3      <!-- Brand and toggle get grouped for better mobile
4      display -->
5      <div class='navbar-header'>
6        <button type='button' class='navbar-toggle'
7        data-toggle='collapse' data-target='#bs-example-
8        navbar-collapse-1'>
9          <span class='sr-only'>Toggle navigation</span>
10         <span class='icon-bar'></span>
11         <span class='icon-bar'></span>
12         <span class='icon-bar'></span>
13       </button>
14       <a class='navbar-brand' href='#'>Brand</a>
15     </div>
16
17     <!-- Collect the nav links, forms, and other
18     content for toggling -->
19     <div class='collapse navbar-collapse' id='bs-
20     example-navbar-collapse-1'>
21       <ul class='nav navbar-nav'>
22         <li class='active'><a href='#'>Link</a></li>
23         <li><a href='#'>Link</a></li>
24         <li class='dropdown'>
25           <a href='#' class='dropdown-toggle' data-
26           toggle='dropdown'>Dropdown <b
27           class='caret'></b></a>
28           <ul class='dropdown-menu'>
```

◀ Quelltext 18
Die unveränderte Bootstrap-Navigation
Quellcode von http://getbootstrap.com/components/ (Stand: 05.07.2014)

```
29              <li><a href='#'>Action</a></li>
30              <li><a href='#'>Another action</a></li>
31              <li><a href='#'>Something else here</a></li>
32              <li class='divider'></li>
33              <li><a href='#'>Separated link</a></li>
34              <li class='divider'></li>
35              <li><a href='#'>One more separated link</a>
36              </li>
37            </ul>
38          </li>
39        </ul>
40      </div><!-- /.navbar-collapse -->
41    </div><!-- /.container-fluid -->
42  </nav>
```

Im oberen Bereich findet sich ein Navigations-Header (Zeile 5-15), welcher aus einem Button besteht und nur angezeigt wird, wenn die Bildschirmgröße auf die eines Smartphones schrumpft. Auch befindet sich das „Branding“ im Header, also in unserem Fall das Unternehmenslogo, welches auf allen Bildschirmgrößen angezeigt wird.

Im Hauptteil ist die eigentliche Navigation untergebracht, welche aus einfachen Links und auch einem Dropdown-Menü besteht. In diesem Fall stimmen natürlich weder Verlinkungen noch die Bezeichnungen. Unsere spätere Navigation soll so aufgebaut sein, dass Sie über 4 Dropdown-Menüs und einen „normalen“ Menüpunkt verfügt. Die Navigation soll folgende Navigationspunkte enthalten:

HOME
- Über Mich
- Kenntnisse
- Kooperation
- Kontakt
- Links

WEB
- Showreel
- Workflow
- Kalkulator

APPs
- Showreel
- Workflow

Preisinformationen
Betatester

IT-SERVICE
Informationen
Leistungen
Beratung

BLOG

Dabei sollen alle Unterpunkte einer Kategorie (z.B. Kenntnisse, Zusammenarbeit, ...) auf einer Seite untereinander angezeigt werden und die Verlinkungen nur die aktuelle Ansicht an die richtige Stelle scrollen (One-Pager). Erst ein anderer Abschnitt soll den Inhalt der Seite neu laden. Das ist ein immer häufiger gesehener Ansatz von Webseiten (siehe Spotify).

Schauen wir uns erstmal die fertige Navigation in ihrer Gesamtheit an und besprechen die Änderungen danach:

◀ Quelltext 19
Die modifizierte und angepasste Navigation
Auf CD: /homepage/index.php

```
1  <nav class='navbar navbar-default' role='navigation'>
2                <div class='container-fluid'>
3
4                  <!-- Header der Navigation -->
5                  <div class='navbar-header'>
6                    <button type='button' class='navbar-
7                    toggle' data-toggle='collapse' data-
8                    target='.navbar-collapse'>
9                      <span class='sr-only'>Toggle
10                     navigation</span>
11                     <span class='icon-bar'></span>
12                     <span class='icon-bar'></span>
13                     <span class='icon-bar'></span>
14                   </button>
15                   <a class='navbar-brand'
16                   href='#about'><img id='logo'
17                   src='images/logo.png' /></a>
18                 </div>
19
20
21                 <!-- Body der Navigation -->
22                 <div class='navbar-collapse collapse'
23                 style='height: 0px;'>
```

```
<ul class='nav navbar-nav'>
  <li class='dropdown'
  id='homenav'>
    <a href='#' class='dropdown-
    toggle' data-toggle='dropdown'>
    HOME <b class='glyphicon glyphicon-
    chevron-down'></b></a>
    <ul class='dropdown-menu'>
        <li><a href='#{{homeScroll}}
        about' onClick='updateNavi(
        „#homenav")'>Über mich</a></li>
      <li><a href='#{{homeScroll}}
      Kenntnisse' onClick='updateNavi(
      „#homenav")'>Kenntnisse</a>
      </li>
      <li><a href='#{{homeScroll}}
      zusammenarbeit'
      onClick='updateNavi(„#homenav")
      '>Zusammenarbeit</a></li>
      <li><a href='#{{homeScroll}}
      kontakt' onClick='updateNavi(
      „#homenav")'>Kontakt</a></li>
      <li><a href='#{{homeScroll}}
      links' onClick='updateNavi(
      „#homenav")'>Links</a></li>
    </ul>
  </li>
  <li class='dropdown'
  id='appnavi'>
    <a href='#' class='dropdown-
    toggle' data-toggle='dropdown'>
    APPs <b class='glyphicon glyphicon-
    chevron-down'></b></a>
    <ul class='dropdown-menu'>
        <li><a href='#{{appScroll}}
      appShowreel' onClick='updateNavi(
       „#appnavi")'>Showreel</a></li>
      <li><a href='#{{appScroll}}
      appWorkflow' onClick='updateNavi(
      „#appnavi")'>Workflow</a></li>
      <li><a href='#{{appScroll}}
      appPreise' onClick='updateNavi(
      „#appnavi")'>
```

```
67                      Preisinformationen</a></li>
68                      <li><a href='#{{appScroll}}
69                      appBeta' onClick='updateNavi(
70                      „#appnavi“)'>Betatester werden
71                      </a></li>
72                    </ul>
73                  </li>
74                  <li class='dropdown'
75                  id='webnavi'>
76                    <a href='#' class='dropdown-
77                    toggle' data-toggle='dropdown'>
78                    WEB <b class='glyphicon glyphicon-
79                    chevron-down'></b></a>
80                    <ul class='dropdown-menu'>
81                        <li><a href='#{{webScroll}}
82                      webShowreel' onClick='updateNavi(
83                        „#webnavi“)'>Showreel</a></li>
84                      <li><a href='#{{webScroll}}
85                      webWorkflow' onClick='updateNavi(
86                      „#webnavi“)'>Workflow</a></li>
87                      <li><a href='#{{webScroll}}
88                      webPreise' onClick='updateNavi(
89                      „#webnavi“)'>Preiskalkulator
90                      </a></li>
91                    </ul>
92                  </li>
93                  <li class='dropdown'
94                  id='itsnavi'>
95                    <a href='#' class='dropdown-
96                    toggle' data-toggle='dropdown'>
97                    IT-SERVICE <b class='glyphicon
98                    glyphicon-chevron-down'></b></a>
99                    <ul class='dropdown-menu'>
100                    <li><a href='#{{itserviceScroll}}
101                       itsInfo' onClick='updateNavi(
102                       „#itsnavi“)'>Informationen</a>
103                       </li>
104                     <li><a href='#{{itserviceScroll}}
105                      itsLeistungen'
106                      onClick='updateNavi(„#itsnavi“)
107                      '>Leistungen</a></li>
108                     <li><a href='#{{itserviceScroll}}
109                     itsBeratung' onClick='updateNavi(
```

```
110                     „#itsnavi“)‘>Beratung</a></li>
111                   </ul>
112                 </li>
113                 <li class=‘dropdown blog-menu‘
114                 id=‘blognavi‘><a href=‘#blog‘
115                 onClick=‘updateNavi(„#blognavi“)
116                 ‘>BLOG</a></li>
117                 <li> </li>
118               </ul>
119             </div><!-- /.navbar-collapse -->
120           </div><!-- /.container-fluid -->
121         </nav>
```

Auf den ersten Blick wirkt diese Navigation der Vorlage sehr ähnlich – das ist auch kein Wunder. Immerhin wurden nur kleine Elemente geändert, aber die Struktur beibehalten.

In dem Head-Bereich der Navigation findet sich nur eine Änderung und zwar die Verlinkung unseres Firmenlogos an der Stelle des Branding (Zeilen 15-17). Dadurch ist das Logo immer in der Navigation sichtbar und verlinkt auf die Startseite.

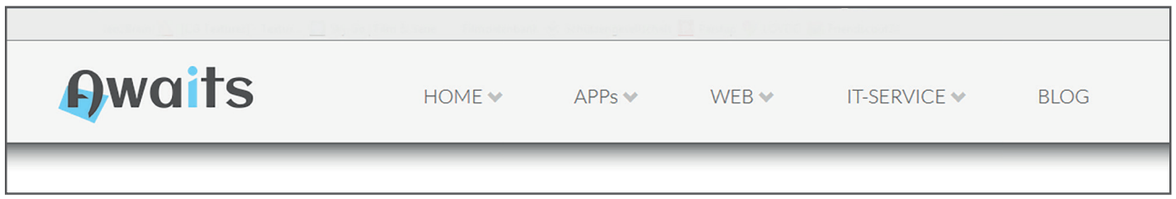


Abbildung 13 ▲
So sieht die fertige Navigation nach ein paar CSS anpassungen aus

Interessant wird es aber im nächsten Abschnitt. Jeder Menüpunkt ist in einer Ungeordneten Liste (`<ul>`) untergebracht und selbst als Listenelement definiert (`<li>`). Wenn es nun ein Dropdownmenü ist, so wird hier eine weitere ungeordnete Liste verschachtelt. Bootstrap verarbeitet diese Verschachtelungen dann richtig und macht uns daraus ein Dropdown-Menü.

Was vielleicht auch gleich auffällt sind die merkwürdigen AngularJS Platzhalter wie z.B. `{{homeScroll}}`. In jedem Dropdown-Punkt findet sich das wieder. Das Problem an der Navigation war, dass wenn man zu einem Unterpunkt springen wollte, die Navigation immer nur an den obersten Punkt springen konnte. Das passierte, weil die ganze Seite noch nicht geladen worden war und so nur das Scrollen innerhalb eines Menüpunktes funktionierte. Abhilfe konnte AngularJS mit einem Trick schaffen. Wenn wir innerhalb eines Menüs navigieren, so ist der Platz-

halter für das eigene Menü einfach leer und in der Verlinkung steht nur `<a href="#Kenntnisse" …>`. Wenn wir uns aber im Abschnitt WEB befinden und auf Kenntnisse im Home-Baum zugreifen wollen, ginge das so nicht. Darum lädt der AngularJS Controller von WEB zwischen das Hashtag(#) und „Kenntnisse" noch einen kleinen Verweis `Menüpunkt?scrollTo=`. Menüpunkt ist hier stellvertretend für den jeweiligen Bereich. In unserem Beispiel stünde hier also, wenn wir auf WEB sind und auf Home -> Kenntnisse zugreifen wollen, `<a href="#home?scrollTo=Kenntnisse" …>`. Doch was passiert nun? Es wird hier eine AngularJS-Funktion aufgerufen. Diese müssen wir in unserer Modul-Datei noch konfigurieren und AngularJS mitteilen, wie es sich verhalten soll, wenn in einer Verlinkung der String scrollTo gefunden wird. Der Vorteil: Die scrollTo-Funktion von AngularJS wird erst geladen, wenn der neue Inhalt geladen wurde und so kann an die richtige Stelle gesprungen werden.

```
1  var awaitsApp = angular.module('awaitsApp', [
2  'ngRoute'])
3    .run(function($rootScope, $location, $routeParams) {
4      $rootScope.$on('$routeChangeSuccess', function(
5      newRoute, oldRoute) {
6        $location.hash($routeParams.scrollTo);
7  window.scrollTo(window.pageXOffset, window.pageYOffset -
8           100);
9      })
10 });
```

◀ Quelltext 20
Im Modul wird die Applikation initialisiert
Auf CD: /homepage/init/module.js

Der Funktion werden wichtige Parameter (wie `$location`) übergeben, welche von AngularJS zur Verfügung gestellt werden. Danach wird, wie bei einer If-Bedingung, geprüft, ob sich der `$rootScope` (also im Prinzip der Inhalt) geändert hat. Wenn das der Fall ist, soll geschaut werden, ob sich in der `$location` (dem Link) ein Parameter `scrollTo` befindet. Hier sieht man, dass das erst passiert, wenn sich die `$location` auch wirklich geändert hat und der neue Inhalt dadurch zur Verfügung steht. Die AngularJS-Interne Funktion `scrollTo()` wird hier also aufgerufen und an den im Link übergebenen Pfad gesprungen. Die Codezeile `window.scrollTo` korrigiert danach noch die Position. In unserem Fall wird immer ein bisschen zu tief gesprungen, da der Header nicht in die Position des Ankers miteinberechnet wurde und wir daher noch 100 Pixel Versatz einberechnen müssen.

Unsere Navigation funktioniert einwandfrei und springt immer an die richtigen Stellen. Doch was ist mit der onClick-Funktion, die in jedem

Link auftaucht? Diese Funktion ruft eine JavaScript-Funktion auf, welche wir selbst schreiben werden und die nach einem Klick auf einem Smartphone das Menü als `aktiv` kennzeichnet.

Quelltext 21 ▶
Die updateNavi()-Funktion, welche den aktuellen Navigationspunkt hervorhebt
Auf CD: /homepage/lib/js/script.js

```
function updateNavi(thisID){
   if (!$(thisID).hasClass('active')){
         $('.nav').find('li.active').removeClass(
         'active');  //Finde und entferne altes „aktiv"
         $(thisID).addClass('active');
   }
}
```

Wir prüfen, ob der Menüpunkt bereits als `aktiv` gekennzeichnet wurde. Bewegen wir uns innerhalb eines Bereichs oder greifen wir auf einen anderen zu? Wenn der zweite Fall eintritt, wird zuerst nach einem aktiven Menüpunkt gesucht und dieser wieder auf inaktiv gesetzt. Das geschieht durch `removeClass`. Danach wird der aktuelle Link ausgewählt (`thisID`) und diesem die Klasse `active` hinzugefügt (`addClass`). So markieren wir immer das aktive Menü.

Doch es gibt noch ein kleines Problem. Bei Smartphones fährt die Navigation nicht wieder von alleine ein sondern bleibt ausgeklappt. Wo bei einem Desktoprechner genug Platz für eine Navigationsleiste ist, wird bei einem Smartphone eine Navigation zum aufklappen dargestellt. Man müsste diese aber über einen Klick auf das Navi-Symbol wieder einklappen lassen, da sie es nicht von alleine macht. Da das nicht sehr Benutzerfreundlich ist wird hier ein Funktionsaufruf definiert, welcher immer aufgerufen wird, wenn wir einen Menüpunkt anklicken. Diese Funktion ist sehr übersichtlich geschrieben:

Quelltext 22 ▶
Bedienung der Navigation für Smartphones optimieren
Auf CD: /homepage/lib/js/script.js

```
if(width<767){
   $('.dropdown-menu a').on('click', function(){
         $('.navbar-toggle').click();
   });
   $('.blog-menu a').on('click', function(){
         $('.navbar-toggle').click();
   });
}
```

Wieder behelfen wir uns mit einem kleinen Trick. Jedes Mal wenn auf einem Smartphone (das stellen wir über die if-Bedingung am Anfang sicher) ein Link angeklickt wird, so wird ebenfalls ein Klick auf `navbar-toggle` simuliert. Und das ist genau der Button, der die Naviga-

tion auf Smartphones ein- bzw. ausklappt. Dadurch wird das Menü auf Smartphones immer ordentlich eingeklappt, sollte man auf einen Link klicken.

Unser Menü ist somit bereit und kann gestaltet werden. Dazu werden ein paar Stile definiert. An und für sich ist unser Menü schon recht ansehnlich, aber dennoch gibt es ein paar Änderungen welche vorgenommen werden müssen. Zum Beispiel, dass die Navigation fixiert am oberen Bildschirmrand positioniert ist.

Das Problem beim Gestalten ist, dass wir die Navigation mehrmals anpassen müssen. Für Desktop-Rechner, für kleinere Bildschirme (wie zum Beispiel Tablets) und für Smartphones. Das machen wir innerhalb eines Stylesheets über Media-Querries. Diese werden innerhalb einer Datei wie schon im Abschnitt Responsive Design z.B. über `@media (min-width: 1120px) {…}` definiert.

Zu anfangs müssen wir die Schrift in Größe, Art und Farbe neu definieren. Auch das Logo muss in Größe und Position jedes Mal definiert werden. Die Menüpunkte brauchen auch ein neues passendes Design für den Fall, dass der Menüpunkt als aktiv markiert ist.

Wie die genauen CSS-Definitionen aussehen lässt sich im Anhang auf CD in der style.css Datei nachlesen.

23 AngularJS und Routing

Kommen wir zu einem anderen Framework, welches von Google entwickelt wurde und im Prinzip eine clientseitige Dynamik hinzufügt. Was AngularJS genau ist und wie es funktioniert kann im Bereich Frameworks (ab Seite 13) nachgelesen werden.

Die nötigen JavaScript-Dateien haben wir schon eingebunden. Jetzt müssen wir der Seite aber noch sagen, dass es sich um eine AngularJS-Anwendung handelt. Das geschieht einfach und schnell über das Attribut ng-app im HTML-Tag: `<html ng-app="awaitsApp">`. Der App-Name „awaitsApp“ ist dabei ein beliebig für die Web-Applikation gewählter Name. Dieser wird später wieder benötigt.

Wir haben unserem Grundgerüst gesagt, dass es eine AngularJS-Applikation ist. Was noch fehlt ist der Bereich, in dem wir später dynamischen Inhalt nachladen möchten. Da das auch über AngularJS geschieht, ge-

ben wir dem im Grundgerüst schon angelegten DIV-Knoten noch das Attribut `ng-view`. Eine `ng-view` ist die von AngularJS bereitgestellte Möglichkeit eine View (Anzeige) dynamisch zu verändern. Das sieht dann wie folgt aus:

Quelltext 23 ▶
Die ng-view definieren, welche den Inhalt lädt
Auf CD: /homepage/index.php

```
1 <div id='content'>
2   <div ng-view>
3     <!-- Hier wird nun der Inhalt der Seiten geladen -->
4   </div>
5 </div>
```

In vorherigen Schritten haben wir Dateien für den Controller, die Module und das Routing verknüpft. Diese müssen wir jetzt noch mit Inhalt füllen. In der Datei module.js befinden sich schon ein paar Codezeilen und zwar die von der Navigation. Dort haben wir schon den ersten Schritt gemacht. Wir haben die Applikation initialisiert und die Navigation bearbeitet. Damit ist schon das wichtigste in der Modul-Datei gemacht worden, denn diese dient nur dazu, die nötigen Dienste zu initialisieren. In unserem Fall sind das momentan zwei. Zum einen die schon geschehene Initialisierung unserer Applikation. Die Zeile `var awaitsApp = angular.module(‚awaitsApp', [‚ngRoute'])` ist nötig, damit AngularJS funktioniert. Dabei wird einfach eine Variable angelegt, welche die Applikation initialisiert. Dazu wird der Appname übergeben (den wir im HTML-Tag definiert haben) und dann noch weitere benötigte Parameter. In unserem Fall noch das `ngRoute,` auf das wir später noch zu sprechen kommen. Danach wird in dieser Datei noch der Scroll-Fehler behoben, der im Vorangegangenen ausführlich behandelt wurde.

Der komplette Inhalt dieser Datei besteht momentan also aus 7 Zeilen Code Die Controller-Datei, zu welcher wir nun kommen, ist da schon weitaus umfangreicher. Gehen wir zuerst nochmal auf den Zweck dieser Datei ein: Ein Controller ist dafür Zuständig, die Daten zu initialisieren und die Interaktion mit dem Unser abzuwickeln. Er kümmert sich sozusagen um In- und Output unserer Seite. Damit das funktioniert, können wir beliebig viele Controller anlegen und nur die Benötigten laden. Man kann auch alles in einen einzigen Controller schreiben, allerdings wird das sehr schnell unübersichtlich. Kommen wir zum Haupt-Controller:

Quelltext 24 ▶
Main-Controller
Auf CD: /homepage/init/controller.js

```
1 awaitsApp.controller('mainController', function($scope,
2 $location) {
3 ...
4 }
```

In den geschweiften Klammern wird der Inhalt definiert. Doch wie funktioniert ein Controller? Als erstes wird unsere App (`awaitsApp`) angesprochen. Danach sagen wir, dass es sich um einen Controller handelt (Zeile 1). In die Klammern kommt ein einmaliger Name zur Identifikation. Der folgende Inhalt wird in eine Funktion geschrieben. Dabei ist darauf zu achten, dass immer der Gültigkeitsbereich (`$scope`) übergeben wird. Über diesen Parameter können wir auf Attribute oder Variablen unserer Seite zugreifen. Der `mainController` hat noch keinen Zweck, er ist einfach ein Controller, welcher immer aktiv ist und globale Variablen speichern und vararbeiten kann. Aber kommen wir jetzt zu den spezifischen Controllern, welche nur auf bestimmten Seiten geladen werden sollen.

```
1 awaitsApp.controller('homeController', function($scope,
2 $rootScope) {
3   $rootScope.homeScroll = '';
4   $rootScope.appScroll = 'apps?scrollTo=';
5   $rootScope.webScroll = 'web?scrollTo=';
6   $rootScope.itserviceScroll = 'itservice?scrollTo=';
7 });
```

◀ Quelltext 25
Home-Controller
Auf CD: /homepage/init/controller.js

Genau wie der `mainController` wird hier der `homeController` initialisiert. Was wir hier allerdings noch übergeben ist der `rootScope`. Dieser ermöglicht uns über den Gültigkeitsbereich hinaus auf übergreifende, globale Variablen zuzugreifen. Das ist nötig, da unser Controller nur in der `ng-view` geladen wird und nur Inhalte der View bearbeiten kann. Die Navigation ist dabei außerhalb.

Navigation? Ja, hier werden die vorhin im Bereich Navigation angesprochenen Platzhalter gefüllt. Jeder Controller wird später andere Werte in die Platzhalter schreiben. Wir befinden uns im `homeController`, daher wird bei home keine `scrollTo-Methode` ausgeführt.

Der Homecontroller ist hierbei noch sehr übersichtlich. Man könnte nun weitere Variablen anlegen und die View entsprechend laden. Allerdings ist die Home-Seite relativ statisch. Interessant wird es bei den nächsten Seiten. Daher kommen wir nun zum `appsController`:

```
1 awaitsApp.controller('appsController', function($scope,
2 $rootScope) {
3   $rootScope.homeScroll = 'home?scrollTo=';
4   $rootScope.appScroll = '';
5   $rootScope.webScroll = 'web?scrollTo=';
```

◀ Quelltext 26
APPs-Controller
Auf CD: /homepage/init/controller.js

```
6    $rootScope.itserviceScroll = 'itservice?scrollTo=';
7
8    $scope.appProjekte = [{
9          'titel': 'CrazyBalloon',
10         'status': 'Veröffentlicht',
11         'datum': 'März 2014',
12         'id': '1',
13         'link': 'https://play.google.
14   com/store/apps/details?id=air.eu.awaits.crazyballoon'
15   }];
16 });
```

Dieser Controller lädt das Showreel unserer Seite. Wir legen ein Array `appProjekte` an, welches wir mit Inhalt füllen. Wir müssen nichts weiter tun um dieses Array zu erstellen. Dabei werden die Attribute Titel, Status, Datum, ID und den Link zum Download verwendet. Durch `$scope` steht uns dieses Array in der HTML-Datei zur Verfügung und wir können später in der View darauf zugreifen. Dazu aber im nächsten Kapitel mehr.

Die folgenden Controller funktionieren genau so, allerdings sind sie sehr viel umfangreicher als die Ersten. Aus diesem Grund werden bei den nächsten Controllern nur Ausschnitte besprochen. Die kompletten Quelltexte finden Sie wie gewohnt im Anhang auf CD.

Unser nächster Controller ist der `webController`. Dieser hat neben dem Showreel ein weiteres Array und weitere Funktionen, die uns auf der Webseite die Realisierung eines Preiskalkulators ermöglichen. Dazu stellt uns der Controller ein Array mit Preisen wie beim Showreel zur Verfügung, die wir auf der Seite anzeigen lassen. Der Clou: wir können in dem Controller direkt auf Platzhalter der HTML-Datei (wir erinnern uns zum Beispiel an `{{homeScroll}}`) zugreifen und diese definieren. Nun können wir aber auch eine Funktion in dem Controller definieren, welche ausgeführt werden soll, wenn der Nutzer einen Button anklickt. Die Anzeige wird dadurch dynamisch und es geschieht alles innerhalb einer View. Stellen wir uns also ein Formular vor, welches etliche Elemente einer möglichen Webseite mit Preisen anzeigt. Jeder Klick auf ein Element verändert den aktuellen Preis. So können sich Kunden einen ungefähren Preis für ihre gewünschte Webseite kalkulieren lassen.

```
1  var $sprachen = 1;
2  var $tempSprachenPreis = 0;
3  $scope.anzSprachen = $sprachen+' Sprache (inklusive)';
4
5  $scope.sprachenFkt = function(richtung){
6    if(richtung=='plus'){
7          $sprachen++;
8          $tempSprachenPreis += 60;
9          $preisTemp += 60;
10         $scope.anzSprachen = $sprachen+' Sprachen ('+
11         $tempSprachenPreis+' Euro)';
12         $scope.endpreis = $preisTemp+' Euro';
13   }else{
14         if($sprachen > 2){
15               $sprachen--;
16               $tempSprachenPreis -= 60;
17               $preisTemp -= 60;
18         $scope.anzSprachen = $sprachen+' Sprachen ('+
19               $tempSprachenPreis+' Euro)';
20               $scope.endpreis = $preisTemp+' Euro';
21         }else if($sprachen == 2){
22               $sprachen--;
23               $tempSprachenPreis -= 60;
24               $preisTemp -= 60;
25         $scope.anzSprachen = $sprachen+'Sprache (inkl)';
26               $scope.endpreis = $preisTemp+' Euro';
27         }
28   }
29 }
```

◀ Quelltext 27
Programmlogik des Preiskalkulators im Web-Controller
Auf CD: /homepage/init/controller.js

Hier sieht man eine Funktion, welche dem Preiskalkulator das Hinzufügen von Sprachen ermöglicht. Also Sprachen, in denen die Webseite später erscheinen soll. Die Erste ist dabei Inklusive. Alle weiteren kosten 60 Euro. Die oberen Variablen dienen nur als temporäre Speicher. Der erste `$scope,` auf den in Zeile 3 zugegriffen wird, setzt den Platzhalter in der View auf „1 Sprache (inklusive)“. Interessant ist die Stelle, in der `sprachenFkt` eine Funktion aufruft. Wird diese aufgerufen, so wird geprüft, welcher Übergabewert übergeben wurde. Wird ein „plus“ übergeben, dann weiß die Funktion, dass der Nutzer eine weitere Sprache wünscht. Nun wird die Anzahl der Sprachen (`$sprachen`) um `1` und der Preis ebenfalls um `60` erhöht. Danach wird die Anzeige über `$scope` aktualisiert. Nun steht hier „2 Sprachen (60 Euro)“. `2` ist hierbei die Variable `$sprachen` und die `60` ist

aus der variable `$tempSprachenPreis` herausgelesen worden. So lässt sich ein dynamischer Preiskalkulator zusammenstellen. Die weiteren Funktionen funktionieren sehr ähnlich und sind im Anhang zu finden.

Kommen wir nun zum Routing. Routing bedeutet, dass wir der Seite die Möglichkeit geben, über Links die `ng-view` zu aktualisieren und anderen Inhalt, inklusive anderen Controllern, zu laden ohne dabei die komplette Seite neu laden zu müssen. Damit das funktioniert, muss das AngularJS-Plugin „angular-route“ eingebunden werden. Das haben wir schon mit den anderen Skripten in einem vorherigen Schritt erledigt. Diese Datei bleibt von uns komplett unberührt. Eine weitere Datei, welche allerdings von uns selbst angelegt wurde und noch leer ist, ist die routing.js. In diese Datei schreiben wir jetzt unser Routing.

Quelltext 28 ▶
Konfigurieren des RouteProviders
Auf CD: /homepage/init/routing.js

```
1  awaitsApp.config(function($routeProvider) {
2          $routeProvider
3
4                  // route for the HOME page
5                  .when('/home', {
6                        templateUrl : 'templates/home.php',
7                        controller  : 'homeController'
8                  })
9
10                 // route for the APPs page
11                 .when('/apps', {
12                       templateUrl : 'templates/apps.php',
13                       controller  : 'appsController'
14                 })
15
16                 // route for the WEB page
17                 .when('/web', {
18                       templateUrl : 'templates/web.php',
19                       controller  : 'webController'
20                 })
21
22                 // route for the IT-SERVICE page
23                 .when('/itservice', {
24                 templateUrl : 'templates/itservice.php',
25                       controller  : 'itserviceController'
26                 })
27
```

```
28              // route for the BLOG page
29              .when('/blog', {
30                    templateUrl : 'templates/blog.php',
31                    controller  : 'blogController'
32              })
33              .otherwise({redirectTo: '/home'});
34  });
```

Routing funktioniert im Grunde relativ simpel. Wir sprechen unsere Applikation über `awaitsApp` an und konfigurieren sie. Die Funktion, die wir danach aufrufen, benötigt den `routeProvider` als Übergabewert. Dieser wird benötigt, damit das Routing funktioniert. Der `RouteProvider` gibt uns dabei als Übergabewert einen String, welcher dem angegebenen Pfad entspricht (z.B. „awaits.eu/home“, in dem Fall ist „/home“ der Übergabewert). Nun definieren wir, wenn (`when`) der angegebene Pfad `/home` ist, dann lade das Template „templates/home.php“ und den Controller `homeController`. Hier wird nun angegeben, welcher Controller bei welcher Unterseite geladen werden soll. Das machen wir analog für alle Unterseiten. Außerdem können wir am Schluss angeben, was passieren soll, wenn ein anderer Pfad (`otherwise`) angegeben wurde. In diesem Fall soll sich das Routing verhalten als wäre `/home` eingegeben worden.

Das ist schon das ganze Hexenwerk. Es wird einfach, je nach angegebenem Pfad, in der Adresszeile eine andere PHP-Datei in die `ng-view` geladen und ein neuer Controller intialisiert. Dadurch ist es möglich über einen normalen Link `<a href="#blog" >Blog</a>` das entsprechende Routing aufzurufen. Es ginge in dem Fall natürlich auch der Link `<a href="http://awaits.eu/blog">Blog</a>`. Allerdings ist die andere Schreibweise eleganter.

Nachdem die Seite soweit funktioniert können wir uns an den Inhalt der Seite machen. Die Navigation, die Verlinkungen und das Routing funktionieren soweit. Auch die Controller sind schon angelegt. Nun müssen wir nur noch den Inhalt verarbeiten.

24 Erstellen der Unterseiten

Wir haben das Grundgerüst gebaut und die Programmierlogik dahinter implementiert. Jetzt können wir uns die einzelnen Templates genauer anschauen, welche in die View (Ansicht) hineingeladen werden sollen.

Wir haben ein Grundgerüst gebaut, welches im Grunde nur eine einfache HTML-Seite ist. An dem Bereich, der sich ändern soll, haben wir einen leeren DIV-Container platziert. Dieser leere Container hat das Attribut `ng-View` bekommen, wodurch wir nun die Möglichkeit haben mittels AngularJS und dem Routing unterschiedliche Templates in diesen DIV-Container zu laden, je nachdem welcher Pfad in die Adresszeile eingegeben wird.

Nun müssen wir noch den Inhalt der Webseite, welcher geladen werden soll, erstellen. Das ist teilweise simpler HTML-Code, manchmal aber ein dynamischer Teil der Webseite. In diesem Fall wird dann genauer darauf eingegangen.

Die Template-Dateien, die hier geladen werden und in dem Verzeichnis „templates“ liegen, sind ganz normale .php Dateien mit schlichtem HTML-Code. Dabei brauchen wir keinen Header oder ähnliches. Der Inhalt wird so geschrieben, als würde man diesen aus dem DIV-Container des Grundgerüstes herauskopieren.

Es gibt dabei ein Grundschema, nach dem ein Beitrag aufgebaut ist.

Quelltext 29 ▶
Grundaufbau eines Menüpunkts
Auf CD: /homepage/templates/home.php

```
<div class='parallax_bg' id='about' data-
type='background'>
  <div class='header_parallax'>
        <p>Über mich</p><br>
        <p>Sascha Schnetz</p><br>
  </div>
</div>
<div class='home_normal'>
  <section class='textContainerLeft' style='background:
  url("bild.png") left no-repeat;'>
        <h1>Überschrift des Beitrags!</h1>
        <p>Text des Beitrags</p>
  </section>
</div>
```

Jeder Beitrag besteht aus einem Header. Dieser beinhaltet ein Hintergrundbild, welches parallax scrollt. Das bedeutet, es scrollt in einer anderen Geschwindigkeit als die restliche Seite, dadurch entsteht eine gewisse Tiefe.[31] Außerdem besteht der Kopfbereich aus einer Überschrift des Abschnitts (hier „Über mich“), und zwei weitere Kommentare, wel-

31 http://webmagazin.de/web/Parallax-Scrolling-erweckt-Short-Story-New-York-Times-zum-Leben-167850 (Stand: 21.07.2014)

che unter der Überschrift angezeigt werden. Unter dem Header gibt es immer den Textbereich. Dieser gliedert sich in einen Bild und einen Text-Bereich. Jeder Abschnitt ist dabei so aufgebaut, dass links ein Text steht und rechts ein Bild welches den Text verstärkt. Im nächsten Abschnitt ist es genau anders herum. Dann befindet sich das Bild links und der Text rechts. Jeder Abschnitt hat seinen eigenen Header. Dieser ist nicht dynamisch und wird statisch so im Klartext in das Dokument geschrieben. Der Inhalt darunter kann dynamisch sein.

24.1 Home

Die Home-Seite ist weitestgehend statisch. Auf ihr befindet sich außer einem Kontaktformular nur statischer HTML-Code. Wir werden noch kurz auf das Kontaktformular eingehen. Zuvor schauen wir uns aber erst einen weiteren, relativ interessanten Bereich an.

Die Rede ist von dem Bereich „Kenntnisse“. Dieser lässt sich dank Bootstrap und den dort definierten Progressbars (Fortschrittsbalken) wunderbar darstellen. Der Grundgedanke bei den Kenntnissen war, dass der Kunde sehen soll, wo meine Stärken und Schwächen liegen. Um diese visuell zu betonen bietet sich nichts besser an als eine farbliche Anzeige. Und da Bootstrap die Progressbars ohnehin im Repertoire hat wäre es unnötige Arbeit diese selbst nachzubauen.

Um die Progressbars verwenden zu können brauchen wir zwei DIV-Container. Einen für den äußeren Rahmen und einen für den farbigen Fortschrittsbalken. Dazu benötigt der äußere Container die Klasse `progress` und die Anzeigeart, z.B. `progress-striped` für einen gestrichelten Balken. Mit einem weiteren Attribut `style` können wir noch die Breite des Balkens definieren. Nun kommt der innere Container. Dieser bekommt die Klasse `progress-bar` und eine vordefinierte Farbe, wie zum Beispiel die Klasse `progress-bar-success` für einen grünen Balken oder `progress-bar-warning` für einen gelben Balken. Ebenfalls im Style-Attribut legen wir noch über `width` die Breite in Prozent fest. Haben wir von Adobe Flash eine Kenntnis von 90%, dann geben wir dem Style-Attribut den Wert `width: 90%;` Ein kompletter Balken mit Bezeichner der Kenntnis darüber sieht dann wie folgt aus:

```
1 <div class='progress progress-striped active' style='max-
2 width: 400px;'>
3   <div class='progress-bar progress-bar-success'
4 role='progressbar' style='width: 80%'>
5         80%
```

◀ Quelltext 30
Einbinden der Fortschrittsbalken
Auf CD: /homepage/templates/home.php

```
6     </div>
7  </div>
```

Die 80%, welche innerhalb des zweiten DIV-Containers steht, gibt den Text an, welcher auf dem Balken stehen wird.

Der nächste interessante Punkt im Home-Bereich der Webseite ist das Kontaktformular. Es soll den Besuchern die Möglichkeit geben entweder eine Beschwerde, einen Auftrag oder allgemeine Fragen einzureichen ohne dabei über ein eigenes Mailprogramm agieren zu müssen. Schlichte Angabe von Name und Emailadresse reichen aus, um das Formular abzuschicken.

Das Formular ist dabei ein absolutes Standard-Formular, wie es im Rahmen des Studiums schon oft verwendet wurde, welches keine großen Besonderheiten hat. Es ist allerdings zu erwähnen, dass das Formular vor dem Absenden überprüft wird. Das geschieht über eine in dem Form-Tag definierte Eigenschaft `onSubmit`. Die dort definierte Funktion `onSubmit="return pruefeMail()"` ruft erst die Funktion auf und nur wenn diese Funktion den Wert `true` zurücksendet, wird das Formular abgesendet. Die Funktion, welche die Email überprüft ist in einem ausgelagerten JavaScript-Dokument definiert und sieht wie folgt aus:

Quelltext 31 ▶
Die Funktion zum überprüfen der Emails
Auf CD: /homepage/lib/js/script.js

```
1  function pruefeEmail(){
2    if (awaitsFormular.email.value != awaitsFormular.
3    email2.
4    value) {
5          alert('Die angegebenen Emailadressen stimmen
6    nicht
7          überein!');
8          return false;
9    }else return true;
10 }
```

Die Funktion überprüft, ob die Emailadressen übereinstimmen. Wenn der Nutzer zwei unterschiedliche Emailadressen in das Formular eingibt wird der Text „Die angegebenen Emailadressen stimmen nicht überein!“ in einem Fenster angezeigt. Wenn das aber nicht der Fall ist und beide Emailadressen übereinstimmen, so wird das Formular abgesendet.

Bevor die Email verschickt wird, muss das Formular erst verarbeitet werden. Das geschieht über PHP. Dazu binden wir in unserem Grund-

gerüst eine weitere Datei ein. Und zwar in der ersten Zeile vor allem anderen. Wir binden die Datei init.php über die Zeile `<?php include ‚init/init.php‘ ?>` ein. Dadurch können wir nun allen PHP-Code in einer separaten Datei verarbeiten.

Wir überprüfen in unserer init.php, ob das Formular abgeschickt wurde. Da das Formular sich selbst aufruft wird überprüft, ob als POST das Attribut `Formulartyp` übergeben wurde und wenn ja, ob der Wert dieses Formulartyps „kontakt“ ist.

◄ Quelltext 32
Verarbeiten des Kontaktformulars uns senden der Email
Auf CD: /homepage/init/init.php

```
1  if(isset($_POST['formulartyp']) && $_POST[
2  'formulartyp'] == 'kontakt'){
3    $zieladresse = 'schnetz@awaits.eu';
4    $absenderadresse = $_POST['email'];
5    $absendername = $_POST['vorname'] . ' ' . $_POST[
6    'nachname'];
7  $dankeText = 'Wir haben ihre Anfrage erhalten und melden
8  uns über die angegebene Emailadresse '.$absenderadresse.
9' bei ihnen zurück, sobald ihre Anfrage bearbeitet wurde.';
10
11   $betreff = $_POST['kategorie'];
12   $nachricht = $_POST['nachricht'];
13
14   $header = array();
15 $header[] = 'From: '.mb_encode_mimeheader($absendername,
16   'utf-8', 'Q').' <'.$absenderadresse.'>';
17   $header[] = 'MIME-Version: 1.0';
18   $header[] = 'Content-type: text/plain; charset=utf-8';
19   $header[] = 'Content-transfer-encoding: 8bit';
20
21     $mailtext = ' ';
22
23     $mailtext .=     'Sie haben eine Email mit dem Betreff
24     \'.$betreff.'\' über ihr Kontaktformular
25     erhalten. \n \n'.
26          'Pflichtangaben: \n'.
27          'Name: '.$absendername.'\n'.
28          'Email: '.$absenderadresse.'\n \n'.
29          'Nachricht: \n'.
30          $nachricht.'\n';
31
32     mail(
33          $zieladresse,
```

```
34          mb_encode_mimeheader($betreff, 'utf-8', 'Q'),
35          $mailtext,
36          implode('\n', $header)
37      ) or die('Die Mail konnte nicht versendet werden.');
38
39    echo '<script type='text/javascript'
40    language='Javascript'>
41          alert('.$dankeText.')
42          </script>';
43  }
```

Kopfdaten wie Empfänger, Absender, etc. werden festgelegt. Die Daten werden über POST weitestgehend aus dem Body des HTML-Response geholt. Danach werden diese Daten verarbeitet und die Mail „zusammengesetzt". Im Anschluss wird diese Mail über die PHP-eigene Funktion `mail();` verschickt. Als Bestätigung, dass die Email verschickt wurde, wird am Schluss noch eine Nachricht in einem Alert-Fenster ausgegeben.

24.2 APPs

Im Bereich APPs soll ein Showreel die aktuellen und vergangenen APP-Projekte aufzeigen. Außerdem gibt es in diesem Bereich Informationen zum Workflow, dem Preis und die Möglichkeit sich als Beta-Tester anzumelden.

In diesem Kapitel widmen wir uns dem Showreel. Die Bereiche Workflow und Preis sind statische Bereiche und das Formular zur Beta-Anmeldung funktioniert in etwa genau so wie das Kontaktformular der Home-Seite.

Um den Workflow optisch ansprechend darzustellen, wird ein JQuery-Plugin eingebunden. Der Grundgedanke ist der, das die APPs als Bilder dargestellt werden. Wenn man mit der Maus über ein Bild fährt werden zusätzliche Informationen zu dieser App eingeblendet. Es gibt hierfür etliche Plugins. Wir werden hier aber das im Kapitel Frameworks getestete Plugin „SlipHover" verwenden. Es ist ein sehr kleines und doch ansehnlich gestaltetes sowie gut animiertes Plugin. Es gestattet die Angabe von mehreren Informationen, welche beim darüberfahren mit der Maus angezeigt werden. Es ist visuell ansprechend, da sich die Informationen von derjenigen Seite in das Bild schieben, von der mit der Maus über das Bild gefahren wird.

Realisiert wird das Ganze über die Einbindung der JavaScript-Datei `<script src="lib/js/jquery.sliphover.min.js"></script>`. Danach müssen wir nur noch unseren HTML-Code entsprechend formatieren:

```
<ul>
<li sliphover>
<a href='#'>
<img width='251' height='150' src='images/app/app1.jpg'
title='Projekt 1<br><hr>Status: Veröffentlicht<br>
Datum: April 2014' style='border: 1px solid #666;'/></a>

<div class='visible-xs' style='text-align: left;
margin: 10px 0 15px 5px;'>
  <table width='100%'>
        <tr><td width='80%'>Projekt:</td>
        <td>Projekt 1</td></tr>
        <tr><td>Status:</td>
            <td>Veröffentlicht</td></tr>
        <tr><td>Datum:</td>
            <td>April 2014</td></tr>
   </table>
</div>
</li>
</ul>
```

◀ Quelltext 33
Einbinden des Sliphover-Plugins
Auf CD: /homepage/templates/apps.php

Wir haben hier eine gewöhnliche ungeordnete Liste. Das Listen-Element hat das Attribut `sliphover`. Das ist grundsätzlich ungewöhnlich für Sliphover, da man in JavaScript eigentlich das jeweilige Element anspricht und dann sliphover für diese Element aktiviert. Hier müssen wir einen anderen Weg gehen. Der Grund ist der, dass wir mittels AngularJS den Content austauschen. Wenn wir hier ein Showreel laden, dann wird nur der Inhalt geladen, aber nicht die Funktionen ausgeführt. Das ist leider ein kleines Problem von AngularJS. Abhilfe schafft hier der Weg über neu definierte Attribute. Diese können über Direktiven definiert und dann in HTML-Tags verwendet werden. Wir schaen uns hier den Quelltext der Sliphover-Direktive an:

```
awaitsApp.directive('sliphover', function() {
  return {
        restrict : 'A',
        link : function(scope, element, attrs) {
  setTimeout(function () { element.sliphover() }, 10);
```

◀ Quelltext 34
Sliphover-Direktive erzeugen
Auf CD: /homepage/init/module.php

```
6            }
7    }
8 });
```

Damit wir das Attribut nutzen können müssen wir eine AngularJS Direktive definieren. Das funktioniert denkbar einfach. Dazu wird zuerst der Name des Attributs und dann die Funktion definiert. Die Funktion enthält die Parameter `restrict` und `link`. `Restrict` gibt an, um was es sich bei der Direktive handelt. Es gibt dabei 4 verschiedene Arten. A definiert ein Attribut, wie hier das Attribut `sliphover`. E ist ein Element. C steht für eine Klasse (Class). Die letzte Möglichkeit ist M und steht für ein Kommentar (comment).

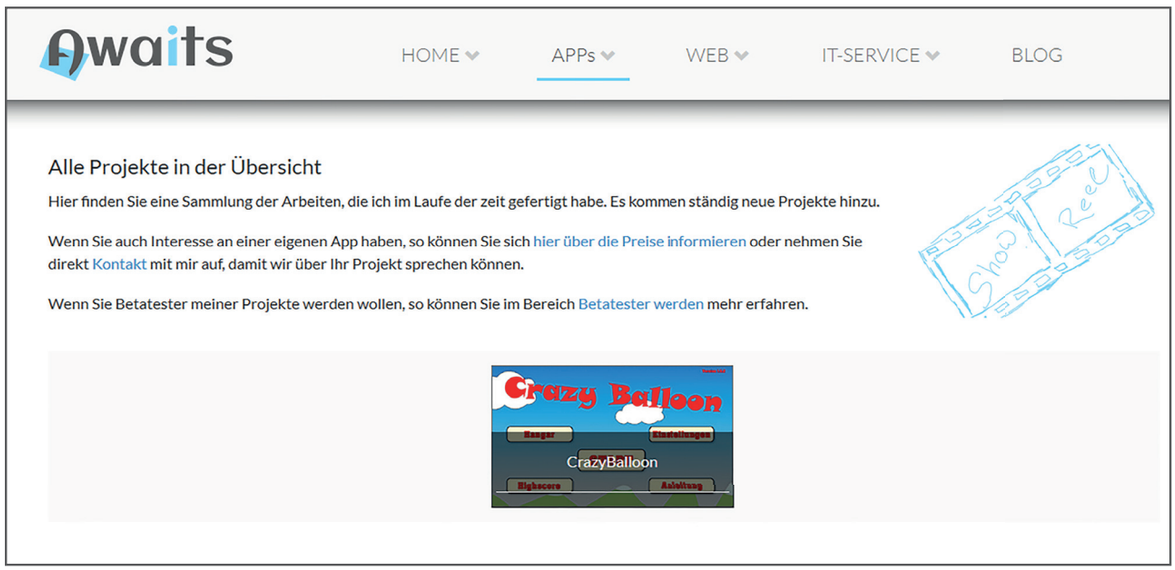


Abbildung 14 ▲
Das Showreel wird durch das Plugin Sliphover optisch aufgewertet

Nachdem wir unserer Direktive gesagt haben, dass es sich um ein Attribut mit dem Namen Sliphover handelt, müssen wir noch festlegen, was dieses Attribut machen soll. Dafür haben wir eine weitere Funktion definiert. Diese macht nichts anderes, als nach einem Timeout auf das Element mit dem Attribut `sliphover` die Funktion `sliphover()` auszuführen, welche in der Sliphover-JavaScript-Datei definiert ist.

Wenn diese Funktion ausgeführt wurde, funktioniert der Sliphover-Effekt.

Was nun noch auffällt ist der DIV-Container, der nochmal den gleichen Inhalt in einer Tabelle ausgibt. Dieser Container wird dank der Bootstrap-Klasse `visible-xs` nur auf kleinen Bildschirmen angezeigt und soll eine Darstellung der Inhalte bieten, wenn kein Hover möglich ist. Das ist bei Tablets und Smartphones der Fall. Allerdings ist das al-

les noch sehr statisch und wir müssten viel HTML-Code schreiben, um mehrere Listeneinträge (Projekte) einzutragen. Hier kann uns auch wieder AngularJS helfen, indem wir alle Projekte in einem Array speichern und im HTML-Code eine Schleife definieren, die für alle Listenelemente eine Ausgabe erzeugt.

```
1  <ul>
2    <li ng-repeat='projekt in appProjekte' sliphover>
3    <a href='{{projekt.link}}' >
4    <img width='251' height='150' src='images/app/{{
5  projekt.id}}.jpg' title='{{projekt.titel}}<hr>Status: {{
6    projekt.status}}<br>Datum: {{projekt.datum}}'
7    style='border: 1px solid #666;'/></a>
8     <div class='visible-xs' style='text-align: left;
9     margin: 10px 0 15px 5px;'>
10      <table width='100%'>
11   <tr><td width='80%'>Projekt:</td><td>{{projekt.titel}}
12         </td></tr>
13   <tr><td>Status:</td><td>{{projekt.status}}</td></tr>
14   <tr><td>Datum:</td><td>{{projekt.datum}}</td></tr>
15      </table>
16    </div>
17   <li>
18 </ul>
```

◀ Quelltext 35
Showreel dank AngularJS mit ng-repeat ausgeben
Auf CD: /homepage/templates/apps.php

Wir haben nun alle Angaben durch AngularJS Platzhalter ersetzt. Durch das Attribut `ng-repeat` können wir ein Array (hier: projekte) auslesen und in einer Variable (hier: projekt) zwischenspeichern. Es wird automatisch das Array durchgegangen und jedes Element in der Variable zwischengespeichert. Nach der Ausgabe wird das nächste Element in der Variable gespeichert. Das ist sehr elegant von AngularJS gelöst.

24.3 WEB

Der WEB-Bereich soll Kunden einen Überblick über die Erstellung von Webseiten geben. Dabei steht wieder ein Showreel zur Verfügung, welches nach dem gleichen Schema wie das der APPs funktioniert. Was allerdings hier eine Besonderheit darstellt, ist der Preiskalkulator. Dieser gibt den Kunden die Möglichkeit einen Preis für ein Webprojekt kalkulieren zu lassen. Das dient den Kunden als Richtwert für ein geplantes Projekt.

Wie das Array und die Funktionen hinter dem Preiskalkulator aussehen haben Sie schon in einem vorherigen Kapitel gesehen, in dem wir uns die Controller-Datei genauer angesehen haben. Schauen wir uns hier noch den Quellcode im HTML-Dokument an:

Quelltext 36 ▶
Der Preiskalkulator
Auf CD: /homepage/templates/web.php

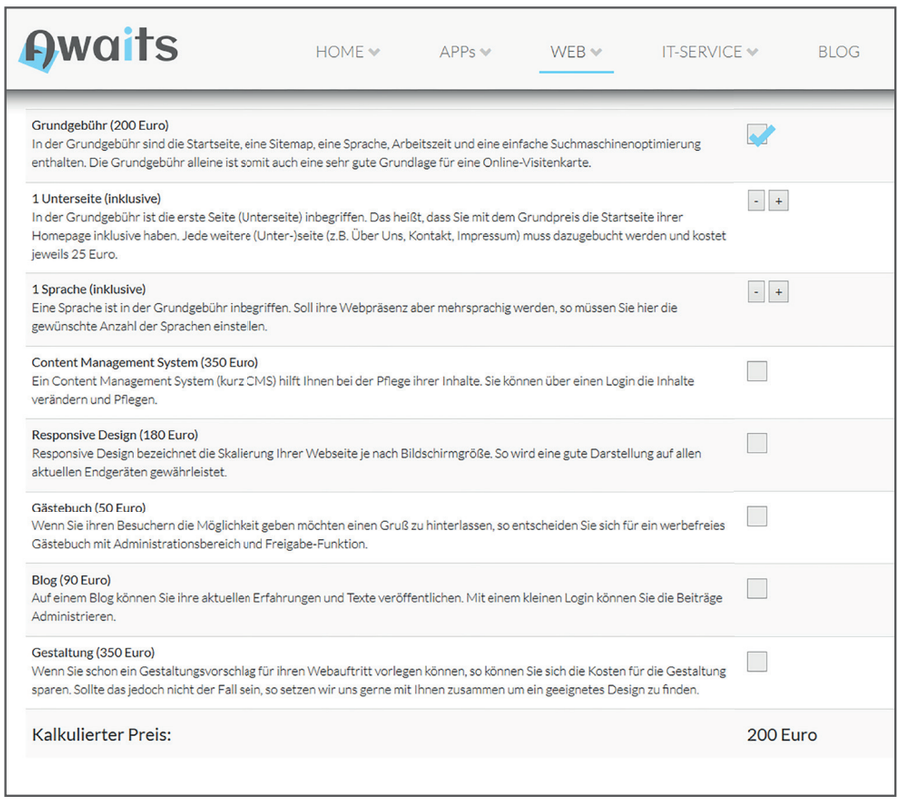
```
<table id='preiskalkulator' class='unselectable table table-striped'>
            <tr>
                <td width='900px'><label for='Grundgebuehr'><strong>Grundgebühr (200 Euro)</strong><br/>
                <span class='pkSubline'>In der Grundgebühr sind die Startseite, eine Sitemap, eine Sprache, Arbeitszeit und eine einfache Suchmaschinenoptimierung enthalten. Die Grundgebühr alleine ist somit auch eine sehr gute Grundlage für eine Online-Visitenkarte.</span></label>
                </td>
                <td><div class='checkboxPK'><input type='checkbox' value='200' id='Grundgebuehr' checked disabled/>
                <label for='Grundgebuehr'></label>
                </div></td>

            </tr>
            <tr>
                <td><label ng-click='unterseitenFkt(„plus“)'><strong>{{anzUnterseiten}}</strong><br/>
                <span class='pkSubline'>In der Grundgebühr ist die erste Seite (Unterseite) inbegriffen. Das heißt, dass Sie mit dem Grundpreis die Startseite ihrer Homepage inklusive haben. Jede weitere (Unter-)seite (z.B. Über Uns, Kontakt, Impressum) muss dazugebucht werden und kostet jeweils 25 Euro.</span>
                </label></td>
                <td><button ng-click='unterseitenFkt(„minus“)'>-</button>
                <button ng-click='unterseitenFkt(„plus“)'>+</button></td>
```

```
39              </tr>
40              <tr>
41                <td><label ng-click='sprachenFkt(
42                „plus")'><strong>{{anzSprachen}}
43                </strong><br/>
44                  <span class='pkSubline'>Eine Sprache
45                 ist in der Grundgebühr inbegriffen. Soll
46                ihre Webpräsenz aber mehrsprachig werden,
47                so müssen Sie hier die gewünschte Anzahl
48                der Sprachen einstellen.</span></label>
49                  </td>
50                <td><button ng-click='sprachenFkt(
51                „minus")'>-</button>
52                  <button ng-click='sprachenFkt(„plus")
53                 '>+</button></td>
54              </tr>
55              <tr ng-repeat='preis in preise' >
56                  <td><label for='{{preis.leistung}}'>
57              </strong>{{preis.leistung}} ({{preis.wert}}
58                  Euro)</strong><br/>
59                    <span class='pkSubline'>{{preis.
60                    beschreibung}}</span></label></td>
61
62                   <td>
63                    <div class='checkboxPK'>
64                        <input type='checkbox' value='
65                        {{preis.wert}}' id='{{preis.
66                        leistung}}' ng-
67                       click='kalkuliere Preis(preis.wert,
68                        preis.id, preis.ausgewaehlt)' />
69                        <label for='{{preis.leistung}}'>
70                        </label>
71                    </div>
72                  </td>
73              </tr>
74              <tr style='border-top: 1px solid #999;'>
75              </td><h1>Kalkulierter Preis:<br/><span
76              class='visible-sm visible-xs'>{{endpreis}}
77              </span></h1></td><td><h1  class='hidden-sm
78              hidden-xs'>{{endpreis}}</h1></td></tr>
79            </table>
```

Der Quellcode des Preiskalkulators sieht sehr verwirrend aus. Doch gehen wir Schritt für Schritt vor, um das besser zu verstehen.

Der komplette Preiskalkulator ist in einer Tabelle untergebracht. Die Tabelle bekommt zur besseren Darstellung die Bootstrap-Klassen `unselectable table table-stripped`. Danach kommt die Grundgebühr. Das komfortable hierbei ist, dass es sich um eine nicht veränderbare Eintragung handelt und wir diese daher ohne jegliche Platzhalter niederschreiben können. Danach kommt die Anzahl der Unterseiten. Diese werden, wie auch die Grundgebühr, niedergeschrieben. Allerdings gibt es hier zwei Buttons, welche die Anzahl erhöhen und reduzieren lassen. Ein Klick auf einen der Buttons ruft entsprechend eine der im Controller definierten Funktionen auf. Je nach Übergabewert („plus“ oder „minus“) wird dann die aktuelle Anzahl der Seiten verändert. Das Gleiche geschieht im Folgenden mit den Sprachen.

Abbildung 15 ▶
Der Preiskalkulator lösst sich dank AngularJS wunderbar bedienen



Alle weiteren Eintragungen sind, wie schon beim Showreel, in einer `ng-repeat`-Direktiven untergebracht und werden in einem Array im Controller definiert. Alle Eintragungen wie Titel, Beschreibung, Preis und Status (ob angewählt oder nicht) werden dort gespeichert. Klickt man auf einen Eintrag wird die entsprechende Funktion ausgeführt und der Preis neu berechnet.

24.4 IT-Service

Der Bereich IT-Service bietet weder ein Showreel noch einen Preiskalkulator. Es gibt auch keine weiteren Besonderheiten in diesem Bereich, weshalb er nicht weiter beachtet wird. Dieser Bereich bietet lediglich statischen Text als Information für potentielle Kunden, die nach IT-Dienstleistungen wie die Reperatur von Computern oder das Einrichten von Routern suchen.

24.5 BLOG-Programmierung mit PHP

Ein weiterer Schwerpunkt der Programmierung ist der Blog. Es gibt viele Möglichkeiten einen Blog auf einer Webseite einzubinden. Der einfachste Weg ist die Einbindung eines CMS oder eines fertigen Blogs mittels PHP. Im Rahmen dieser Thesis wurde aus persönlichem Interesse der komplette Blog selbst programmiert. Dabei wurde versucht einen ansehnlichen Blog mittels des im Rahmen des Studiums und meiner selbst gemachten Erfahrungen erworbenen Wissens zu programmieren.

Bevor es jedoch an die Programmierung geht, müssen noch einmal die wichtigsten Aspekte unseres zukünftigen Blogs besprochen werden. Unser Blog sollte auf der Startseite die enthaltenen Blogbeiträge anzeigen. Allerdings nicht alles, sondern nur einen kurzen Ausschnitt (etwa die ersten 300 Zeichen). Die Vorschau soll dabei aus der Überschrift, dem Datum und dem Autor bestehen. Unter jedem Artikel soll dann eine Schaltfläche „Weiterlesen“ platziert werden, welche den Artikel komplett anzeigt. Außerhalb der Artikel ist es äußerst wichtig, dass es die Möglichkeit gibt die Beiträge nach Jahren zu filtern. Desweiteren soll es eine Schaltfläche zum Login geben.

Nach einem erfolgreichen Login soll ein Administrationsbereich angezeigt werden, bei dem man die Artikel bearbeiten, löschen und neu erstellen kann. Über eine Logout-Schaltfläche kann man den Administrationsbereich verlassen.

Alle Blog-Beiträge werden in einer MySQL-Datenbank gespeichert und ausgelesen.

Wir fangen wie immer mit dem gleichen Grundgerüst an. Wir erstellen also ein Template blog.php, welches wir in der `ng-view` wie gewohnt laden lassen, sobald man in der Navigation auf Blog klickt. Innerhalb dieser Datei befindet sich der bereits bekannte Aufbau aus Header- und Content-Bereich. Allerdings gibt es hier nur einen einzigen sol-

chen Bereich. In den Content-Abschnitt kommt hier die Programmlogik des Blogs.

Da es ein langer Quellcode ist, werden wir hier Schritt für Schritt vorgehen und nicht den Quellcode von oben bis unten einfach Durchsprechen. Es ist hier nötig, den Quellcode aufzuteilen und Bereiche in logischer Reihenfolge zu besprechen.

Bevor wir anfangen müssen wir eine SESSION initialisieren. Diese ermöglicht das Speichern von Informationen über die Webseite hinaus, ähnlich wie bei Cookies. Dabei kann man einen Login-Status abspeichern oder Variablen übergeben, die normalerweise beim erneuten Laden einer Seite verloren gingen.

Quelltext 37 ▶
SESSION beginnen
Auf CD: /homepage/init/init.php

```
1 <?php
2   session_start();
3 ?>
```

Wenn wir einen Blog erstellen, welcher eine Datenbank auslesen soll, brauchen wir danach erstmal eine Verbindung zu dieser Datenbank.

Quelltext 38 ▶
Verbindung zur Datenbank herstellen
Auf CD: /homepage/init/init.php

```
1  require_once ('../init/konfiguration.php');
2  $verbindung = mysql_connect (MYSQL_HOST, MYSQL_BENUTZER,
3  MYSQL_KENNWORT)
4  or die ('keine Verbindung möglich. Benutzername oder
5  Passwort sind falsch');
6
7  mysql_select_db(MYSQL_DATENBANK)
8  or die ('Die Datenbank existiert nicht.');
9
10 $abfrage  = 'SET NAMES „utf8"';
11 $result = mysql_query( $abfrage);
```

Diese Vorgehensweise ist Standard im Webentwickler-Bereich. Zuerst wird eine Datei konfiguration.php verknüpft. Diese enthält in einer separaten, geschützten Datei die Login-Informationen. Diese werden bei `$verbindung` verwendet, um mit `mysql_connect` eine Verbindung aufzubauen. Wenn das funktioniert, wird noch mit `mysql_select_db` eine Datenbank ausgewählt. Alle diese Daten stehen in der konfiguration.php. Sobald alle Abfragen erfolgreich waren wird die `$abfrage` definiert. Dabei handelt es sich um den SQL-Befehl, welcher in einen String gespeichert wird. Dieser String wird dann noch über `mysql_query()` gesendet. Diese erste Abfrage ist nötig um der Datenbank den verwen-

deten Zeichensatz mitzuteilen. In den weiteren Schritten wird aber exakt so (mit `$abfrage` und danach mit `$result`) eine Datenbankabfrage gestellt. `$result` enthält dann die Ausgabe der Datenbank, welche wir verwenden können.

Wenn die Datenbankverbindung steht, wird eine Abfrage gesendet, die alle Artikel zurückliefert. Diese stellen wir dann in einer passenden Form dar.

```
1  if($_SESSION['aktiverArtikel'] == 0){
2     $abfrage = 'SELECT * FROM artikel WHERE datum like '%'.
3     $_SESSION['aktivesJahr'].'%' ORDER BY ID DESC';
4     $result = mysql_query( $abfrage );
5     while ($zeile = mysql_fetch_assoc( $result)){
6           echo '<h1>'.$zeile['titel'].'</h1>';
7     echo '<p><small>Artikel wurde am <strong>'.$zeile['
8     datum'].'</strong> von <strong>Sascha Schnetz</strong>
9           geschrieben</small></p>';
10          echo '<div class='blog_text'>'.substr($zeile[
11          'inhalt'], 0, 300).' ...</div><br/>';
12          echo '<form name='form' class='weiterlesen'
13          method='POST' action=''>';
14          echo '<input type='hidden' name='artikelNr'
15          value=''.$zeile['ID'].'' />';
16    echo '<button class='btn' onClick='form.submit();'>
17          Weiterlesen   <span class='glyphicon
18          glyphicon-align-left'></span></button></p>';
19          echo '</form><br><br>';
20    }
21 }
```

◀ Quelltext 39
POST-verarbeitung zur Anzeige aller Artikel
Auf CD: /homepage/init/init.php

In der ersten Zeile gibt es eine Abfrage, ob die SESSION-Variable `aktiverArtikel` gleich `0` ist. Diese Variable steuert die Anzeige. Ist die Variable `0`, dann werden alle Artikel angezeigt. Ist die Variable größer, dann wird der Artikel mit entsprechender Nummer angezeigt. Im Administrationsbereich gibt es dann noch die Variante, dass die Variable -1 beträgt, dann wird ein Formular für einen neuen Eintrag aufgerufen.

Wenn nun der aktive Artikel `0` ist, dann werden hier alle Einträge ausgegeben. Das geschieht durch die SQL-Abfrage `SELECT *` (wähle alles aus) `FROM artikel` (der Tabelle Artikel) `WHERE datum LIKE '%".$_SESSION[,aktivesJahr']."%'` (bei denen das Datum dem Sessionwert entspricht, welcher über ein anderes Formular geändert werden kann) `ORDER BY DESC` (und ordne diese so, dass der aktuellste Beitrag oben steht). Nun bekommen wir also eine Antwort, welche alle Artikel geordnet in die Variable `$result` speichert. Nun können wir mit einer While-Schleife alle Einträge nach und nach durchgehen. Dabei bekommt jeder Eintrag sein eigenes Formular, welches die Seite einfach neu lädt und eine `artikelNr` als POST übergibt. Wenn die Seite also neu lädt, und eine Variable mittels POST erhält, die `artikelNr` heißt, dann müssen wir das verarbeiten. Das geschieht über ein PHP-Skript, welches wir schon im Grundgerüst an oberster Stelle eingefügt haben. Wir erinnern uns an den Abschnitt der Home-Seite, dort haben wir die init.php schon mit etwas Inhalt gefüllt.

Jetzt müssen wir weitere Codezeilen einfügen, um unsere ankommende POST-Variable zu verarbeiten.

Quelltext 40 ▶
Verarbeitung der POST-Variablen „artikelNr“
Auf CD: /homepage/init/init.php

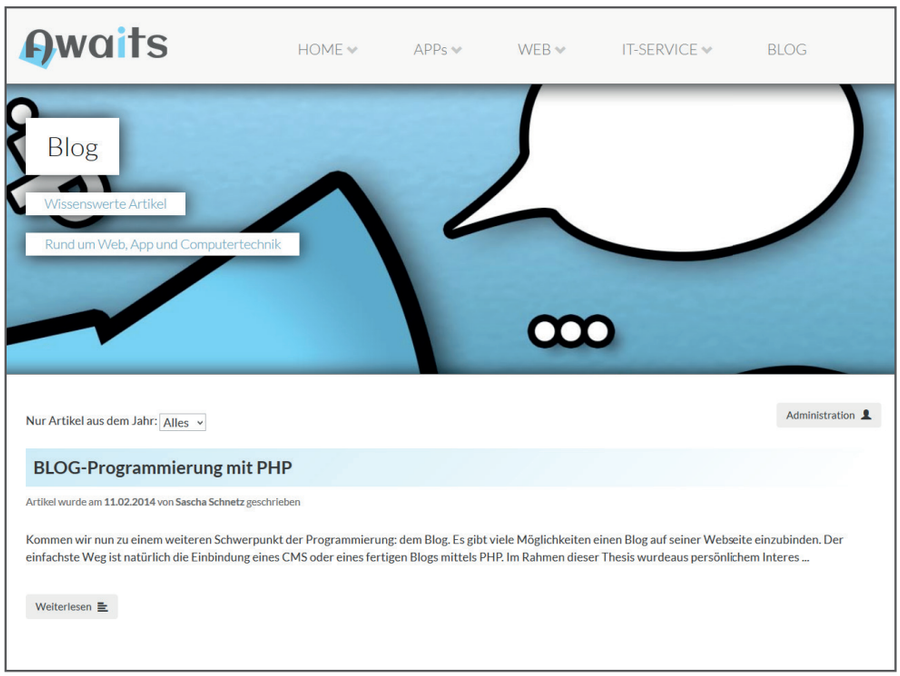
```
1 if(isset($_POST['artikelNr'])){
2    $_SESSION['aktiverArtikel'] = $_POST['artikelNr'];
3 }
4 else{
5    $_SESSION['aktiverArtikel'] = 0;
6 }
```

Mit wenigen einfachen Codezeilen ist es hier möglich eine POST-Variable abzufragen, und wenn sie vorhanden ist in eine SESSION-Variable zu schreiben. Dadurch bleibt der aktive Artikel gespeichert und wir können ihn entsprechend abfragen.

Doch warum müssen wir das hier in eine SESSION-Variable schreiben? Warum kann man nicht einfach den Post im Template abfragen und entsprechend reagieren? Der Schritt über die SESSION-Variable ist hier leider nötig, da der Inhalt im Content-Bereich nachgeladen wird und die Post-Variablen bis dahin nicht mehr im Gültigkeitsbereich liegen. Man kann auf diese dann nicht mehr zugreifen. Daher müssen wir alle POST-Ereignisse im Grundgerüst verarbeiten und in SESSION-Variablen speichern, sofern wir diese Daten noch in einem Template brauchen.

Damit das so funktioniert, müssen wir außerdem in der obersten Zeile unseres init.php-Dokuments ebenfalls eine SESSION starten. So steht hier ebenfalls in der ersten Zeile `session_start();`

Nun wird, wenn man den Blog aufruft, alles angezeigt was sich in unserer Datenbank befindet. Der Ablauf gestaltet sich so, dass wir „BLOG“ anklicken und die Seite neu geladen wird. In der init.php gibt es keine Übergabewerte, also wird die SESSION-Variable `aktiverArtikel` mit `0` initialisiert. In unserem Blog wird diese Variable überprüft, und es werden alle Artikel aus der Datenbank aufgerufen. Diese werden nacheinander ausgegeben.



◀ Abbildung 16
Die Oberfläche des BLOGs

Nun wollen wir noch eine Filterfunktion einbinen, welche uns die Möglichkeit gibt, die Blogeinträge nach Jahren zu sortieren. Das Gleiche ginge auch mit Namen oder Ähnlichem. Aber wir beschränken uns hier auf das Jahr. Wir legen in unserer blog.php über dem ersten Beitrag ein Formular an, mit dem man über ein Dropdown das Jahr auswählen kann.

```
1 <form class='blogFilter' action='' method='post'>
2 Nur Artikel aus dem Jahr:
3 <select name='jahrFilter' onChange='form.submit();'>
4   <option value='' > Alles
5   </option>
6   <option value='2014'> 2014
7   </option>
8 </select>
9 </form>
```

◀ Quelltext 41
Ein Formular zum Filtern der Beiträge nach der Jahreszahl
Auf CD: /homepage/templates/blog.php

Wir haben hier im Formular ein onChange definiert. Das bedeutet, dass, sobald der Inhalt geändert wird, das Formular abgesendet wird. Dann wird wieder in der init.php geschaut, ob die POST-Variable verarbeitet werden kann.

Quelltext 42 ▶
Verarbeitung des Jahre-Filters
Auf CD: /homepage/init/init.php

```
1 if(isset($_POST['jahrFilter'])){
2   $_SESSION['aktiverArtikel'] = 0;
3   $_SESSION['aktivesJahr'] = $_POST['jahrFilter'];
4 }else{
5   $_SESSION['aktivesJahr'] = '';
6 }
```

In dem Fall, dass die POST-Variable `jahrFilter` übergeben wird, wird dann die SESSION-Variable angepasst. Danach geschieht dasselbe wie auch zuvor bei den Artikeln.

24.6 BLOG-Administration

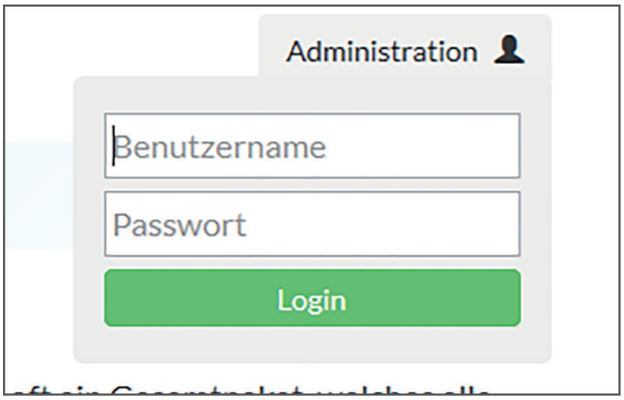
Unser Blog funktioniert soweit hervorragend. Was noch fehlt ist die Möglichkeit, unseren Blog zu administrieren. Die Administrationsoberfläche wird ebenfalls selbst mittels PHP programmiert. Wir fügen hierzu ein Login-Formular auf unserer Seite ein, welches uns die Möglichkeit gibt uns anzumelden.

Quelltext 43 ▶
Anzeige des Login-Fensters, wenn kein Nutzer angemeldet ist
Auf CD: /homepage/templates/blog.php

```
1 if($_SESSION['loginstate']==false){
2 echo '<div id='login'>
3 <button class='btn adminLink'
4 onClick='changeVisibility();'>Administration   
5 <span class='glyphicon glyphicon-user'></span>
6 </button>
7 <div id='loginFenster'>
8   <form action='' method='post'>
9         <input type='text' name='username'
10        placeholder='Benutzername'/><br>
11        <input type='password' name='password'
12        placeholder='Passwort'/><br>
13        <input type='success' class='btn btn-success'
14        value='Login' onClick='form.submit();'>
15  </form>
16 </div> </div>';
```

Die SESSION-Variable wird entsprechend auf `true` oder `false` gesetzt. Dadurch wird das Login-Formular nur angezeigt, wenn man

nicht eingeloggt ist. Dieses Login-Formular ist so aufgebaut, dass der Button „Administration“ ein Loginfenster einblendet. Standardmäßig ist es ausgeblendet. Nachdem man einen Nutzernamen und ein Passwort eingegeben hat, wird in der init.php überprüft, ob die Anmeldeinformationen stimmen. Ist das der Fall wird die SESSION-Variable `loginstate` auf `true` gesetzt.



◄ Abbildung 17
Das Login-Feld kann per Klick ein- und ausgeblendet werden

Schauen wir uns genauer an, wie das in der init.php aussieht:

```
1  $un = 'Sascha';
2  $pw = 'awaitsadmin!';
3  if(!isset($_SESSION['loginstate'])) $_SESSION[
4  'loginstate'] = false;
5  if(isset($_POST['username']) && isset($_POST[
6  'password'])){
7    if(($_POST['username'] == $un) && ($_POST[
8    'password'] == $pw)){
9          $_SESSION['loginstate'] = true;
10   }
11 }
```

◄ Quelltext 44
Verarbeitung des Anmeldevorgangs
Auf CD: /homepage/init/init.php

In den oberen Zeilen definieren wir unseren Nutzernamen (`$un`) und das Passwort (`$pw`). Das sind die hinterlegten Daten. Danach wird überprüft ob es schon einen Login-Status gibt oder nicht. Wenn kein Login-Status vorliegt, wird er standardmäßig immer auf `false` gesetzt. Beim Inhalt des POST wird geprüft und geschaut, ob die Variablen für Nutzername und Passwort übergeben werden. Wenn das der Fall ist, wird überprüft ob diese Angaben mit unseren Angaben übereinstimmen. Wurde eine Übereinstimmung von Eingabe und hinterlegten Daten festgestellt, so wird der Login-Status auf `true` gesetzt. Ab jetzt ist der Nutzer eingeloggt. Der Login-Status, welchen wir in der SESSION-Variable `loginstate` definieren, gibt somit immer an, welcher Inhalt gezeigt wird. Wenn wir also eingeloggt sind, können wir im Blog einen alternativen Inhalt anzeigen lassen. Das machen wir einfach über eine If-Bedingung:

Quelltext 45 ▶
Überprüfen des Loginstatus
Auf CD: /homepage/templates/blog.php

```php
if($_SESSION['loginstate']==false){
   ...
}else{
   ...
}
```

Sind wir bereits eingeloggt, so kommen wir in den „else“-Fall und können hier eine ganz neue Ausgabe definieren. Zum Beispiel, wie im vorherigen Abschnitt, die Ausgabe aller Artikel:

Quelltext 46 ▶
Anzeigen aller Artikel
Auf CD: /homepage/templates/blog.php

```php
//Übersicht aller Artikel
if($_SESSION['aktiverArtikel'] == 0){
   echo '
<form class='blogFilter' action='' method='post'>
   <input type='hidden' name='artikelNr' value='-1'/>
   <button class='btn btn-success' onClick='form.sub-
mit();'>
         Neuer Artikel     
         <span class='glyphicon glyphicon-plus'></span>
   </button>
</form><br>';
   $abfrage = 'SELECT * FROM artikel WHERE datum like '%'.
   $_SESSION['aktivesJahr'].'%' ORDER BY ID DESC';
   $result = mysql_query( $abfrage );
   while ($zeile = mysql_fetch_assoc( $result)){
         echo '<h1>'.$zeile['titel'].'</h1>';
         echo '<p><small>Artikel wurde am <strong>'.$zei-
le[
   'datum'].'</strong> von <strong>Sascha Schnetz</strong>
   geschrieben</small></p>';
         echo '<div class='blog_text'>'.substr($zeile[
   'inhalt'], 0, 300).'...</div><br/>';
         echo '<form name='form' class='weiterlesen'
   method='POST' action=''>';
         echo '<input type='hidden' name='artikelNr'
   value=''.$zeile['ID'].'' />';
         echo '<p><button class='btn btn-warning'
   onClick='form.submit();'>Bearbeiten   
    <span class='glyphicon glyphicon-pencil'></span>
   </button></p>';
         echo '</form><br><br>';
   } }
```

Im Prinzip ist alles sehr ähnlich. Wir fragen mit der SQL-Abfrage die Datenbank ab und geben die Artikel auf 300 Zeichen beschränkt aus. Danach kommt ein Button, der uns diesesmal die Möglichkeit gibt, die Artikel zu bearbeiten. Allerdings funktioniert es hier genau so, dass wir einfach die ID des Artikels als `ArtikelNr` übergeben und diesen Artikel anzeigen lassen. Bei der Anzeige wird nicht einfach der Artikel angezeigt, wie es vorher der Fall war. Die Inhalte werden stattdessen in einem Formular dargestellt, damit sie bearbeitet werden können.

```
1  //Formular zum Ändern von Einträgen
2  else{
3  $abfrage = 'SELECT * FROM artikel where ID = '.$_SESSION[
4     'aktiverArtikel'].' ORDER BY ID DESC';
5     $result = mysql_query( $abfrage );
6     while ($zeile = mysql_fetch_array( $result)){
7           echo '<form enctype='multipart/form-data'
8           name='form' class='bearbeiten' method='POST'
9           action=''>';
10          echo '<input type='text' name='changeTitel'
11          value=''.$zeile['titel'].'' />  ';
12          echo '<input type='text' name='changeDatum'
13          value=''.$zeile['datum'].'' />';
14          echo '<br><br><textarea id='changeInhaltArea'
15          type='text' name='changeInhalt'>'.$zeile[
16          'inhalt'].'</textarea><br><br>';
17          echo '<input type='hidden' name='MAX_FILE_SIZE'
18          value='3000000' />';
19          echo '<input type='file' id='uploadFile'
20          name='userfile' accept='image/*'/><br><br>';
21          echo '<label for='delete'><input type='checkbox'
22     id='delete' name='loescheArtikel' /> Diesen Artikel
23          unwiederuflich löschen!</label><br><br>';
24          echo '<input type='hidden'
25          name='artikelBearbeiten' value='update' />';
26     echo '<button class='btn btn-success' onClick='form.
27          submit();'>Übernehmen</button>  ';
28          echo '<button class='btn btn-danger'
29          onClick='history.go(-1);'>Abbruch</button></p>';
30          echo '</form><hr>';
31     }
32 }
```

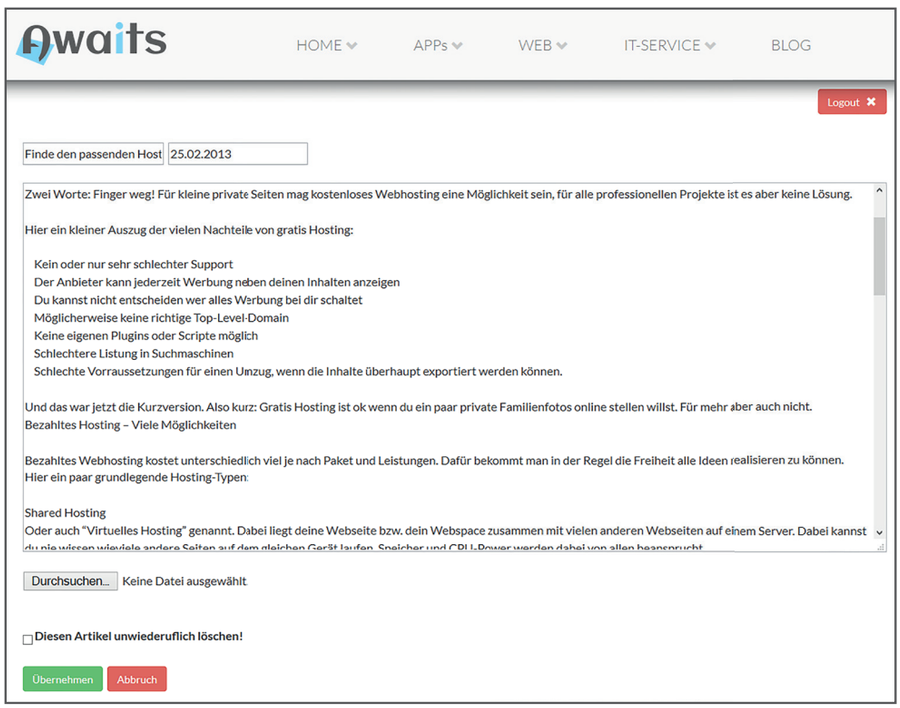
◀ Quelltext 47
Formular zum bearbeiten der Blog-Einträge
Auf CD: /homepage/templates/blog.php

Es funktioniert wie bei den anderen SQL-Abfragen auch. Die Antwort der Datenbank wird ausgegeben, allerdings in einem Formular. Wenn man nun die Änderungen vorgenommen hat, dann schickt man das Formular ab.

Die Daten werden wie üblich in der init.php abgefangen und verarbeitet. Kommt dort ein Bearbeitungswunsch an, so wird eine neue SQL-Abfrage mit den neuen Daten formuliert, welche die Datenbank-Einträge ändert.

Quelltext 48 ▶
Aktualisieren der Artikel in der Datenbank
Auf CD: /homepage/init/init.php

```
if($_POST['artikelBearbeiten'] == 'update' && !isset(
  $_POST['loescheArtikel'])){
  $abfrage = 'UPDATE artikel SET
        titel = ''. $_POST['changeTitel'].'',
        datum = ''.$_POST['changeDatum'].'',
        inhalt = ''.$_POST['changeInhalt'].''
        WHERE ID = '.$_SESSION['aktiverArtikel'];
  $result = mysql_query( $abfrage );

  if($_FILES['userfile']['name']){
        $uploaddir = 'images/blog/';
        $uploadfile = $uploaddir . basename($_FILES[
        'userfile'][
        'name']);

        if (move_uploaded_file($_FILES['userfile'][
        'tmp_name'], $uploadfile)) {
        } else {
        echo '<script>alert('Beim Hochladen des Bildes
              ist ein Fehler aufgetreten!');
              </script>';
        }
  }
}
else if($_POST['artikelBearbeiten'] == 'update' &&
isset($_POST['loescheArtikel'])){
  $abfrage = 'DELETE FROM artikel WHERE ID = '.$_SESSION[
  'aktiverArtikel'];
  $result = mysql_query( $abfrage );
}
mysql_close( $verbindung );
```



◀ Abbildung 18
Wenn ein Artikel bearbeitet werden soll, so wird aller Text in einem Formular dargestellt.

Die neue SQL-Abfrage wird gesendet und die Änderung wird vorgenommen. Genau so funktioniert es auch beim Löschen. Neben dem Bearbeiten und Löschen gibt es noch den Fall, dass man einen neuen Beitrag schreiben möchte. Dabei wird ein leeres Formular erstellt und angezeigt. Wenn dieses ausgefüllt wurde, werden die Daten wie gewohnt per POST weggeschickt. Die init.php agiert hier wie bei den anderen Fällen und übernimmt die Formulareingaben um eine SQL-Anfrage zu formulieren. Diese lautet allerdings `INSERT INTO`. Das Erstellen eines neuen Beitrags bietet neben dem normalen Beitrag aber noch die Möglichkeit, ein Bild hochzuladen. Das Bild kann von der lokalen Festplatte ausgewählt werden und dann mit einem Image-Tag (`<img>`) im Beitrag eingebunden werden. Ein Upload-Feld ist inzwischen kein Hexenwerk mehr. Es lässt sich mittels dem Attribut `type="file"` problemlos in jedes Formular einbinden. Bei unserem Formular wäre es die Zeile `<input type="file" id="uploadFile" name="userfile" accept="image/*"/>`. Durch das Attribut `accept` wählen wir die erlaubten Datei-Formate. In unserem Fall alle Bilddateien wie .gif, .png, .jpg, etc.

```
1 if($_FILES['userfile']['name']){
2   $uploaddir = 'images/blog/';
3   $uploadfile = $uploaddir . basename($_FILES[
4   'userfile']['name']);
5
6   if (move_uploaded_file($_FILES['userfile'][
```

◀ Quelltext 49
Upload einer Bild-Datei
Auf CD: /homepage/init/init.php

```
 7          'tmp_name'], $uploadfile)) {
 8    } else {
 9          echo '<script>alert("Es ist ein
10          ein Fehler aufgetreten!");</script>';
11    }
12 }
```

Mit den Zeilen in unserer init.php überprüfen wir, ob ein Bild mitgeschickt wurde. Wenn ja, dann wird es in dem Verzeichnis „images/blog/“ gespeichert (`$uploaddir`). Der Name unter dem das Bild gespeichert wird entspricht dem Originalnamen der Datei auf dem Datenträger.

Unser Backend ist fertig und funktionsfähig. Wir haben mit wenig Aufwand und ein bisschen PHP und HTML einen eigenen BLOG programmiert, welcher sich mit einem Login jederzeit warten lässt. Es ist somit nicht nur möglich Beiträge zu bearbeiten, löschen und zu erstellen, wir haben auch die Möglichkeit unseren Beiträgen ein Bild anzuhängen ohne den Weg über einen FTP-Clienten zu gehen.

25 Die Seite „responsive“ machen

Unsere Homepage ist soweit fertig. Nun müssen wir nur noch kontrollieren, ob unsere Seite auch den Anforderungen des responsiven Design gerecht wird. Unsere Navigation ist, da wir die Bootstrap-Navigation verwenden, schon in vollem Umfang responsive. Damit der Rest der Webseite auch responsive wird, lässt sich das mit einfachen Mitteln gewährleisten. Bootstrap liefert ein 12 Spalten Grid-System. Um dieses Grid-System verwenden zu können, müssen wir nur unseren DIV-Containern eine Klasse zufügen. Die Klasse `col-md-6` heißt, dass der Container bei der Bildschirmgröße „medium“ (`md`) die Breite `6` haben soll. Wir erinnern uns, dass das Grid-System 12 Spalten bietet, wenn wir also nun 6 Spalten nutzen, dann hat der Container eine Breite von 50 Prozent. Das Grid-System ist dabei vollkommen dynamisch und passt sich immer der Fensterbreite an. Wenn wir zwei Container mit den Klassen `col-md-6` definieren, dann werden sie automatisch nebeneinander angezeigt. Wenn der Bildschirm kleiner wird, dann rutschen die Container automatisch untereinander. Nun müssen wir nur unsere Webseite mit den Klassen aufrüsten.

Oft ist es auch gar nicht nötig die Bootstrap-Klassen zu verwenden. So können wir zum Beispiel bei unseren Abschnitten im Inhalts-Bereich die Breite 100%, aber maximale Breite 800 Pixel, per CSS zuweisen. Da-

durch ist der Container immer dem Bildschirm angepasst, es sei denn der Bildschirm wird breiter als 800 Pixel. In dem Fall bleibt der Container 800 Pixel breit.

Interessant sind die Bootstrap-Klassen zum Beispiel bei unseren Fortschrittsbalken im Bereich „Kenntnisse“ oder bei den zweispaltigen Formularen. Diese rutschen auf Smartphones untereinander.

Beispielhaft schauen wir uns hier den Bereich Kenntnisse nochmal genau an:

```
<div class='parallax_bg' id='Kenntnisse' data-
type='background'>
                <div class='header_parallax'>
                    <p>Kenntnisse</p><br>
                    <p>Abitur und Studium im
                  Informationstechnischen Bereich</p><br>
                    <p>Seit über 10 Jahren im Web-Bereich
                    tätig</p>
                </div>
            </div>

            <div class='home_normal'>
              <section class='textContainerRight'
              style='background: url(
              'images/home/kenntnisseArt.png') right no-
              repeat;'>
               h1>Meine Kenntnisse und Erfahrungen</h1>
                    <p>Im Laufe der Zeit habe ich viele
                    Erfahrungen in Computer- und Web-
                    Technologien entwickelt.</p>
                <p>Wie jeder Programmierer habe auch ich
                    Stärken und Schwächen. Hier können
                 Sie meinen aktuellen Wissensstand in den
                    einzelnen Bereichen einsehen.</p>
                </section>
                    <br/>
                    <div class='progressbars'>
                        <div class='col-md-6'>
                            <h1>Programme</h1>
                            Photoshop
                           <div class='progress progress-
                            striped active' style='max-
```

◀ Quelltext 50
Programmcode des Bereichs „Kenntnisse“
Auf CD: /homepage/templates/home.php

```
33                                  width: 400px;'>
34                                    <div class='progress-bar
35                                    progress-bar-success'
36                                    role='progressbar'
37                                    style='width: 90%'>90%
38                                    </div>
39                                  </div>
40                                  // ...
41                                </div>
42                                 
43                              </div>
44      </div>
```

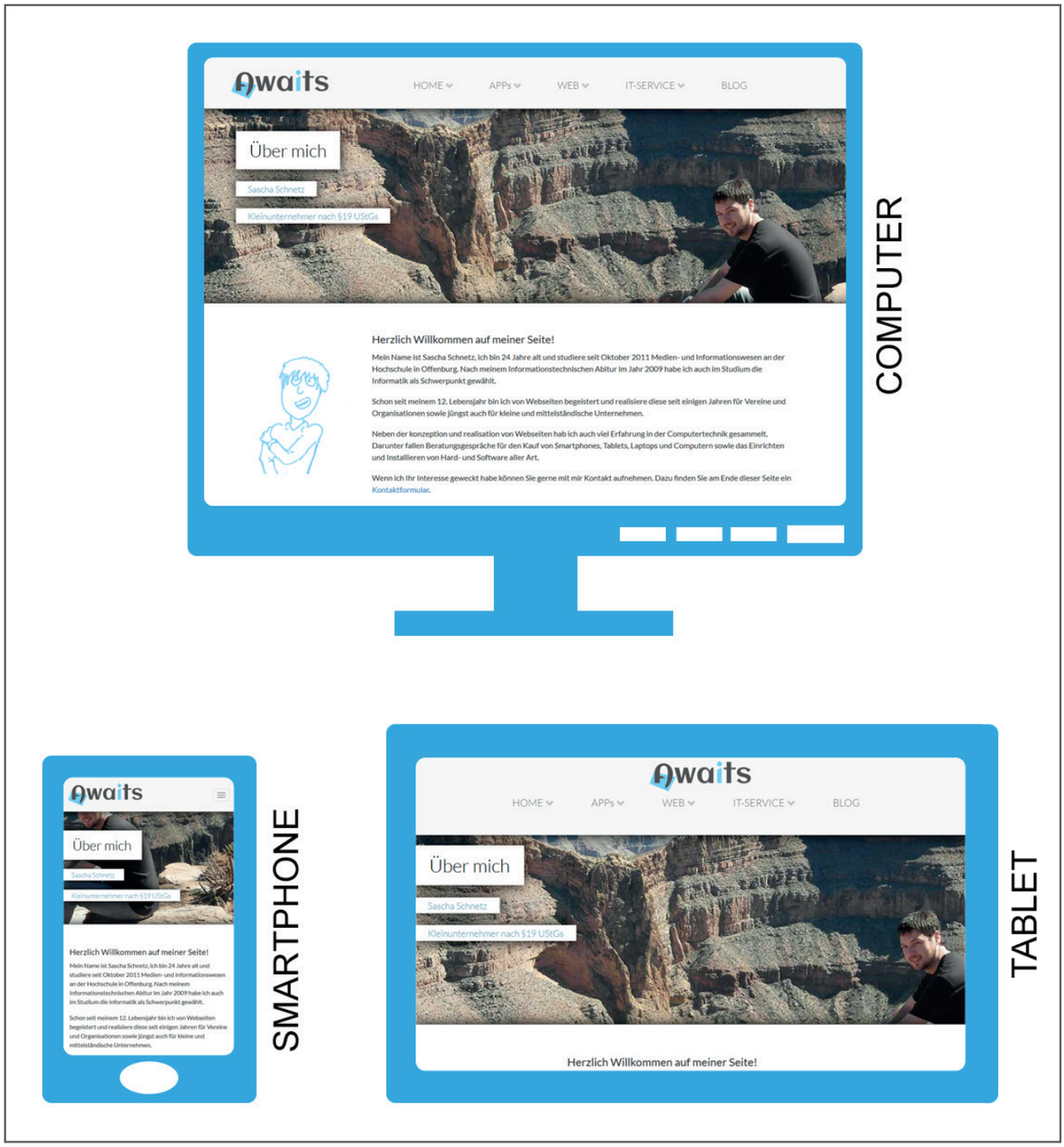
Wir sehen hier im Container „progressbars“, dass die Fortschrittsbalken die Bootstrap-Klassen `col-md-6` enthalten, um sich in Spalten zu organisieren. Der übergeordnete Bereich mit den Abschnitten (`<p>`) ist allerdings nicht mit den Bootstrap-Klassen definiert worden. Da jeder Navigationspunkt (Über mich, Kenntnisse, Links, ...) gleich aufgebaut ist, ist es hier sinnvoll die dynamische Anpassung über die CSS-Klassen zu definieren.

Quelltext 51 ▶
Anpassungen in der CSS-Datei
Auf CD: /homepage/lib/css/style.css

```
1  .home_normal p,
2  .blog_text{
3    max-width: 1150px;
4    margin: 0 auto 12px auto;
5    padding: 3px 0;
6  }
7
8  .home_normal section{
9    max-width: 1150px;
10   margin: 0 auto;
11 }
```

Die CSS-Klasse-Definitionen legen eine maximale Breite von 1150 Pixel fest. Dadurch sind die Container immer 100% breit, bis sie die maximale Breite erreicht haben. Durch das `margin`, welches hier rechts und links als „auto“ (automatisch) definiert ist, werden die Container darüber hinaus immer zentriert.

An dieser Stelle wird nicht weiter auf die Gestaltung mit CSS eingegangen, da es den Rahmen der Thesis bei weitem sprengen würde, den CSS-Code zu analysieren. Die Stylesheet-Datei befindet sich im Anhang auf CD und kann dort eingesehen werden.



Hier ist noch einmal die fertige Webseite auf den drei wichtigsten Ausgabemedien zu sehen. Auf dem Desktop-PC ist eine große Darstellung mit kleinem Bild neben den Artikeln zu sehen. Auf dem Tablet bricht die Navigation sinnvoll um, damit die Navigationspunkte weiterhin groß genug bleiben können, um mit den Fingern problemlos angetippt zu werden. Auf dem Smartphone wird auf die Bilder neben den Artikeln verzichtet und die Navigation in ein komplettes Dropdown-Menü verwandelt. Dadurch bleibt möglichst viel Platz für den Inhaltsbereich der Seite. Der Parallaxe-Effekt wird auf dem Smartphone nicht geladen, da die meisten Smartphones diesen nicht richtig darstellen können.

Abbildung 19 ▲
Die drei Versionen der fertigen Webseite

VII Schluss

„Der Computer rechnet mit allem - nur nicht mit seinem Besitzer."
- Dieter Hildebrandt

26 Fazit

Nach langer Arbeit wurde eine komplette Webseite erstellt, welche modernste Techniken und Ansätze verwendet. Die Seite ist dank Bootstrap nicht nur responsiv, sondern durch weitere Frameworks wie JQuery und AngularJS auch dynamisch und optisch ansprechend. Spielereien wie der Preiskalkulator oder das Showreel lassen die Seite professionell erscheinen. Der Blog wurde Zeile für Zeile selbst programmiert und bietet neben einer anschaulichen Darstellung der Beiträge auch ein Backend zum einfügen und löschen neuer Artikel. Die Verwendung von aktuellen Frameworks und professionellen Programmen hat nicht nur Spaß gemacht, sondern auch die im Studium erworbene Fähigkeiten gefestigt. Nachdem man ein Projekt in diesem Umfang realisiert hat, fühlt man sich beim bevorstehenden Berufseinstieg weit sicherer.

27 Ausblick

Die Webseite ist nun für den ersten Upload gerüstet. Allerdings ist die Arbeit noch lange nicht getan. Die Webseite wird sich mit dem noch jungen Unternehmen weiterentwickeln und die Möglichkeiten immer ausschöpfen, die die Welt des HTML, PHP und JavaScript mitbringen. Auch werden ständig neue Frameworks veröffentlicht, welche sich nach und nach den Weg in die Webseite bahnen werden.

Es gibt auch Funktionen, die noch nicht (aber sicher in absehbarer Zeit) auf der Webseite implementiert werden. So wäre eventuell ein Login für Kunden möglich, bei dem Sie sich über ihr aktuelles Projekt informieren können. Auch ein Backend zum Pflegen der Showreels wäre denkbar. Nachdem man sich beim BLOG angemeldet hat bleibt die SESSION immerhin erhalten bis man sich ausloggt. Egal auf welcher Seite man sich gerade befindet. Man könnte so die Administration von aktuellen und zusätzlichen Bereichen einpflegen. Das ist keine schwere, allerdings eine zeitaufwändige Aufgabe, weshalb sie zu einem späteren Zeitpunkt realisiert werden wird.

VIII Anhang

„Ich weiß zwar nicht, was ich da mache, aber es funktioniert. Und das ist die Hauptsache.“ - Peter Becker, deutscher Informatiker

28 Quellenverzeichnis

Hier finden Sie eine Auflistung aller Quellen, die zum erstellen dieser wissenschaftlichen Ausarbeitung verwendet wurden. Da es sich um eine sehr praxisnahe Arbeit handelt, sind viele Wissensbereiche über die Jahre im Rahmen des Studiums erlernt worden.

28.1 Internetquellen

http://api.jquery.com/ (Stand: 12.05.2014)
http://plugins.jquery.com/ (Stand: 12.05.2014)
http://jqueryui.com/tooltip/ (Stand: 12.05.2014)
http://fronteed.com/iCheck/ (Stand: 22.05.2014)
http://formstone.it/components/naver/ (Stand: 03.06.2014)
http://formstone.it/components/scroller/ (Stand: 03.06.2014)
http://formstone.it/components/wallpaper/ (Stand: 03.06.2014)
http://designers.hubspot.com/docs/snippets/design/implement-a-parallax-effect (Stand: 06.06.2014)
http://web-argument.com/2011/03/07/jquery-accordion-image-menu-plugin/#examples (Stand: 06.06.2014)
http://codepen.io/pederan/full/Hheuy (Stand: 15.06.2014)
http://darsa.in/sly/ (Stand: 06.06.2014)
http://wayou.github.io/SlipHover/ (Stand: 12.06.2014)
http://www.purecss.io (Stand: 22.06.2014)
http://www.angularjs.org (Stand: 22.06.2014)
http://getbootstrap.com/components/ (Stand: 02.07.2014)
http://getbootstrap.com/css/ (Stand: 02.07.2014)
http://angularjs.de/artikel/angularjs-tutorial-deutsch (Stand: 15.05.2014)
http://www.topcoat.io (Stand: 05.06.2014)
http://framework.zend.com/ (Stand: 19.05.2014)
https://cloud.google.com/developers/articles/angularjs-cloud-endpoints-recipe-for-building-modern-web-applications?hl=de (Stand: 21.07.2014)

http://www.itwissen.info/definition/lexikon/Framework-framework.html (Stand 21.07.2014)
https://docs.angularjs.org/guide/scope (Stand: 21.07.2014)
http://learn.jquery.com/plugins/ (Stand: 21.07.2014)
http://kritzelblog.de/erste-schritte-mit-bootstrap-3-0/ (Stand: 21.07.2014)
http://www.typolexikon.de/c/corporate-design.html (Stand: 21.07.2014)
http://devangelist.de/spa-webapi-knockout/ (Stand: 21.07.2014)
http://techcrunch.com/2012/11/30/why-enterprise-apps-are-moving-to-single-page-design/ (Stand: 21.07.2014)
http://webmagazin.de/web/Parallax-Scrolling-erweckt-Short-Story-New-York-Times-zum-Leben-167850 (Stand: 21.07.2014)

28.2 Videotutorials

STEYER, Ralph: JQuery: Webseiten dynamisieren und weiterentwickeln. Video2Brain. 2010.
ÜNLÜ, Saban: AngularJS – Crashkurs: Browser-basierte Anwendungen im MVC-Modell entwickeln. Video2Brain. 2013.

28.3 Literatur

DAMASCHKE, Giesbert: Jetzt lerne ich PHP und MySQL.
1. Aufl. [Germany] : Markt+Technik Verlag, 2012 .
FRANKE, Florian; **IPPEN**, Johannes: Apps mit HTML5 und CSS3 für iPad, iPhone und Android.
2. Aufl. Bonn : Galileo Press, 2013. S. 37-136, S. 341-430.
FLANAGAN, David: JavaScript : das umfassende Referenzwerk.
6. Aufl. Köln : O'Reilly, 2012.
GASSTON, Peter: Moderne Webentwicklung : geräteunabhängige Entwicklung - Techniken und Trends in HTML5, CSS3 und JS.
1. Aufl. Heidelberg : dpunkt-Verlag, 2014 .
JÄGER, Kai: AJAX in der Praxis.
1. Aufl. Berlin : Springer-Verlag, 2008. S. 27-117.
Kadlec, Tim: Praxiswissen Responsive Webdesign.
1. Aufl. Köln : O'Reilly, 2013.
MAURICE, Florence: PHP 5.4 & MySQL 5.5 - Der Einstieg in die Programmierung dynamischer Webseites.
1. Aufl. München : Addison-Wesley Verlag, 2012.
MÖHRKE, Carsten: Besser PHP programmieren : Handbuch professioneller PHP-Techniken.
3. Aufl. Bonn : Galileo Press, 2009.

MÖHRKE, Carsten: Zend Framework : das Entwickler-Handbuch.
1. Aufl. Bonn : Galileo Press, 2008. S. 13-58.
STÄUBLE, Markus: Programmieren für iPhone und iPad.
4. Aufl. Heidelberg : dpunkt.verlag GmbH, 2012. S. 65-71.
VDOVKIN, Andreas: jQuery : kurz & gut.
2. Aufl. Beijing : O'Reilly Verlag, 2011.

28.4 Bildquellen

Abbildung 1: https://angularjs.org/ (Stand: 22.06.2014)
Abbildung 3: http://fronteed.com/iCheck/ (Stand: 22.05.2014)
Abbildung 4: http://web-argument.com/2011/03/07/jquery-accordion-image-menu-plugin/#examples (Stand: 06.06.2014)
Abbildung 8: http://framework.zend.com (Stand: 19.05.2014)

29 Abbildungsverzeichnis

30 Quelltextverzeichnis

31 Abkürzungsverzeichnis

Abb.	Abbildung
Bsp.	Beispiel
CD	Compact Disc
CMS	Content-Management-System
CSS	Cascading Stylesheet
DIV	Division (dt. Bereich)
HTML	Hypertext Markup Language
img	Image (dt. Bild)
init	Initialisieren
JS	JavaScript
lib	Library (dt. Bibliothek)
MVC	Model-View-Controller
PHP	PHP Hypertext Preprocessor
z.B.	zum Beispiel