Konzipierung und Umsetzung einer Webpräsenz

Bachelorarbeit

erstellt am: 20. Juli 2017

Hochschule Anhalt

Name:	Fabian Klautke
Matrikelnummer:	4054946
Studiengang:	Angewandte Informatik - Medieninformatik
Erstgutachter:	Prof. Dr. Alexander Carôt
Zweitgutachter:	Prof. Dr. Michael Worzyk

Inhaltsverzeichnis

1 Einleitung

In einer Zeit, in der das Internet eine entscheidende Rolle in unserem Alltag spielt und uns auf unzähligen Endgeräten zur Verfügung steht, ist es so leicht wie noch nie, sich und seine Interessen einer großen Masse von Leuten zu präsentieren. Neben den offensichtlichen Möglichkeiten, wie ein Profil, bei den vielen Sozialen Medien, ist eine eigene Webpräsenz ein hervorragender Weg seine Person und/oder seine Interessen vorzustellen. Vorteil einer eigener Website ist, dass man seine Inhalte auf eine Art und Weise präsentieren kann die einem zusagt und nicht von den Einschränkungen und Vorgaben, die Anbieter Sozialer Medien durchsetzen, eingeengt zu sein. Daraus ergab sich die Motivation für diese Bachelorarbeit.

1.1 Motivation

Im Internet gibt es zahlreiche Angebote für sogenannte Website-Baukästen, die einen Weg bieten, sich auf einfache Art und Weise eine eigene Webpräsenz zu erstellen. Da Nutzer bei diesen Angeboten allerdings nur aus einer begrenzten Anzahl von Elementen wählen können, um die eigene Website zu realisieren, bleibt es nicht aus, dass Webpräsenzen, die mit Hilfe eines solchen Baukastens erstellt wurden, sich zu einem gewissen Grad ähneln und einen individuellen Aspekt vermissen lassen. Zudem hat der Nutzer bei der Verwendung von Website-Baukästen „meist keinen Zugriff auf den Quellcode“ [Fahl 2010 S. 83] und kriegt somit keinen Eindruck von dem Gerüst aus HTML-Code(4.2) und Stylesheets(4.3), auf das eine Website aufbaut.

Die Motivation war also, im Rahmen der Bachelorarbeit eine Website von Grund auf selbst zu erstellen. Es soll dabei erarbeitet werden, was Nutzer bei der Konzipierung und Umsetzung ihrer Webpräsenz beachten bzw. vermeiden sollten und wie sie überprüfen können, ob sich die Nutzeroberfläche ihrer erstellten Website wie gewünscht verhält.

1.2 Rahmen der Arbeit und Zielgruppe

Im Rahmen der Arbeit wird ein kurzer Einblick in die Entwickelung von Webdesign und -entwicklung gegeben, sowie eine Übersicht über die häufigsten Fehler bei der Konzipierung und Umsetzung einer Website. Zudem wird dargelegt, wie man ein verwertbares Konzept für seine Webpräsenz erarbeitet und dieses dann umsetzt und auf Funktionalität testet. Am Ende der Arbeit wird ein kurzes Fazit gezogen und ein Ausblick darauf gegeben, wie nun weiter verfahren werden kann.

Der Inhalt der Arbeit richtet sich hauptsächlich an Laien - das heißt Personen, die bisher keine oder nur sehr wenig Erfahrung in Sachen Webdesign und Webentwicklung haben. Dementsprechend wird ein einfach verständliches Vokabular verwendet und die angesprochenen Punkte ausführlich erläutert.

2 Grundlagen

In diesem Abschnitt werden die Grundlagen für die Bachelorarbeit gelegt. Es wird eine kurze Übersicht über die Entwicklung von Webdesign und Webentwicklung gegeben. Zudem wird angesprochen, welches die häufigsten Probleme bei der Konzeptionierung und technischen Umsetzung von Webpräsenzen sind.

2.1 Geschichte von Webdesign und -entwicklung

- 1990 - Tim Berners-Lee entwickelt am CERN die erste Website der Welt. Sie stellt das World Wide Web Project vor, das von Berners-Lee erdacht wurde.
- September 1991 - HTML wird zum ersten Mal öffentlich diskutiert. Die WWW talk mailing list wird eingerichtet, damit Interessierte ihre Ideen austauschen können.
- 1992 - Dave Raggett entwickelt HTML+, eine erweiterte Version von HTML.
- 1994 - Eine Gruppe von Wissenschaftlern um Dan Connolly stellt HTML 2 vor, mit allen Tags, welche bis dato eingeführt wurden.
- 1994 - Um HTML zu standardisieren und dessen Fragmentierung entgegen zu wirken gründen sich die IETF (Internet Engineering Task Force) und das W3C (World Wide Web Consortium).
- März 1995 - Dave Raggett veröffentlicht einen Vorschlag für HTML 3 welcher allerdings nicht von der IETF ratifiziert wird, da er als zu umfangreich erachtet wird.
- November 1995 - CSS wird zum ersten Mal erwähnt, als Werkzeug, um das Aussehen von Websites zu beeinflussen.
- Februar 1996 - das HTML Editorial Review Board ersetzt die HTML-Arbeitsgruppe der IETF. Dieses Review Board bestand aus Vertretern verschiedener Browserentwickler und aus Vertretern des W3C.
- 1996 - durch die Arbeit des Review Board entsteht zum ersten Mal ein allgemeiner HTML-Standard. Ebenfalls wird CSS 1.0 veröffentlicht.
- 1997 - HTML 3.2 wird als erste standardisierte und allgemein anerkannte HTML-Version veröffentlicht.
- 1998 - HTML 4.0 und CSS 2.0 werden veröffentlicht. Zum ersten Mal wird in dieser HTML-Version speziell auf CSS hingwiesen.
- 1999 - eine überarbeitete Version von HTML 4.0 wird als HTML 4.01 veröffentlicht. Dies ist zum Zeitpunkt dieser Arbeit die letzte ratifizierte HTML-Version.
- 1999 - Das W3C stoppt die Arbeit an HTML und wendet sich XHTML zu.
- 2004 - Mehrere Browserentwickler gründen die WHATWG (Web Hypertext Application Technology Working Group) und beginnen Ideen für HTML5 zu entwickeln.
- 2007 - Das W3C wendet sich ebenfalls der Entwicklung von HTML5 zu.
- 2011 - Eine überarbeitete Version von CSS 2.0 wird als CSS 2.1 veröffentlicht.
- heute - HTML5 und CSS 3.0 befinden sich weiterhin in Entwicklung und wurden bisher nicht als Standard ratifiziert.

[Vgl. CERN, Raggett 1998, WHATWG]

2.2 Konzeptionsprobleme bei der Erstellung von Webpräsenzen

Dieser Abschnitt erläutert einige der häufigsten Probleme, denen Laien bei der Konzipierung ihrer ersten Webpräsenz begegnen. Dabei wird die Webpräsenz unterteilt in Website, womit die Gesamtheit der Webpräsenz gemeint ist und Webseite, womit eine einzelne Seite der Webpräsenz gemeint ist.

2.2.1 Undurchdachte Navigation

Obwohl die Navigation eines der am häufigsten genutzten Elemente einer Website ist, werden ihr im Konzeptionsprozess oft nicht viele Gedanken gewidmet. Zum Beispiel mag eine Navigation, die lediglich Bilder verwendet, um die verschiedenen Navigationsziele zu symbolisieren für den Webdesigner schlüssig erscheinen, allerdings treten auch einige Probleme mit einem solchen Menü auf. Zum einen assoziiert nicht jeder Menschen die gleichen Sachen miteinander, zum anderen „können Sehbehinderte es nicht nutzen, da ihr Screenreader [...] nicht mit dem bebilderten Menü umgehen kann“ zudem „sind Suchmaschinen nicht in der Lage, das Menü zu indexieren“ [Rupf 2014, S. 45]. Allerdings können auch bei textbasierten Navigationsmenüs Fehler gemacht werden. Werden bei der Beschriftung der Navigation generische Begriffe verwendet, wird dem Nutzer nicht sofort klar, was er unter diesem Navigationspunkt zu erwarten hat. Ebenso sollten Fachbegriffe in Navigationen nur verwendet werden, wenn sicher ist, dass die Zielgruppe diese versteht. Des weiteren sollte bei der Navigation auf Animationen und versteckte Menüs verzichtet werden. Navigationen sollen Nutzer schnellstmöglich dahin bringen, wo sie hin möchten. Je effizienter sie gestaltet wird, desto wahrscheinlicher bleibt das Interesse der Nutzer vorhanden. Eine Ausnahme sind Kinder, diese wissen die Möglichkeit, eine Website mit Animationen und versteckten Sachen zu erforschen, zu schätzen [vgl. Nielsen 2006, S. 182]. Ein weiterer Fehler, der häufig bei der Navigation auf Webpräsenzen gemacht wird, ist das verwenden von Duplikaten. Dabei wird die Navigation an mehreren Stellen auf der Webseite angezeigt, z.B. am Anfang und am Ende der Seite. Oft wird angenommen, dass diese zusätzlichen Areale mehr Zugriffe auf die verlinkten Seiten generieren. Wenn ein User zum Beispiel am Beginn der Seite einen Link anklicken möchte, sich aber dazu entscheidet vorher den Rest der Seite anzuschauen, würde er am Ende der Seite vielleicht auf das Link-Duplikat klicken. Dies bedeutet aber nicht, dass er nicht auch wieder zum Beginn der Seite gescrollt hätte, wäre das Duplikat nicht da gewesen [vgl. Nielsen 2006, S. 189]. Statt für mehr Zugriffe sorgen Duplikate daher meist eher dafür, dass die Seite unübersichtlich und überfüllt wirkt.

Während die vorherigen Punkte hauptsächlich auf Fehler zum Aussehen der Navigation eingehen, ist auch der Inhalt oft eine Problemstelle. So werden die Themengebiete oft nach einem Schema sortiert, dass für den Websitebetreiber sinnvoll erscheint. Da dieser aber logischerweise in das Thema der Webpräsenz investiert ist, muss eine Themenanordnung die ihm intuitiv erscheint, von anderen Nutzern nicht genauso empfunden werden.

2.2.2 Kein einheitliches Layout

Eine Webpräsenz mit einem inkonsistenten Layout kann beim User schnell für Verwirrung und letztendlich zum Interessenverlust und Abkehr von der Website sorgen. Befindet sich die Navigation zum Beispiel plötzlich rechts, obwohl sie sich auf der Seite davor noch am oberen Rand befand, oder fehlt das Logo, das ebenfalls als Homebutton funktionierte, sorgt das dafür, dass die Nutzer sich mit der Bedienung der Seite befassen müssen, statt mit den Inhalten, wegen derer sie eigentlich die Seite aufgesucht haben. Allgemein sorgt ein chaotisches Layout dafür, dass Nutzer die Orientierung verlieren und sich lieber anderen Angeboten zuwenden. Denn es gilt, „im Internet werden die Benutzer zuerst mit der Qualität der Nutzung konfrontiert. Kann der Benutzer die gewünschten Informationen nicht zügig finden, so wechselt er die Website einfach“ [Burmester 2007, S. 246].

2.2.3 Inkonsequentes Farbschema

Das Farbschema einer Website ist ebenfalls eine häufige Fehlerquelle. Ein zu niedriger Kontrast zwischen Hintergrund und Schrift strengt das Auge an und erschwert das Lesen des Inhalts bei schlechten Lichtverhältnissen, wie z.B. Sonneneinstrahlung. Zudem nimmt im Alter die Fähigkeit des Auges ab, Kontraste wahrzunehmen, wodurch älteren Menschen die Benutzung der Website erschwert wird. Andererseits macht ein zu starker Kontrast das Lesen anstrengend. Gesättigte Farben für den Vorder- und Hintergrund zu verwenden ist ebenfalls problematisch. Selbst bei Komplementärfarben erschwert die Verwendung von gesättigten Farben es dem Auge die Schriften genau zu fokussieren, was das Lesen des Textes erschwert [Vgl. Baltzert 2004, S. 151]. Außerdem führt es zu Problemen, wenn Elemente mit der gleichen Funktion unterschiedliche Farben haben, oder auf Unterseiten die Farbe wechseln. Das Gesetz der Gleichheit, eines der Gestaltgesetze, welche 1923 von Max Wertheimer formuliert wurden, besagt, dass Objekte mit gleichen Eigenschaften, wie z.B. gleicher Farbe, von Menschen als eine Gruppe wahrgenommen werden [Vgl. Thesmann 2016 S. 225]. Wenn also Verlinkungen auf der Startseite mit einer blauen Schriftfarbe eingeführt wurden, aber auf einer der Unterseiten eine Verlinkung plötzlich eine grüne Farbe hat, wird Letzteres von Nutzern nicht direkt als Verlinkung wahrgenommen.

2.2.4 Keine einheitliche Schriftart

Eine passende Schriftart ist für eine Webpräsenz enorm wichtig. Auch dort muss auf einige Fehlerquellen geachtet werden. Zum Beispiel sollte bewusst sein, dass die Schriftarten, oder Fonts, die auf dem Entwicklungssystem vorhanden sind, nicht zwingender Weise auch auf den Systemen der Anwender existieren. Ebenso sollten nicht zu viele unterschiedliche Schriftarten auf einer Seite verwendet werden. Dies stellt einen groben typographischen Fehler dar und wird von der Mehrzahl der Anwender als unprofessionell empfunden. Mehr als zwei Arten sollten daher nicht verwendet werden [Vgl. Münz 2006, S. 120]. Bei der Benutzung von Webfonts stellen viele unterschiedliche Schriftarten ebenfalls ein Problem dar. Webfonts sind Schriftarten, welche extra so entwickelt wurden, dass alle Browser sie möglichst stilnah interpretieren können, unabhängig vom Betriebssystem. Da diese Webfonts daher oft von anderen Websites geladen werden, verlängern viele unterschiedliche Schriftarten unweigerlich die Ladezeit der Webpräsenz.

2.3 Technische Probleme bei der Erstellung von Webpräsenzen

Neben der Konzeption, können auch bei der darauf folgenden technischen Umsetzung Fehler gemacht werden. Einige der häufigsten werden in diesem Abschnitt angesprochen.

2.3.1 Frames

Frames wurden 1995 vom Browserentwickler Netscape eingeführt. Damals wurden sie dazu benutzt, um fixe Inhalte und scrollbare Inhalte auf derselben Seite darstellen zu können. Mittlerweile sind Frames von anderen Techniken abgelöst worden. Trotzdem sind sie weiterhin bei der Umsetzung einiger Webpräsenzen anzutreffen. Dies führt allerdings zu einigen Problemen. Da Frames, welche Inhalte darstellen, keine separaten Seiten sind, haben sie auch keine eigenen Verlinkungen. Dadurch wird zum einen die Navigation beeinträchtigt, da ein Zurück-Button nicht mehr die erwartete Funktion hat. Zudem haben die Frames somit auch keine individuelle Webadresse und können nicht als Lesezeichen gespeichert werden. Weiterhin erhöht die Verwendung von vielen Frames die Ladezeit der Webpräsenz, da pro Frame die Ressourcen vom Webserver geladen werden müssen.

2.3.2 Chaotische Linkbezeichnungen

Sowohl bei der Bezeichnung von Verlinkungen im Inhalt, als auch bei den URL-Adressen der Webpräsenz an sich, können schlecht gewählte Linkbezeichnungen zu Problemen führen. Neben dem Seitentitel ist die URL zum Beispiel das Erste, was Nutzer in einer Suchmaschine von der Webpräsenz sehen. Daher sind lange, unverständliche URLs oder URLs, welche lediglich eine Ansammlung von Zahlen und Buchstaben sind, wenig aussagekräftig. Ebenso sollte darauf geachtet werden, dass die URL-Struktur so angelegt wird, dass der Nutzer auf die nächst höhere Ebene geleitet wird, wenn er in der Adresszeile des Browser die Pfadangabe der aktuellen Unterseite manuell entfernt. Bei Verlinkungen im Inhalt der Webpräsenz sollte der „Seitentitel verwendet werden, auf den der Link zeigt. Redaktionelles Zutun ist bei der Linkbezeichnung nicht gefragt“ [Ewert 2001, S. 132]. Zudem sollte das Ziel der Verlinkung auch den Erwartungen des Nutzers entsprechen. Verlinkt man eine Institution oder Firma, so sollte sich mit einem Klick auf den Link die Website der Einrichtung öffnen. Dies bietet dem Nutzer weiterführende Informationen, die für ihn interessant sind. Er erwartet hingegen nicht, dass sich z.B. ein Bild des Firmensitzes oder eine Ortsangabe auf einer der zahlreichen virtuellen Maps im Internet öffnet. Der Nutzer erhält dadurch nicht die gewünschten Informationen, zudem könnte er zu dem Fehlschluss kommen, dass es keine Webpräsenz der Einrichtung gibt.

2.3.3 Keine Optimierung

Schlechte Optimierung von Webpräsenzen drückt sich oft in langen Ladezeiten und fehlerhafter Darstellung der Website aus. Heutzutage sind Smartphones und Tablets fester Bestandteil des Alltags. Wenn eine Webpräsenz ihren Inhalt nicht an die Auflösung des Endgerätes anpassen kann, wird Navigation, Bedienung und Informationsgewinnung für die Nutzer sehr schwierig und wie schon im Punkt (2.2.2) erwähnt, werden Nutzer die Website einfach wechseln, wenn sie nicht zügig an die gewünschten Informationen kommen. Lange Ladezeiten sind ebenfalls ein Anzeichen für schlechte Optimierung einer Website. Dies hat verschiedene Ursachen. Hauptursache für lange Ladezeiten sind Bilder. Da unkomprimierte Bilder in voller Auflösung meist mehrere MB groß sind, sollten sie nicht direkt angezeigt werden. Ebenso sind multimediale Inhalte, wie Animationen und oder Sounds, ein häufiger Grund für einen langsamen Seitenaufbau und funktionieren zudem nicht in allen Browsern korrekt. Ein weniger einflussreicher Grund für verlängerte Ladezeiten sind unkomprimierte HTML- und CSS-Dokumente. Unnötige Whitespaces, also Leerzeichen oder Zeilenumbrüche, sowie lange Tag-Bezeichnungen vergrößern die Datenmenge der Dokumente, welche für den Seitenaufbau geladen werden müssen. Ebenfalls erhöht nicht nur die Größe, sondern auch die Anzahl der zu ladenden Dateien die Ladezeiten. Wenn z.B. viele einzelne CSS-Dokumente geladen werden müssen, erhöht sich die Anzahl der HTTP-Anfragen, welche sonst für andere Daten genutzt werden könnten.

2.3.4 Unnötiger Technikeinsatz

Werden Nutzer gleich zu Beginn ihres Besuches darauf hingewiesen, dass ein Plugin installiert oder aktualisiert werden muss, um die Webpräsenz nutzen zu können, stößt dies oft auf Unmut. Ein gutes Beispiel dafür ist der Flash-Player von Adobe. Große Browser-Anbieter wie Google und Mozilla haben in ihren jeweiligen Browsern bereits Flash-Inhalte standardmäßig deaktiviert, um das Surferlebnis sicherer und schneller zu gestalten [Vgl. Weniger Adobe Flash in Firefox 2016, LaForge 2016]. Stattdessen müssen Nutzer nun diese Inhalte manuell mit einem Klick erlauben, was wieder dazu führt, dass sie sich mit der Bedienung der Website beschäftigen müssen ehe sie die Inhalte zu Gesicht bekommen. Zudem birgt die Verwendung von Plugins auch ein erhöhtes Sicherheitsrisiko. Immer wieder werden Sicherheitslücken und unentdeckte Exploits, sogenannte Zero-Day-Exploits in Plugins und Komponenten wie dem Flash-Player und Java entdeckt. Sollten Nutzer sich auf der Webpräsenz mit Schadsoftware infizieren, oder diese Website aufgrund von Schadcode vom Browser gesperrt werden, führt das zu Nutzerverlust und schlimmstenfalls zu Haftungsrisiken [Vgl. Rohr 2015, S. 92-93].

3 Konzept

Dieser Abschnitt beschäftigt sich mit der Erarbeitung eines Konzeptes für die Webpräsenz, vom Wireframe bis hin zur Wahl der passenden Schriftart für die Website. Dabei wird gleichzeitig das Konzept für die im Rahmen der Bachelorarbeit entwickelte Webpräsenz erstellt.

3.1 Wireframe

> „A wireframe defines the page layout for a Web site, showing what content goes where." [Snyder 2003, S. 10]

Nachdem die Idee feststeht und klar ist, welche Zielgruppe angesprochen werden soll, ist der Wireframe der erste Schritt bei er Konzeption einer Webpräsenz. Dabei werden die feineren Details wie Farbe und Schrift außen vorgelassen und sich nur auf den Aufbau der späteren Webpräsenz konzentriert. Es werden also die Grundelemente wie Header, Footer, Seiteninhalt usw. am besten auf einem Grid, Raster oder Gitter auf deutsch, angeordnet. Das Grid hilft dabei, die Elemente anzuordnen und die Seite gleichmäßig aufzuteilen. Falls man ein CSS-Framework verwendet, ist es eine gute Idee, die Spaltenanzahl an die des Frameworks anzupassen. Der Wireframe kann dabei verschiedene Formen haben. Einige werden auf einem Blatt Papier gezeichnet andere werden mit entsprechender Software am PC, oder mit Hilfe von Web-Diensten erstellt. Egal für welche Form man sich entscheidet, zu Beginn sollte man sich einen der typischen Layout-Typen für eine Webpräsenz als Ausgangspunkt nehmen. Stefan Münz, ehem. Autor bei und ehem. Vereinsvorsitzender von SELFHTML [SELFHTML, heise online], beschreibt zum Beispiel drei Layout-Typen:

- **Portallayout** Das Portallayout ist den Triumphbögen der alten Römer nachempfunden. Zwei Säulen links und rechts auf denen ein prominenter Oberbau ruht. Dieses Layout strahlt Ruhe und Ausgewogenheit aus und präsentiert gleichzeitig eine Menge an Informationen. Dieses Layout wird oft bei Onlineauftritten von Zeitungen, oder großen Unternehmen und Einrichtungen verwendet.
- **Winkellayout** Das Winkellayout ist offener gestaltet als das Portallayout. Beim Winkellayout wird der Seiteninhalt links und oben von zwei Leisten eingeschränkt. Somit wird mehr Dynamik suggeriert und dem Seiteninhalt wird optisch mehr Platz zugesprochen. Dieses Layout ist universell für jede Website geeignet, bei der Funktion und Inhalt an erster Stelle stehen und Originalität erst danach kommt.

- **Individuelles Layout** Das individuelle ist das am freisten gestaltete Layout der hier vorgestellten Layouts. Hier sind die Bereiche einer Website meist unkonventioneller verteilt als beim Portal- oder Winkellayout, wobei dennoch der professionelle Charakter erhalten bleibt. Dieser Typ wird oft von kleineren Webpräsenzen verwendet, um dem Auftritt einen persönlichen Akzent zu geben.

[Vgl. Münz 2006, S. 231 - 254]

Ein gutes Beispiel für das Portallayout ist der Webauftritt der Hochschule Anhalt. Hier sind die Seitennavigation und die andere Themen jeweils links und rechts in den Säulen positioniert. Die globale Navigation, das Logo sowie die Suche werden im Kopfbereich dargestellt.

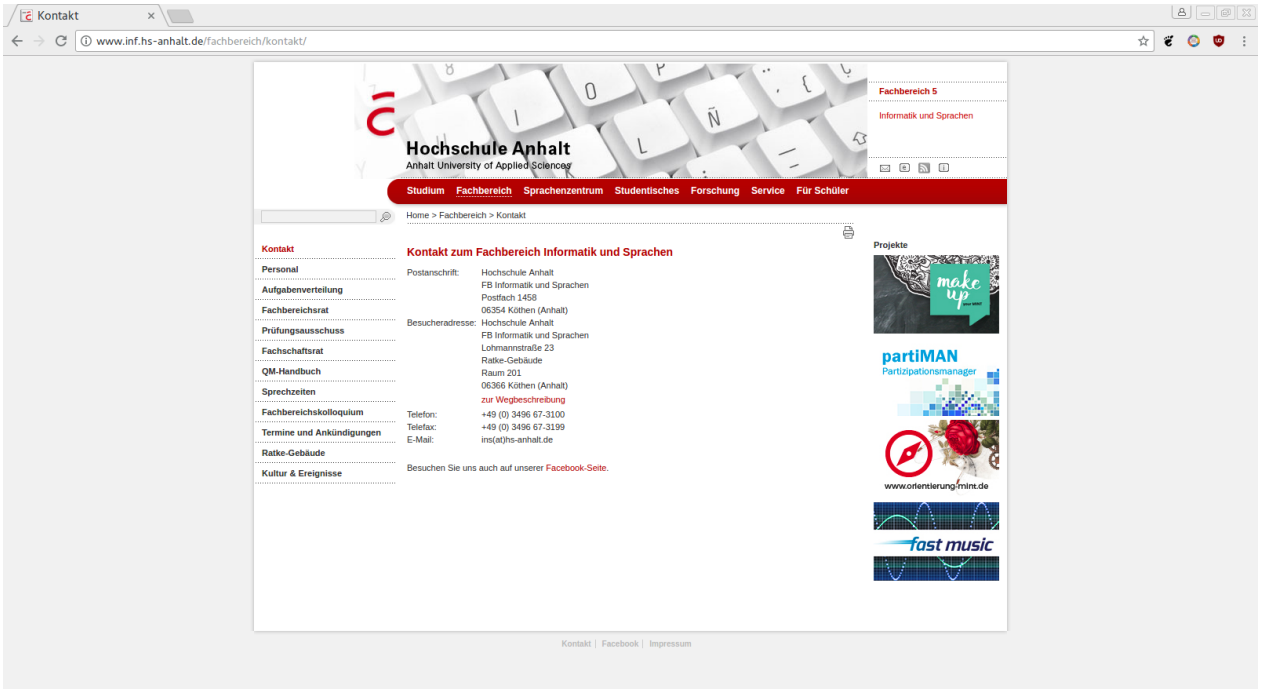

Abbildung 1: Portallayout (hier Hochschule Anhalt)

Ein Vertreter des Winkellayout findet sich mit der Onlineenzyklopädie Wikipedia. Mit dem Layout schafft es Wikipedia seine umfangreichen Informationen den Nutzern effizient zu präsentieren. In der linken Leiste stehen das Logo, die Navigation, die Seitenwerkzeuge und die Sprachwahl, während sich in der oberen Leiste die Suche und die Loginmöglichkeit befindet.

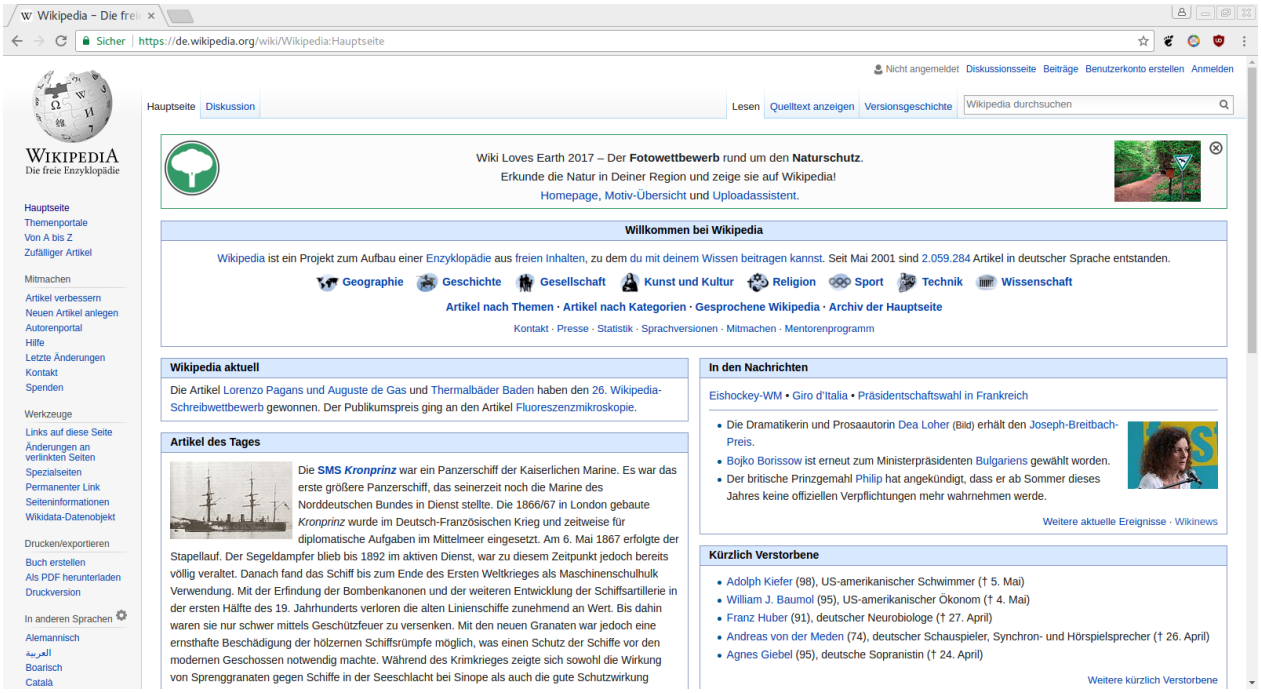

Abbildung 2: Winkellayout (Hier Wikipedia)

Durch Webseitenbaukästen und vorgefertigten HTML- und CSS-Vorlagen haben individuelle Layouts zwar etwas an ihrer Individualität verloren, aber an Beliebtheit gewonnen. In letzter Zeit verwenden auch mehr Onlineauftritte von großen Zeitungen individuelle Layouts. Dabei sind die Layouts meist so angelegt, dass sie auch gut auf kleineren Endgeräten skalieren. Logo, Navigation, Suche und Login werden im Kopfbereich gebündelt und der Seiteninhalt an die Breite der Kopfleiste angepasst. Zusätzlich wird soviel Inhalt wie möglich auf einer kontinuierlich nach unten scrollbaren Fläche präsentiert, um den Navigationsaufwand zu minimieren. Somit sind keine separaten Layouts für Computer und mobile Geräte nötig.

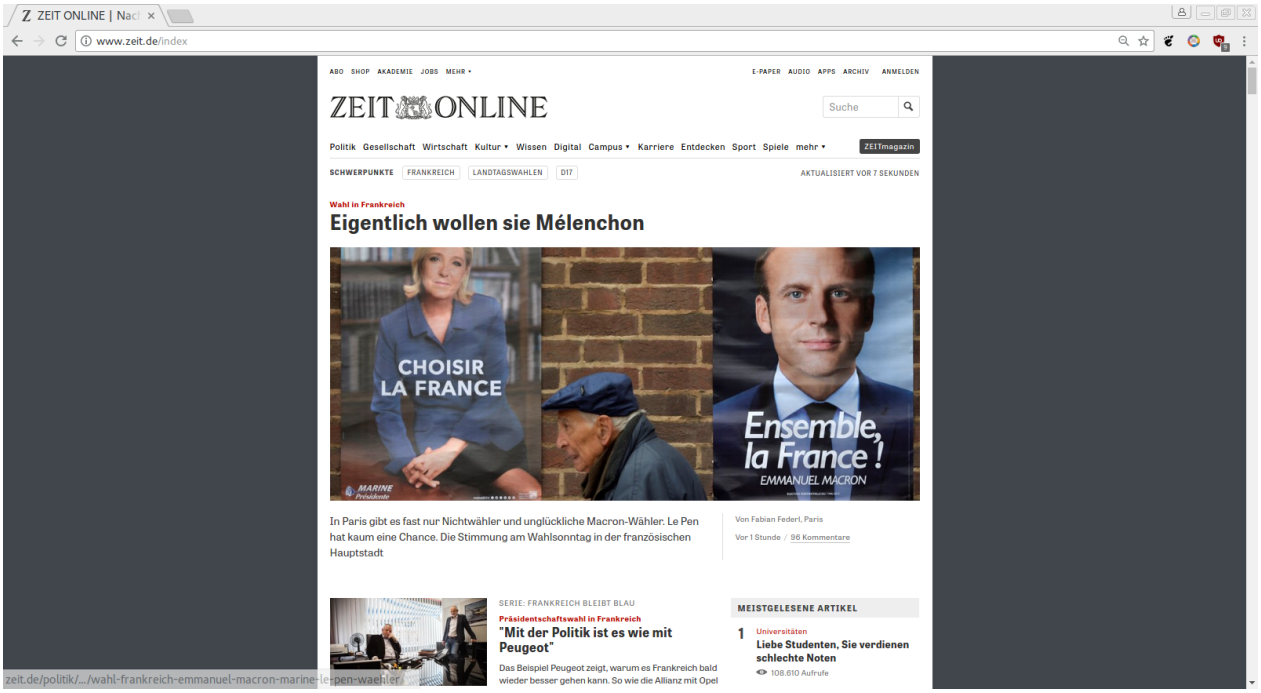

Abbildung 3: Individuelles Layout (Zeit Online)

So lässt sich also das Ausgangslayout für den Wireframe recht gut bestimmen. Soll die Website eine gewissen Seriosität ausstrahlen und oder eine konservativere Zielgruppe ansprechen, ist das Portallayout ein guter Ausgangspunkt. Das Winkellayout ist universal für jeden Website-Typ geeignet. Wenn das Hauptanliegen die Informationsvermittlung ist, oder eine sehr breite Nutzergruppe angesprochen werden soll, ist das Winkellayout eine gute Wahl. Das individuelle Layout wird von Seiten bevorzugt, die kreative Inhalte, wie Fotos, Videos oder Zeichnungen, präsentieren. Dort ist von vornherein schon eine große kreative Energie vorhanden, welche gleich in die Personalisierung der Webpräsenz einfließen kann. Zudem kann von den Wireframes anderer Seiten Inspiration gewonnen werden. Dabei helfen verschiedene Browsererweiterungen, welche die aktuelle Seite zu einem Wireframe abstrahieren, oder ganze Websites, welche Sammlungen von Wireframes verschiedener beliebter Webpräsenzen darbieten. Zudem kann es helfen, verschiedene Skizzen zu machen, bevor ein endgültiger Wireframe festgelegt wird.

Für die Webpräsenz im Rahmen der Bachelorarbeit wurde das Winkellayout ausgewählt, da sie hauptsächlich Informationen weitergeben soll. Zusätzlich ist aufgrund der Fülle an Text ein Layout, welches dem Seiteninhalt viel Platz lässt, geeigneter. In der oberen Leiste befinden sich dabei der Titel der Webpräsenz und eine globale Navigation mit Verlinkungen zu den verschiedenen Kapiteln der Arbeit, welche immer sichtbar sein wird. Somit können Nutzer jederzeit zwischen den Kapiteln wechseln. In der linken Leiste wird die lokale Navigation für das gerade geöffnete Thema sein, damit sich der Nutzer bequem im akutellen Kapitel zurechtfinden kann. Außerdem wird ein zusätzlicher Wireframe für das Layout auf mobilen Endgeräten erstellt, bei dem die Navigation mit Hilfe eines Buttons ausgeklappt werden kann.

Somit sehen die drei unterschiedlichen Wireframes für die Website folgendermaßen aus:

Abbildung 4: Wireframe Startpage

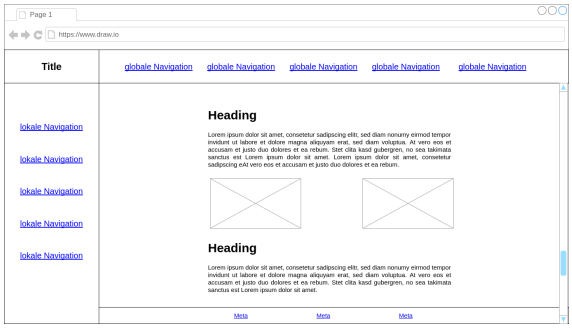

Abbildung 5: Wireframe Inhaltsseite

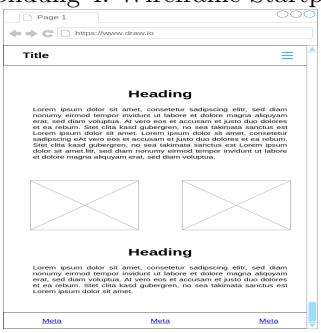

Abbildung 6: Wireframe Mobile

3.2 Navigation

Nachdem bei den Konzeptionsproblemen schon erarbeitet wurde, was man beim Navigationsdesign vermeiden sollte (2.2.1), lässt sich nun eine gute Navigation für die Webpräsenz entwerfen. Die Navigation sollte diese Punkte erfüllen:

- **Aussagekräftige Labels** Die Navigation sollte nur Begriffe beinhalten, die auch auf den ersten Blick vermitteln, was sich hinter ihnen verbirgt. Wenn Bilder verwendet werden sollen, dann nur in Verbindung mit Beschriftungen [Vgl. Hammer 2011, S. 128 u. 228; Thesmann 2016, S. 388ff.; Böhringer 2014, S. 72ff.].

- **Sinnvolle Reihenfolge** Thematisch zusammenhängende Navigationspunkte sollten nach Wichtigkeit von links nach rechts bzw. von oben nach unten, angeordnet werden, damit der Nutzer schnell zu den gesuchten Informationen kommt. Sogenannte Meta-Angaben, wie zum Beispiel das Impressum, sollte, wenn überhaupt, als einer der letzten Punkte in der Navigation sehr weit rechts oder an unterer Stelle stehen, da es für Nutzer meist wenig interessant ist [Kalbach 2008, S. 5; Stapelkamp 2007, S. 187].
- **Minimale Animation, Keine Versteckspiele** Die Navigation dient zur Bewegung über die Webpräsenz. Dies muss schnell und funktionell sein. Animationen beschränken sich am besten auf kleine Reaktionen bei Mausklicks oder Mouseover, um den Nutzer eine Rückmeldung zu geben. Alle Navigationspunkte sollten sofort sichtbar sein. Dropdown-Menüs werden am besten vermieden, oder erkennbar als solche markiert, wobei sie ihre Untermenüs dann bei Mouseover offenbaren. Falls die Webpräsenz allerdings an Kinder gerichtet ist, können die Animationen aufwändiger ausfallen und einige Navigationspunkte versteckter angelegt sein, um den kindlichen Forscherdrang zu befriedigen. Dabei sollte aber die Performance im Auge behalten werden [Böhringer 2014, S. 27; Nielsen 2006, S. 182].
- **Breadcrumbs** Neben der eigentlichen Navigation, gibt es noch die sogenannte Breadcrumb Navigation. Angelehnt an das Märchen der Gebrüder Grimm, in dem Hänsel und Gretel ihren Weg zurück nach Hause mithilfe von Brotkrumen finden, hilft die Breadcrumb Navigation dem Nutzer seinen bisherigen Weg auf der Website zurück zu verfolgen [Vgl. Thesmann 2016, S. 101; Hammer 2011, S. 178f.; Böhringer 2014, S. 74].

Von diesen Punkten abgesehen sollten noch verschiedene Konventionen beachtet werden, an die sich Internetnutzer gewöhnt haben. Zum Beispiel sollte ein Klick auf den Seitentitel oder das Logo zurück zur Startseite führen. Zudem ist es gut, zusätzlich einen extra Home-Button in der Navigation zu haben [Vgl. Krug 2014, S. 70]. Noch besser als am Ende der globalen Navigation machen sich die Verlinkungen zu Dingen wie dem Impressum, oder der „Über uns“-seite am Ende der Website, im sogenannten Footer. Dies sind zwar notwendige Meta-Angaben, aber nur in bestimmten Fällen interessant, somit sind solche Angaben im Footer gut aufgehoben [Vgl. Florin 2015, S. 175].

Für die Abschnitte und Unterabschnitte dieser Arbeit müssen also aussagekräftige Bezeichnungen gefunden werden, die im Rahmen der Webpräsenz für die globale und lokale Navigation verwendet werden können. Die Reihenfolge der Abschnitte und Unterabschnitte in der Arbeit muss ebenfalls angepasst werden. Abschnitt Eins wird als kurzer Begrüßungstext auf der Startseite realisiert und die Abschnitte 6 bis 9 nicht mit in die Navigation aufgenommen. Sie haben nur im Rahmen der schriftlichen Bachelorarbeit Relevanz bzw. werden die Quellen direkt auf der jeweiligen Webseite angegeben. Ebenfalls wird der Link zum Unterabschnitt 2.1 Geschichte von Webdesign und -entwicklung als Meta-Angabe in den Footer verschoben.

Die Inhalte der dritten Gliederungsebene werden nicht in die Navigation aufgenommen, sondern durch Überschriften auf den entsprechenden Themenseiten repräsentiert. Die Breadcrumb-Navigation nicht als separates Element erscheinen. Stattdessen wird sie durch die farbliche Hervorhebung des aktuellen Themas und Unterabschnitts in den globalen und lokalen Navigation realisiert. Das Impressum und die About-Seite befinden sich auch im Footer der Seite. Die Abschnitte 2 bis 5 können in ihrer Reihenfolge beibehalten werden. Es müssen also für die Titel dieser Abschnitte, für die Navigation geeignete Alternativen gefunden werden:

- **Abschnitt 2: Grundlagen** Da die Geschichte des Webdesigns und der Webentwicklung ausgelagert wurde, beschäftigt sich Abschnitt 2, im Rahmen der Webpräsenz, nur noch mit den konzeptionellen und technischen Problemen bei der Erstellung einer Website. Somit kann der Titel entsprechend angepasst werden in „Don'ts der Website-Erstellung". Die Unterabschnitte 2.2 und 2.3 können verkürzt werden auf „Konzeptionsprobleme" und technische Probleme".
- **Abschnitt 3: Konzept** Nur das Wort Konzept ist wenig aussagekräftig, auch wenn der Zweck der Website bekannt ist. Von daher wird für die Webpräsenz der Titel „Zuerst das Konzept" gewählt. Bei den Unterabschnitten werden folgende Titel verwendet: „Der Wireframe", „Responsive Webdesign", „Globale und lokale Navigation", „Der Seiteninhalt", „Seitenelemente: Positionierung/Ausrichtung", „Die richtige Farbwahl", „Welche Schriftart?".
- **Abschnitt 4: Umsetzung** Ebenfalls wenig aussagekräftig. Stattdessen „Dann die Umsetzung" und für die Unterabschnitte: „Die passende Software", „Inhalt und Struktur mit HTML", „Das gute Aussehen mit CSS".
- **Abschnitt 5: Evaluierung** Genau wie Abschnitt 3 und 4 muss hier ein deskriptiver Titel gefunden werden. Für die Webpräsenz wird der Titel „Testen: Funktioniert alles?" verwendet. Die Unterabschnitte werden mit „Funktionstests", „Oberflächentests", „Weitere Tests" betitelt.

3.3 Inhalt

Ebenso wichtig, oder vielleicht sogar noch wichtiger als das Aussehen der Website, ist der Inhalt. Er ist der Grund warum Nutzer die Webpräsenz überhaupt aufgerufen haben. Daher sollte direkt auf der Startseite eine kurze Zusammenfassung des Inhalts, oder das Thema der Webpräsenz zu lesen sein, damit Nutzer direkt wissen, was sie erwartet und auch, welche Informationen sie hier nicht finden werden.

Auf einer Website gelten genauso die Regeln der Typografie wie in gedruckten Medien [Vgl. Hammer 2011, S. 265]. Dabei werden Satzformen, Zeilenabstände, Zeichenabstände und Spaltenbreite dazu gebraucht, ein attraktives, angenehm zu lesendes Schriftbild zu liefern. Wenn z.B. längere Texte auf der Website zu finden sind, bietet sich für diese der Blocksatz (Abbildung 8) mit 60 bis 70 Zeichen je Zeile „einschließlich der Satzzeichen und Zwischenräume" an, da diese „die besten Voraussetzungen für ein ermüdungsfreies Lesen bieten" [Bollwage 2005, S. 44]. Kurze Texte wie Bildunterschriften, oder Kommentare und sollten dagegen im Flattersatz (Abbildung 7), in Textbearbeitungsprogrammen meist kurz als linksbündig bzw. rechtsbündig benannt, stehen. Sind auf der Website Texte mit feierlichem oder offiziellem Charakter zu finden kommt die symmetrische Satzform zum Einsatz. Beim symmetrischen Satz (Abbildung 9) richtet sich die Zeilenlänge eher nach der Sprechweise als nach dem Text.

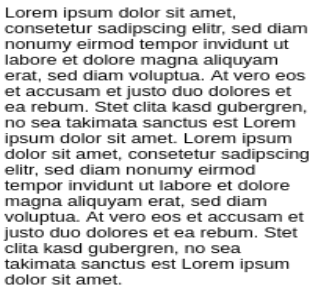

Abbildung 7: Flattersatz

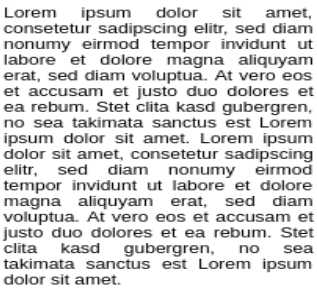

Abbildung 8: Blocksatz

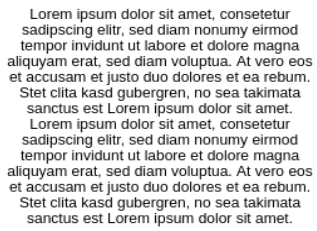

Abbildung 9: Symmetrischer Satz

Ist der Text in mehrere Spalten geteilt, muss darauf geachtet werden, dass ein passender Spaltenabstand gelassen wird. Da der Blocksatz auf beiden Seiten bündig ist, kann es passieren, dass der Nutzer beim Lesen in die nächste Spalte rutscht wenn der Abstand zwischen beiden Spalten geringer als der Abstand zwischen den einzelnen Worten ist. Bei Flattersätzen kann der Spaltenabstand geringer sein, da durch die unebenen Ränder der Abstand größer wirkt.

Der Zeilenabstand wird von den meisten Textbearbeitungsprogrammen automatisch auf einen Abstand festgelegt, der gut lesbar ist. Dort muss nur in den seltensten Fällen nachgebessert werden. Bei Überschriften die über mehr als eine Zeile gehen, kann der Zeilenabstand verringert oder sogar ganz weg gelassen werden. So entsteht ein größerer Zusammenhalt.

Bilder können auf Webpräsenzen auf vielerlei Art eingesetzt werden. Bilder können lange Texte auflockern, oder textarme Seiten auffüllen. Bilder können Sachverhalte erklären, die sonst umständlich mit Worten erklärt werden müssten.

Bilder können auch der Hauptinhalt einer Seite sein, wenn zum Beispiel ein Fotostudio seine Arbeit bewerben möchte und der Text als Bildunterschrift nur eine Nebenrolle spielt. Allgemein gesagt sind Bilder ein mächtiges gestalterisches Werkzeug. Sie können mit ihren Motiven oder Perspektiven den Blick leiten. Treppen können z.B. den Blick auf andere Inhalte lenken oder die Aufmerksamkeit auf der Seite halten [Vgl. Hammer 2011, S. 271]. Wenn der Webseiteninhalt hauptsächlich aus Bildern besteht, können auch diese verschieden angeordnet werden, um einen weniger eintönigen Eindruck zu vermitteln. Zum Beispiel können mehrere kleine Bilder im Hochformat nebeneinander gestellt und von einem großen Bild im Querformat untermauert werden. Hat man verschiedene Kategorien auf einer Seite, kann das erste Bild in der Kategorie größer dargestellt werden als der Rest, damit die Abtrennung durch Bild und Überschrift erfolgt.

Desweiteren sollte noch erwähnt werden, dass bei Websites zwischen dynamischen und statischen Inhalten unterschieden wird. Bei statischen Websites steht der Inhalt in einer HTML-Datei, welche direkt vom Server abgerufen wird und ändert sich dementsprechend nur, wenn man ihn direkt in der Datei ändert. Somit stehen die Inhalte, welche auf der Website zu sehen sind fest, sind also statisch. Bei dynamischem Inhalt hingegen wird der Inhalt auf der Serverseite erst anhand entsprechender Kriterien mit Hilfe von Skripten generiert und danach an den Browser des Nutzers weitergeleitet. Bei dieser generierten Datei muss es sich nicht immer um die typischen .html Dateien handeln. Das bekannteste Beispiel für eine Website mit dynamischen Inhalt ist die Websuche von Google. Dort werden je nach Suchanfrage die entsprechenden Inhalte für die Website generiert. Um solche Inhalte zu generieren, „kommen typischerweise serverseitige Programmiersprachen wie PHP, JSP oder Perl zum Einsatz. Nicht selten werden auszugebende Inhalte dabei aus einer ebenfalls auf dem Server laufenden Datenbank geholt“ [Münz 2006, S. 21]. Insbesondere das in den 90iger Jahren speziell für den Einsatz im Web entwickelte PHP ist schon länger weit verbreitet [Vgl. Walter 2008]. Da das Generieren dynamischer Inhalte schon fortgeschrittenere Programmierkenntnisse erfordert und für die meisten Laien eher uninteressant ist, wird dieses Thema im Rahmen der Bachelorarbeit nicht weiter behandelt.

Für die Bachelorwebsite wird aufgrund des vielen Texts ein einspaltiger Blocksatz mit 60-70 Zeichen pro Zeile und einem 1.3-fachen Zeilenabstand verwendet. Bei Überschriften wird der Abstand zum darauf folgenden Text auf die Hälfte des normalen Zeilenabstands verringert. Der Text wird von Bildern unterbrochen, die dazu verwendet werden, vorher beschriebene Sachstände weiter zu erklären.

3.4 Positionierung und Ausrichtung

Die Positionierung und Ausrichtung von Elementen auf der Website wird mit Hilfe der HTML-Eigenschaften position, float oder shape vorgenommen.

position Die Eigenschaft position hilft dabei Elemente auf der Seite zu positionieren. Dabei kann mit den Richtungsangaben top, bottom, left, right festgelegt werden wo der obere, untere, linke oder rechte Rand des Elements beginnt. Ebenfalls kann man mit margin und padding der äußere bzw. der innere Abstand definiert werden. position hat dabei folgende Spezifikationen:

- **position:fixed** dient dazu, das Element vom restlichen Inhalt loszulösen und an einer Position zu fixieren. Dadurch ist es immer an dieser Position sichtbar und wird nicht durch Scrollen beeinflusst. Dies macht die Nutzung von Frames überflüssig.
- **position:absolute** richtet ein Element beliebig auf der Website aus. Das Element wird vom Inhalt losgelöst und als Referenzpunkt wird das letzte Elternelement gewählt, welches ebenfalls mit der Eigenschaft position: positioniert wurde. Existiert dieses nicht, wird auf das nächste Wurzelelement zurückgegriffen.
- **position:relative** lässt das Element eine Position relativ zu seiner Standardposition im Seiteninhalt annehmen. Anders als bei fixed und absolute wird das Element dabei nicht vom restlichen Inhalt losgelöst und es entsteht eine Lücke, sollte man das Element mit top, bottom, left, right von seiner Ausgangsposition verschieben.
- **position:static** bezeichnet die Standardangaben für position. Angaben zur Positionierung werden dabei ignoriert.

float Mit float wird ein Block definiert, welcher vom restlichen Inhalt losgelöst wird und an den rechten bzw. linken inneren Rand des Elternelements positioniert wird. Dieser Block wird dann vom restlichen Inhalt umflossen. Zusätzlich kann mit dem dazugehörigen Befehl clear bestimmt werden, ob der folgende Inhalt den float-Block tatsächlich umfließt oder einen neuen Abschnitt unter dem Block beginnt.

shape shape wird verwendet wenn der Textfluss an die Kontur einer Form oder eines Bildes angepasst werden soll. Dabei wird vorausgesetzt, dass die Form oder das Bild in einem float-Block stehen. Zudem muss für die Form eine feste Breite und Höhe angegeben werden, bei Bildern ist dies nicht nötig, da diese schon eine feste Höhe und Breite besitzen. Mit shape-outside wird dabei festgelegt, was für ein Objekt der Text umfließen soll:

- **inset()** für Rechtecke
- **circle()** für Kreise
- **ellipse()** für ovale Formen
- **polygon()** für abstraktere Formen mit vielen Ecken.
- **url()** für Bilder. Diese müssen dazu auf dem gleichen Webserver wie die Website liegen.

Weiterhin können sogenannte Grid-Systeme genutzt werden, um flexibeles Webdesign einfacher zu ermöglichen. Bei einem Grid-System wird das Layout der Webpräsenz in Zeilen und Spalten unterteilt, welche auf ausreichend großen Bildschirmen nebeneinander und bei kleineren Bildschirmen untereinander dargestellt werden. So können zum Beispiel Navigation und Inhalt in mehrere Spalten unterteilt werden. Hat der Bildschirm eine ausreichende

Größe wird die globale Navigation oben dargestellt, die lokale Navigation am linken Rand und der Seiteninhalt in der Mitte. Soll die Website nun auf einem kleineren Bildschirm dargestellt werden, falten sich die Spalten zusammen und die lokale Navigation erscheint nun unterhalb der globalen und oberhalb des Seiteninhalts.

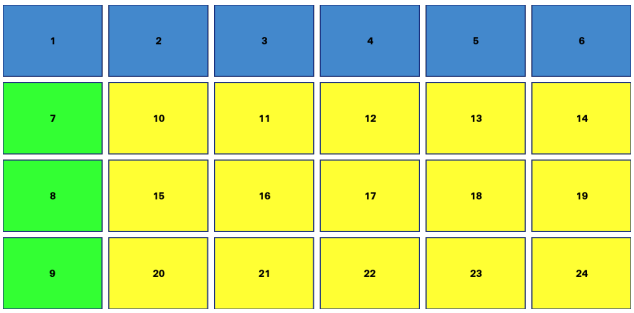

Abbildung 10: Grosse Bildschirme

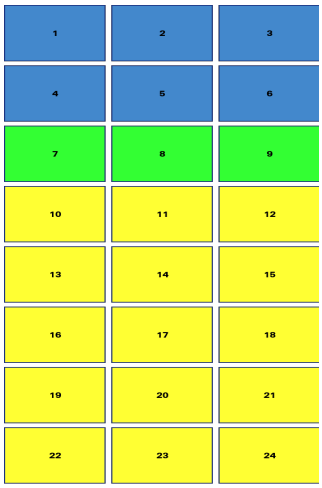

Abbildung 11: Kleine Bildschirme

3.5 Farben und Kontrast

Farben machen eine Website lebendig. Sie können dem Nutzer bei der Navigation helfen, ihn auf wichtige Inhalte aufmerksam machen und emotional beeinflussen. Für die Gestaltung von Websites werden Farbkreise verwendet, welche sich „an der Farbwahrnehmung des Menschen (Farbton, Farbsättigung, Farbhelligkeit) orientieren“ [Thesmann 2016, S. 301]. Insofern kein Styleguide, d.h. Angaben bezüglich zu verwendender Farben und Schriftarten, vorgegeben ist, wird im ersten Schritt bei der Farbgestaltung einer Webpräsenz eine Primär- bzw. Basisfarbe festgelegt. Diese Farbe wird auf der Webpräsenz dominant sein und bestimmen, welche Sekundärfarben genutzt werden. Die Basisfarbe sollte dabei das Thema der Website und dessen Grundaussage reflektieren. Dabei sollte sich vorher über die physiologische und emotionale Wirkung von Farben informiert werden. Rot wirkt zum Beispiel aufregend und wärmend, kann aber auch aggressiv machen, lässt Nutzer also „rot sehen“. Blau hingegen wirkt ruhig und harmonisierend und sorgt für Entspannung. Die genauen Wirkungen von Farben auf den Menschen ist ein sehr komplexes Thema und würde den Rahmen dieser Arbeit sprengen. Stattdessen kann sich auf der Seite Küppers' Farbenlehre mit der gewünschten Intensität mit der Farbenlehre beschäftigt werden. Dies ist die offizielle Seite von Harald Küppers, dem Erschaffer des aktuellsten Farbkreises und beschäftigt sich mit dessen Farbenlehre und bietet kostenlose Downloads zu zwei seiner Bücher an.

Hat man sich auf eine passende Basisfarbe festgelegt, werden abhängig davon 2-5 Sekundärfarben gewählt, welche über die gesamte Webpräsenz konsistent bleiben. Dabei gibt es zwei Möglichkeiten die Farben auszuwählen.

- **Benachbarte Farben** Soll eine ruhige und ausgeglichene Wirkung erzielt werden, sind benachbarte Farben im Farbkreis geeignet. Bei einer gelben Basisfarbe würde man als Sekundärfarben also orangene und rötliche Farben wählen, da diese im Uhrzeigersinn an die Basisfarbe angrenzen. Sättigung und Helligkeit sollte dabei gleich sein.
- **Komplementärfarben** Soll die Webpräsenz hingegen sehr lebendig wirken, entscheidet man sich für Komplementärfarben. Dabei wird als Sekundärfarbe die Farbe gewählt, welche der Basisfarbe direkt gegenüberliegt um einen hohen Kontrast zu erzielen. Damit der Kontrast dabei nicht zu aufdringlich ist, wird dabei oft lieber auf die beidseitigen Nachbarn der jeweiligen Komplementärfarbe zurückgegriffen.

[Vgl. Thesmann 2016, S. 317ff.]

Wurde ein Farbschema zusammengestellt, kann nun damit begonnen werden, die Elemente der Seite entsprechend zu kolorieren. Zum Beispiel kann die Basisfarbe als Hintergrundfarbe im Header der Website verwendet werden, um sie als dominante Farbe zu etablieren. Name der Website und Elemente der Navigation können dann in den Sekundärfarben gehalten werden, um den nötigen Kontrast zu schaffen. Dabei sollte darauf geachtet werden, das die Sättigung und Helligkeit der dort verwendeten Farben eher gering sind, um nicht vom Inhalt der Website abzulenken. Allgemein sollte die Grundstimmung der Website ausgeglichen sein. Sollen dann einzelne wichtige Elemente besonders hervor gehoben werden, kommen gesättigte helle Farben zum Einsatz, die sich deutlich abheben und Aufmerksamkeit auf sich ziehen. Für den Hintergrund der Website wird meist eine weißliche Farbe verwendet, um eine reine, ordentliche Grundfläche zu bieten. Ein reines Weiß ist als Hintergrund eher nicht geeignet, da der Bildschirm dadurch zu grell strahlt. Natürlich können auch schwarz oder sehr dunkle Farbtöne und sogar Bilder als Hintergrund verwendet werden, dies ist aber eher selten. Im Internet gibt es zahlreiche Websites, welche dabei helfen das passende Farbschema zu finden. So hilft die Website www.paletton.com zum Beispiel dabei zu einer Basisfarbe die passenden Sekundärfarben zu finden und bietet auch gleich Beispiele, wie diese im Rahmen der Webpräsenz eingesetzt werden können. Dabei kann zwischen Farbschemas mit benachbarten Farben und geteilten oder doppelt geteilten Komplementärfarben gewählt werden. Zudem können visuelle Darstellungen für Personen mit Farbfehlsichtigkeit simuliert werden.

Um die Informationen einer Website gut lesen zu können, ist natürlich auch der richtige Kontrast zwischen Text und Hintergrund nötig. Allgemein gilt, für kleine Schriftgrößen muss der Kontrast mindestens 4,5:1 betragen, für Großschriften reicht ein Kontrast ab 3:1 [vgl. Thesmann 2016, S. 80].

Wie schon erwähnt, sollte auch auf Menschen mit Farbfehlsichtigkeit, also Rot-Grünschwäche usw., geachtet werden. Das heißt, dass der eben genannte Kontrast eingehalten werden sollte und wichtige Informationen nicht nur mit Hilfe von Farben kodiert sein sollten.

Für die Bachelorwebsite wird ein weißlicher Hintergrund und eine blaue Basisfarbe gewählt. Diese wird mit Ruhe und Sachlichkeit assoziiert und fördert die Kommunikation. Mit der schon erwähnten Website www.paletton.com wird davon ausgehend folgendes Farbschema mit benachbarten Farben generiert.

Abbildung 12: Farbschema Bachelor Website

Die verschiedenen Helligkeiten werden dabei für Hintergründe aktiver bzw. inaktiver Navigationselemente und als Markierung für bereits besuchte Verlinkungen verwendet. Zusätzlich wurde dieses Farbschema mit Hilfe der Website auf gute Lesbarkeit für farbfehlsichtige Menschen geprüft.

3.6 Schriftarten

Welche die richtige Schriftart für die Webpräsenz ist, hängt von verschiedenen Faktoren ab. Am Anfang sollte entschieden werden, ob auf dem Nutzersystem installierte Schriftarten verwendet werden sollen, oder ob sogenannte Webfonts von anderen Seiten eingebunden werden sollen. Die Auswahl ist bei lokal installierten Schriftarten geringer, allerdings sorgt das Einbinden von Webfonts zu zusätzlichen Datenübertragungen und verlangsamt so den Seitenaufbau. Wurde eine Möglichkeit ausgewählt, kann nun die passende Schriftart für den Seiteninhalt gewählt werden. Besteht der Inhalt der Website hauptsächlich aus längeren Texten sollte ein serifenloser Schrifttyp, wie Verdana oder Open Sans gewählt werden, denn diese „sind, mit Ausnahme der schmalen Fonts, besser zu lesen" [Bollwage 2005, S. 62]. Noch besser ist es, wenn die Schriftart zusätzlich für die Darstellung von Texten im Internet entwickelt wurden, wie dies bei den eben genannten Schriftarten der Fall ist. Am Besten ist es verschiedene Schriftarten zu probieren. Natürlich können einige Schriftarten schon durch den Inhalt der Website ausgeschlossen werden. Der Blog eines Anwalts sollte nicht in Comic Sans verfasst werden und Anleitungen auf einer Bastelseite für Kinder muss nicht in Frakturschrift geschrieben sein. Zudem sollte die gewählte Schriftart und -größe an verschiedenen Orten getestet werden. Anhand des längsten Navigationslinks kann getestet werden, ob der Link in dieser Schriftart genug Platz hat. Wird der Einsatz von Serifen-Schriften gewünscht, sollten diese nur für große Texte genutzt werden. Da Serifen-Schriften im Gegensatz zu den serifenlosen Schriften meist dünner sind, werden diese in kleinen Schriftgraden und bei geringen Auflösungen oft schwer leserlich. Zudem werden die Serifen bei hellen Hintergründen vom Licht, welches vom Monitor ausgeht, überstrahlt. Dies ist der Lesbarkeit ebenfalls abträglich. Ein guter Kompromiss sind daher serifenlose Schriften, welche die eleganten, schwungvollen Formen von Serifen-Schriften imitieren. Ein Beispiel für eine sogenannte Semi-Serifen-Schrift ist Rotis® Semi Serif von Otl Aicher.

Wie schon bei den konzeptionellen Fehlern (2.2) erwähnt, sollten auf der Webpräsenz nicht mehr als zwei unterschiedliche Schriftarten verwendet werden. Stattdessen kann man den Inhalt mit unterschiedlichen Schriftschnitten auflockern. Die bekanntesten Schriftschnitte sind dabei die Schriftstärke fett und die Schriftlage kursiv. Bei den Schriftstärken gibt es noch mager und normal und bei den Schriftlagen normal und oblique. Zusätzlich gibt es noch die Schriftbreite, welche die Buchstaben enger zusammen oder weiter auseinander rückt. Mit diesen Mitteln kann man auch ohne viele Schriftarten und auffällige Farben wichtige Informationen hervorheben, indem sie z.B. mit einer fetten Schriftstärke geschrieben werden. Allerdings sollten diese Schriftschnitte sparsam eingesetzt werden, da sie sonst ihr Alleinstellungsmerkmal verlieren und schnell überlesen werden. Außerdem ist zu beachten, dass das Unterstreichen nur Verlinkungen vorbehalten sein sollten, da sich dies als Konvention etabliert hat [Vgl. Hammer 2011, S. 229].

Die Verwendung von sollte Schriftgrafiken weitestgehend vermieden werden. Schriftgrafiken sind Texte, welche in Bildbearbeitungsprogrammen erstellt wurden und als Bilder in die Website eingebunden werden. Aus diesem Grund können sie nicht von Suchmaschinen oder Screenreadern gelesen werden. Einer der wenigen Orte an denen Schriftgrafiken verwendet werden können, ist Logos, wenn sichergestellt werden soll, das die Schriftart dort immer auf die selbe Art und Weise dargestellt wird. Dies ist nötig, wenn zum Beispiel Anforderungen des Styleguides eingehalten werden müssen.

Zum Schluss ist zu erwähnen, dass es mit CSS möglich ist, mehrere Schriftarten zu definieren, um eine Ausweichmöglichkeit zu haben, sollte die bevorzugte Schriftart nicht verfügbar sein. Es ist also keine schlechte Idee, ein bis zwei Alternativschriftarten bereitzuhalten.

Für die Website im Rahmen der Bachelorarbeit wird der von Google entwickelte und bereitgestellt Webfont Open Sans in die Website eingebunden. Dieser wurde direkt für die Darstellung von Texten im Web entwickelt und hat einen neutralen Charakter. Als Ausweichmöglichkeit wird auf die Schriftart Verdana zurückgegriffen, da diese auf den meisten Systemen lokal installiert ist.

4 Umsetzung

Nachdem nun ein Konzept erarbeitet wurde und bekannt ist, welche Farben, Inhalte und Schriften genutzt werden sollen, kann die Website nun mit Hilfe dieser Richtlinien umgesetzt werden. Dieser Abschnitt wird kurz den verwendeten Code-Editor vorstellen und beschreiben wie man das Konzept mittels HTML und CSS realisieren kann.

4.1 Code Editor Brackets

Bei der Wahl der richtigen Software kommt es darauf an, welche Funktionen persönlich als wichtig für eine produktive Arbeitsumgebung empfunden werden. Selbst ein simpler Texteditor wie Notepad würde sich für die Entwicklung eignen, wäre aber äußert umständlich. Daher gibt es Editoren, welche sich speziell an Webentwickler richten. Für die Entwicklung der Website im Rahmen der Bachelorarbeit wurde der Open Source Editor Brackets von Adobe gewählt. Neben dem Open Source Charakter spricht die Live Vorschau des Projekts im Browser, die Syntaxhervorhebung, die visuelle Interpretation von Farbcodes, Autovervollständigung und die Möglichkeit Erweiterungen zu installieren, für diesen Editor.

Adobe Brackets wird seit 2012 von Adobe als Communityprojekt unter der MIT-Lizenz entwickelt und ist freie Software (Adobe Brackets). Weitere beliebte Editoren sind zum Beispiel Atom (Atom) oder Sublime Text (Sublime Text).

4.2 Inhalt mit HTML

Bei einer Website wird die Struktur und der Inhalt in einer oder mehreren HTML-Dateien festgelegt. Es wird noch nichts formatiert oder gestaltet. Am Anfang der Datei wird der Doctype festgelegt. Der Doctype teilt mit, in welcher Version von HTML das Dokument verfasst ist. <!DOCTYPE HTML> beschreibt dabei ein HTML5 Dokument. Die Verwendung dieses Doctypes wird empfohlen, da alle Browser HTML5 Dokumente verstehen und Funktionen aus früheren HTML-Versionen trotzdem verwendet werden können. Danach folgt der eigentliche HTML-Code welcher durch sogenannte „Tags“ definiert wird. Diese Tags werden in spitze Klammern <Tag> eingefasst und mit </Tag> abgeschlossen. Wird der Tag in der selben Zeile abgeschlossen reicht ein </>. Der Tag <html> dient als Wurzelelement und umschließt den gesamten restlichen Inhalt. Danach ist das Dokument in zwei Abschnitte unterteilt:

- <head> - Beinhaltet Informationen über den Inhalt des Dokuments, wie Titel, Beschreibung, Keywords, verwendeter Zeichensatz, Pfad zum CSS-Dokument.
- <body> - Beinhaltet den sichtbaren Teil der Website, wie Navigation, Seiteninhalt, Kopfbereich bzw. <header> und Fußbereich bzw. <footer>.

Dabei ist zu beachten, dass <head> und <header> unterschiedliche Dinge sind. Eine frische HTML-Datei sieht also folgendermaßen aus:

```
1 <!DOCTYPE html>
2 <html lang="de"> <!-- Legt die Sprache des Dokumentes fest. -->
3
4    <!-- Pflichtelement. Hier werden die allgemeinen Angaben zum HTML-Dokument wie Titel,
         Zeichenkodierung, Beschreibung, Verwendetes CSS, Viewport etc. festgelegt. Ist nicht fuer
         den Nutzer sichtbar.-->
5     <head>
6       <meta charset="utf-8">
```

```
7         <title>Titel der Webpraesenz</title>
8         <meta name="description" content="Beschreibung der Website">
9         <meta name="keywords" content="Beispiel, HTML, Kopfbereich, Schluesselwort">
10    </head>
11
12 <!-- Pflichtelement. Beinhaltet den sichtbaren Teil der Website. Kopfbereich, Seiteninhalt,
       Fussbereich und Alles, was der Nutzer sehen soll befindet sich hier.-->
13    <body>
14        <header>
15            <p>Hier ist der header</p>
16        </header>
17        <main>
18            <h1>Inhalt</h1>
19            <p>Hier ist der Seiteninhalt zu sehen</p>
20        </main>
21        <footer>
22           <p>Hier ist der footer</p>
23        </footer>
24    </body>
25 </html>
```

Listing 1: Minimales HTML

Abbildung 13: Darstellung des minimalen HTML im Browser

Der Inhalt des HTML-Dokumentes wird dabei von oben nach unten abgearbeitet. Aus diesem Grund ist der <body> zusätzlich eingeteilt in <header>, <main>und <footer>. Im <header> stehen üblicherweise der Seitentitel und das Seitenlogo und Elemente, die beim Besuch der Website als erstes aufgebaut werden sollten. Im <footer> stehen Meta-Elemente wie das Impressum. Da der <footer>, wie der Name schon sagt, am Fuß der Seite steht, müssen Elemente, die dort stehen, nicht sofort geladen werden. Zwischen diesen beiden Bereichen wird in <main> dann der eigentliche Seiteninhalt festgelegt.

Im <main> kann der Inhalt dann zusätzlich als <article> definiert werden, welches einen abgeschlossenen Inhaltsbereich darstellt. <article> kann wiederum durch <section> in einzelne Abschnitte geteilt werden. Auch in diesen Unterteilungen kann jeweils das <header>-Attribut verwendet werden, um einen eigenen Kopfbereich zu markieren.

Der darzustellende Text an sich kann mit Hilfe weiterer Tags strukturiert werden. Mit <h1> bis <h6> können Überschriften und Unterüberschriften in 6 Abstufungen definiert werden. Textabsätze werden meist mit dem <p>-Tag umschlossen. Mit dem <figure>-Tag lassen sich Grafiken, aber auch Videos, Programmcode oder Zitate einfügen. Soll zusätzlich eine Bildunterschrift hinzufügt werden, kann dies innerhalb des <figure>-Tags mit <figcaption> verwirklicht werden. Aufzählungen und Listen lassen sich je nach Typ mit <ol> für geordnete Listen, <ul> für ungeordnete Listen oder für einen Block, der die gesamte Seitenbreite einnimmt definieren. Die einzelnen Einträge innerhalb der Listen lassen sich mit <li> definieren. Ein Beispieldokument mit diesen Elementen würde also folgendermaßen aussehen:

```
<!DOCTYPE html>
<html lang="de"> <!-- Legt die Sprache des Dokumentes fest -->
    <!-- Pflichtfeld. Hier werden die allgemeinen Angaben zum HTML-Dokument wie Titel,
        Zeichenkodierung, Beschreibung, Verwendetes CSS, Viewport etc. festgelegt. Ist nicht fuer
        den Nutzer sichtbar.-->
  <head>
    <meta charset="utf-8">
    <title>Titel der Webpraesenz</title>
    <meta name="description" content="Beschreibung des Inhalts">
    <meta name="keywords" content="Beispiel, HTML, Kopfbereich">
  </head>

    <!--Pflichtelement. Beinhaltet den sichtbaren Teil der Website. Kopfbereich, Seiteninhalt,
       Fussbereich und Alles, was der Nutzer sehen soll befindet sich hier.-->
  <body>
    <header>
      <p>Hier ist der header</p>
    </header>
       <!-- Die Navigation wird mit Hilfe einer Liste realisiert. Die einzelnen Listenpunkte <li>
           umschliessen dabei den Pfad zum referenzierten HTML-Dokument in Form eines Hyperlinks <a
           > und den Text, der anstelle des Hyperlinks auf der Website erscheinen wird. -->
    <nav>
      <ul>
        <li><a href="Erster/Link.html">Erste Seite</a></li>
        <li><a href="Zweiter/Link.html">Zweite Seite</a></li>
        <li><a href="Ditter/Link.html">Dritte Seite</a></li>
        <li><a href="Vierter/Link.html">Vierte Seite</a></li>
      </ul>
    </nav>
    <main>
      <h1>Ueberschrift 1. Ebene</h1>
      <p>Hier ist der Seiteninhalt zu sehen</p>
      <h2>Ueberschrift 2. Ebene</h2>
      <p>Noch mehr Seiteninhalt</p>

      <h1>Ein Bild</h1>
           <!-- Kann den Pfad zu einem Bild, Video, Programmcode, Diagramme oder aehnlichen
               grafischen Darstellungen enthalten. Mit figcaption wird der Untertitel festgelegt.
               Dateien im gleichen Ordner wie die HTML-Datei koennen mit Dateiname.endung
               referenziert werden. Dateien in anderen Ordnern brauchen die genauen Pfadangaben. --
               >
      <figure role="img">
        <img src="beispielbild.png">
        <figcaption>Dies ist eine Bildunterschrift</figcaption>
      </figure>
           <!-- Symbolisiert einen in sich abgeschlossenen Teil der Website, aehnlich einem Artikel
               in der Zeitung. Kann wiederum in mehrere sections aufgeteilt sein und durch header
               und footer umschlossen sein.-->
      <article>
```

```
39          <header>
40            <h1>Ein Artikel</h1>
41          </header>
42          <section>
43          <h2>Eine geordnete Liste</h2>
44          <ol> <!-- Geordnete Liste -->
45            <li>Erster Eintrag</li>
46            <li>Zweiter Eintrag</li>
47            <li>Dritter Eintrag</li>
48          </ol>
49
50          <h2>Eine ungeordnete Liste</h2>
51            <ul><!-- Ungeordnete Liste -->
52              <li>Erster Eintrag</li>
53              <li>Zweiter Eintrag</li>
54              <li>Dritter Eintrag</li>
55            </ul>
56          </section>
57          <footer>
58            <h1>Quellenangabe</h1>
59            <div>
60              <li>Erste Quelle</li>
61              <li>Zweite Quelle</li>
62              <li>Dritte Quelle</li>
63            </div>
64          </footer>
65        </article>
66      </main>
67
68      <footer>
69        <p>Hier ist der Footer</p>
70      </footer>
71    </body>
72  </html>
```

Listing 2: Grundlegende Elemente eines HTML-Dokuments

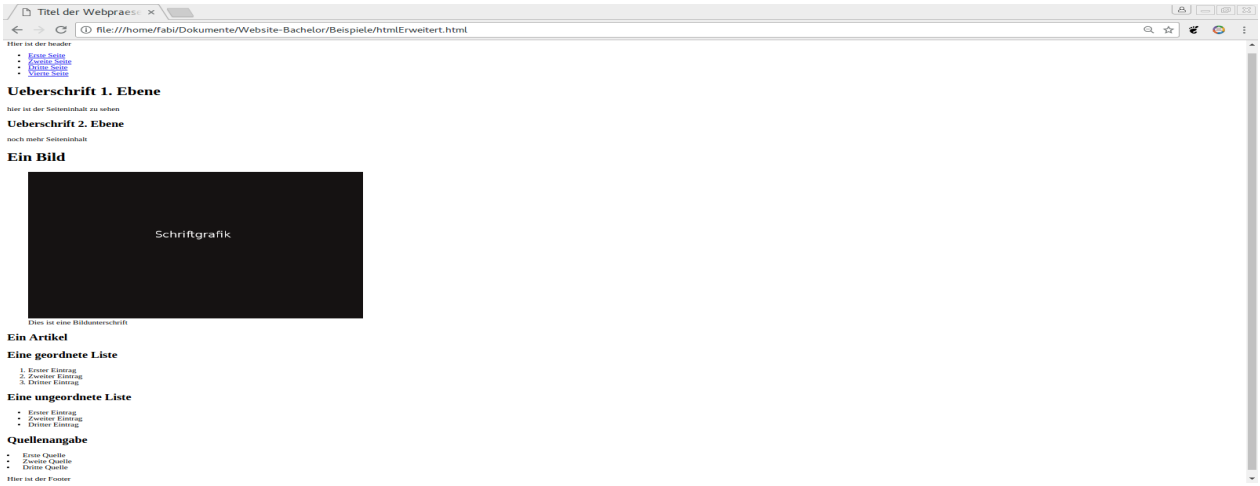

Abbildung 14: Darstellung der Elemente im Browser

Mit dem <nav>-Tag wird die globale Navigation in Form einer ungeordneten Liste <ul>, definiert. Mit <li> werden die einzelnen Listenpunkte festgelegt und mit Navigationstitel wird zu erst der Pfad zum verlinkten HTML-Dokument angegeben und danach, mit welchem Namen es auf der Webpräsenz erscheinen soll. Durch diese Angabe erscheint auf der Website ein Link, welcher die Nutzer zum hinterlegten HTML-Dokument weiterleitet. Dieser Link wird automatisch blau und unterstrichen im Browser dargestellt und ändert die Farbe, wenn er einmal besucht wurde. Bei der Pfadangabe für die gewünschte Datei im <figure>-Tag reicht der vollständige Name mit Dateiendung aus, insofern sich die Datei im gleichen Ordner wie das HTML-Dokument befindet. Ist sie in einem Unterordner abgespeichert, muss der Pfad ab dem Wurzelverzeichnis angegeben werden. Liegt die Datei ganz woanders, muss der vollständige Pfad mit Ordnerstruktur bzw. URL angegeben werden.

Soll die Website auch für blinde oder schwer sehbehinderte Nutzer geeignet sein, sollten den HTML-Elementen noch zusätzlich sogenannte ARIA-Rollen zugeordnet werden. ARIA steht für „Accessible Rich Internet Applications“, eine Initiative des W3C, welche die Zugänglichkeit zum Internet, insbesondere für blinde Nutzer, erhöhen soll. Damit diese Nutzer über die Website navigieren können, werden ARIA-Rollen als Orientierungspunkte, auch Landmarks genannt, festgelegt. Durch diese Landmarks wissen Screenreader, welche Informationen in den einzelnen Elementen zu erwarten sind. Zudem können mit zusätzlichen ARIA-Rollen wie aria-current, aria-busy, aria-autocomplete das aktuell fokussierte Element, der Status des Elements oder zusätzliche Funktionen kommuniziert werden. Eine vollständige Liste von Inhalten und ihren empfohlenen ARIA-Rollen haben Mitglieder des W3C in einem Arbeitsentwurf [W3C Arbeitsentwurf] zusammengefasst.

Ein rudimentäres HTML-Dokument mit ARIA-Rollen sieht folgendermaßen aus:

```
1 <!DOCTYPE html>
2 <html lang="de">
3
4   <head>
5     <meta charset="utf-8">
6     <title>Titel der Webpraesenz</title>
7     <meta name="description" content="Beschreibung des Inhalts">
```

```
8     <meta name="keywords" content="Beispiel, HTML, Kopfbereich">
9   </head>
10
11  <body>
12        <!-- Kann mit der Rolle "banner" versehen werden. Ist aber normalerweise nicht noetig, da
              Standardannahme fuer dieses Element. -->
13    <header>
14      <p>Hier ist der header</p>
15    </header>
16        <!-- Setzt Orientierungspunkt fuer die Navigation der Seite. Kann sowhl lokale als auch
              globale Navigation enthalten. -->
17    <nav role="navigation">
18      <ul>
19        <li><a href="Erster/Link.html">Erste Seite</a></li>
20        <li><a href="Zweiter/Link.html">Zweite Seite</a></li>
21        <li><a href="Ditter/Link.html">Dritte Seite</a></li>
22        <li><a href="Vierter/Link.html">Vierte Seite</a></li>
23      </ul>
24    </nav>
25        <!-- Setzt Orientierungspunkt fuer den Hauptinhalt der Seite. -->
26    <main role="main">
27      <h1>Ueberschrift 1. Ebene</h1>
28      <p>Hier ist der Seiteninhalt zu sehen</p>
29      <h2>Ueberschrift 2. Ebene</h2>
30      <p>Noch mehr Seiteninhalt</p>
31
32            <h1>Ein Bild</h1>
33      <figure>
34        <img src="beispielbild.png">
35        <figcaption>Dies ist eine Bildunterschrift</figcaption>
36      </figure>
37      <article>
38        <header>
39          <h1>Ein Artikel</h1>
40        </header>
41        <section>
42        <h2>Eine geordnete Liste</h2>
43        <ol>
44          <li>Erster Eintrag</li>
45                  <li>Zweiter Eintrag</li>
46          <li>Dritter Eintrag</li>
47        </ol>
48
49        <h2>Eine ungeordnete Liste</h2>
50          <ul>
51            <li>Erster Eintrag</li>
52            <li>Zweiter Eintrag</li>
53                      <li>Dritter Eintrag</li>
54          </ul>
55        </section>
```

```
56              <!-- Setzt Orientierungspunkt fuer Fussnoten, Quellen, Copyright etc. Angaben -->
57        <footer role="contentinfo">
58          <h1>Quellenangabe</h1>
59          <div>
60                      <li>Erste Quelle</li>
61                      <li>Zweite Quelle</li>
62            <li>Dritte Quelle</li>
63          </div>
64        </footer>
65      </article>
66    </main>
67
68    <footer>
69      <p>Hier ist der Footer</p>
70    </footer>
71  </body>
72 </html>
```

Listing 3: Grundlegende Elemente mit den dazugehörigen ARIA Rollen

Der Doctype entspricht einem HTML5-Dokument, danach wird angegeben, dass das Dokument in deutscher Sprache verfasst ist. Im <head> werden dann die oben genannten Angaben gemacht. Danach folgen die schon bekannten Inhalte mit ihren jeweiligen ARIA-Rollen. Dies ist natürlich nur nötig, wenn damit gerechnet wird, dass auch blinde oder stark sehbehinderte Menschen das Angebot der Webpräsenz nutzen. Im Falle der Website für die Bachelorarbeit ist dies nicht der Fall, weshalb die ARIA-Rollen von nun an nicht mehr verwendet werden.

Zum Schluss sollte noch erwähnt werden, dass die Navigation bei großen Projekten in eine separate Datei ausgelagert werden sollte. Während es bei Projekten wie der Bachelorwebsite, welche recht wenige Links haben, noch möglich ist, die Navigation in jedes einzelne HTML-Dokument zu schreiben, wird es bei größeren Projekten schnell zu einer sehr zeitaufwendigen Aufgabe. Wird die Navigation ausgelagert, müssen Änderungen lediglich in einer Datei vorgenommen werden und nicht in allen. Zudem wird durch die verringerte Größe der HTML-Dateien die Performance der Seite begünstigt. Möchte man die Navigation daher auslagern, muss die Navigationsdatei dann, als externe Datei, in die anderen HTML-Dokumente eingebunden werden. Dies muss bisher noch mit einer serverseitigen Programmiersprache wie PHP verwirklicht werden.

Mit Hilfe dieser Tags können also die gewünschten Inhalte hinzugefügt und grob strukturiert werden. Für jede Seite ist dabei ein eigenes HTML-Dokument nötig. Diese Dokumente können untereinander verlinkt sein.

Mit Hilfe der erarbeiteten Grundlagen kann nun das HTML-Dokument für die Startseite des Bachelorprojektes erstellt werden:

```
1 <!DOCTYPE html>
2 <html lang="de">
3    <head>
4       <meta charset="utf-8">
5       <title>Willkommen</title>
6       <meta name="description" content="Startseite des Bachelorprojektes von Fabian Klautke">
7       <meta name="keywords" content="Website, HTML, CSS, Tutorial, Anfaenger">
8       <meta name="viewport" content="width=device-width, initial-scale=1.0">
9       <!-- Zeit in Sekunden, wie lange diese Seite aus dem lokalen Zwischenspeicher geladen wird,
            bevor sie neu von Server abgerufen wird. Hier 1209600 sek = 14 Tage. -->
```

```
10          <meta http-equiv="Cache-Control" content="max-age=1209600">
11      </head>
12
13
14      <body>
15          <!-- Die Liste und Listenelemente wurden weggelassen, um ein schlankeren HTML-Code bei
                gleichen Positionierungsmoeglichkeiten zu ermoeglichen. -->
16          <nav>
17
18              <!-- Seitentitel und gleichzeitiger Home-Link. -->
19              <a href="startseite.html">Bachelorarbeit Fabian Klautke</a>
20
21              <a href="Grundlagen/Grundlagen.html">Don'ts der Website Erstellung</a>
22              <a href="Konzept/Konzept.html">Zuerst das Konzept</a>
23              <a href="Umsetzung/Umsetzung.html">Dann die Umsetzung</a>
24              <a href="Evaluierung/Evaluierung.html">Testen: Funktioniert alles?</a>
25          </nav>
26          <main>
27              <h1>Willkommen!</h1>
28              <p>
29                  Auf dieser Seite koennen Sie hilfreiche Tipps und Hinweise zur Erstellung Ihrer ersten Website finden.
                        Dabei ist die Website im ersten Sinne fuer Laien gedacht, welche noch nicht viel mit
                        Webentwicklung oder Design zu tun hatten. In den folgenden Seiten finden Sie kurze Erklaerungen
                        zu den wichtigsten Dingen, die bei der Erstellung einer Website zu beachten sind und wie Sie am
                        Ende selbst testen koennen, ob alles so laeuft wie es soll. Die Website wurde im Rahmen der
                        Bachelorarbeit von Fabian Klautke an der Hochschule Anhalt, Standort Koethen erstellt.
30              </p>
31          </main>
32
33          <!-- Die Liste und Listenelemente wurden weggelassen, um ein schlankeren HTML-Code bei
                gleichen Positionierungsmoeglichkeiten zu ermoeglichen. -->
34          <footer>
35              <a href="Meta/About.html">About</a>
36              <a href="Meta/Impressum.html">Impressum</a>
37          </footer>
38      </body>
39  </html>
```

Listing 4: HTML-Code für die Startseite der Bachelorwebsite

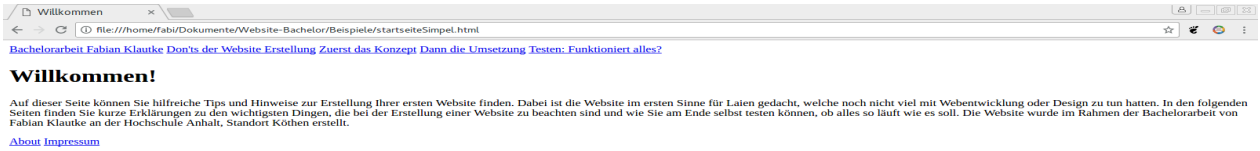

Abbildung 15: Darstellung der Startseite im Browser. Hier ohne CSS

4.2.1 Optimierung für verschiedene Auflösungen

Soll die Website sich zusätzlich noch automatisch an die Bildschirmgröße des Endgerätes, den sogenannten Viewport, anpassen, wird das Gebiet des Responsive Webdesign betreten. In HTML-Dokumenten wird dies mit dem „viewport“-Attribut bewerkstelligt. Dazu wird im HTML-Dokument im <head> folgende Zeile hinzugefügt:

```
1 <meta name="viewport" content="width=device-width, initial-scale=1">
```

Dies gibt an, dass sich der Inhalt immer der Breite des Endgerätes anpasst und der Zoomgrad beim Aufrufen der Seite 1.0 beträgt, also alle Inhalte 1:1 dargestellt werden. Alles Weitere wird dann im zugehörigen CSS-Dokument (4.3.1) mit Media Queries erledigt.

4.2.2 Benennung der Elemente

Da die Webpräsenz auch noch mit einem CSS gestaltet werden soll, müssen für die einzelnen Seitenelement noch Identifikatoren gefunden werden, welche in den CSS-Dateien referenziert werden können. Dies kann mit den HTML-Attributen class, name oder id geschehen. Eine id bezeichnet immer ein bestimmtes Element auf einer Webseite und muss daher auf dieser einen Seite einzigartig sein. Es darf also keine zwei ids mit dem gleichen Bezeichner im HTML-Code einer einzelnen Seite geben. Das class-Attribut fast mehrere Elemente in einer Klasse zusammen.

Wird die Startseite der Bachelorarbeit durch Bezeichner erweitert, sieht der Code wie folgt aus:

```
1 <!DOCTYPE html>
2 <html lang="de">
3     <head>
4         <meta charset="utf-8">
5         <title>Willkommen</title>
6         <meta name="description" content="Startseite des Bachelorprojektes von Fabian Klautke">
7         <meta name="keywords" content="Website, HTML, CSS, Tutorial, Anfaenger">
8         <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
9         <!-- Zeit in Sekunden, wie lange diese Seite aus dem lokalen Zwischenspeicher geladen wird,
              bevor sie neu von Server abgerufen wird. Hier 1209600 sek = 14 Tage. -->
10        <meta http-equiv="Cache-Control" content="max-age=1209600">
11    </head>
12
13    <body>
14        <!-- Die Liste und Listenelemente wurden weggelassen, um ein schlankeren HTML-Code bei
              gleichen Positionierungsmoeglichkeiten zu ermoeglichen. -->
15        <nav class="globalNavi" id=navi>
16
17            <!-- Seitentitel und gleichzeitiger Home-Link. -->
18            <a href="startseite.html" id=home>Bachelorarbeit Fabian Klautke</a>
19
20            <a href="Grundlagen/Grundlagen.html" id=Gru>Don'ts der Website Erstellung</a>
21            <a href="Konzept/Konzept.html" id=Konz>Zuerst das Konzept</a>
22            <a href="Umsetzung/Umsetzung.html" id=Ums>Dann die Umsetzung</a>
23            <a href="Evaluierung/Evaluierung.html" id=Eva>Testen: Funktioniert alles?</a>
24        </nav>
25        <main id=cont class="inhalt">
26           <h1 id=ue>Willkommen!</h1>
27            <p id=beg>
28                    Auf dieser Seite koennen Sie hilfreiche Tipps und Hinweise zur Erstellung Ihrer ersten Website
                      finden. Dabei ist die Website im ersten Sinne fuer Laien gedacht, welche noch nicht viel mit
                      Webentwicklung oder Design zu tun hatten. In den folgenden Seiten finden Sie kurze
                      Erklaerungen zu den wichtigsten Dingen, die bei der Erstellung einer Website zu beachten sind
                      und wie Sie am Ende selbst testen koennen, ob alles so laeuft wie es soll. Die Website wurde
                      im Rahmen der Bachelorarbeit von Fabian Klautke an der Hochschule Anhalt, Standort
                      Koethen erstellt.
29            </p>
30        </main>
31
32        <!-- Die Liste und Listenelemente wurden weggelassen, um ein schlankeren HTML-Code bei
              gleichen Positionierungsmoeglichkeiten zu ermoeglichen. -->
33        <footer id=fuss>
34            <a href="Meta/About.html">About</a>
35            <a href="Meta/Impressum.html">Impressum</a>
36        </footer>
37    </body>
38 </html>
```

Listing 5: HTML-Code der Startseite mit individuellen Bezeichnern

Dabei sollten die Bezeichner immer dann eingesetzt werden, wenn das Element besondere Styleangaben erhalten soll. Zum Beispiel wird die Klasse „.globaleNavigation“ dazu genutzt, alle Elemente dieser Klasse einheitlich zu gestalten. Soll dann der Link zur Seite „Umsetzung“ weiter individualisiert werden, kann die id „#Umsetzung“ im CSS-Dokument angesprochen werden. Diese Bezeichner helfen ebenfalls dabei, den Code für die späteren Funktionstests performant und übersichtlich zu halten. Das Browserautomatisierungstool kann die Elemente direkt durch ihre Bezeichner ansprechen und es muss nicht erst auf lange Pfadangaben und sonstige Umwege zurückgegriffen werden. Für Anschauungszwecke wurden in diesem Beispiel der Startseite lange ids und Klassennamen gewählt. Um die HTML-Dokumente möglichst klein zu halten, werden bei der weiteren Umsetzung der Website kürzere Bezeichner gewählt.

4.3 Oberfläche mit CSS

Nun, wo ein HTML-Dokument vorhanden ist, kann die Seite mit Cascading Style Sheets, kurz CSS, optisch angepasst und attraktiv gestaltet werden. Um ein CSS einem HTML-Dokument zuzuweisen, muss es zuerst im <head> des HTML-Dokumentes verlinkt werden. Das geschieht durch diese Zeile:

```
<link rel="stylesheet" href="pfad/zum/stylesheet.css">
```

Dies weist das HTML-Dokument darauf hin, das Stylesheet zu verwenden, was unter dem angegebenen Pfad hinterlegt ist. Es ist auch möglich, den CSS-Code intern in die HTML-Datei einzufügen. Dazu muss dieser mit dem <style>-Tag umschlossen werden. Allerdings wird empfohlen, ein externes CSS zu verwenden. Somit können mehrere HTML-Dokumente das gleiche Stylesheet referenzieren. Wenn mehrere HTML-Dokumente das gleiche CSS verwenden, verringert das nicht nur die Anzahl der HTTP-Anfragen und somit auch die Ladezeit der Webpräsenz, es ist auch leichter, einen einheitlichen Stil für zusammengehörige Seitenelemente zu definieren. Die Stilangaben werden dabei immer so notiert, dass zu erst das betroffene HTML-Element benannt wird und die Stilangaben dahinter in geschwungenen Klammern folgen

```
Tag{Angabe}
```

Mehrere Angaben werden mit einem Semikolon getrennt und zur besseren Lesbarkeit untereinander geschrieben.

```
Tag{
  Angabe;
  Angabe;
  Angabe;
}
```

Stilangaben können gleich mehreren Tags auf einmal zugeordnet werden. Dies kann über Gruppierung von Elementen, oder die so genannten descendant- und child-Selektoren geschehen. Dabei stellen die descendants, auf deutsch Nachfahren, eines Tags jedes Element dar, das auf einer beliebigen Ebene von diesem Tag umschlossen ist. child-Elemente eines Tags sind dagegen solche Elemente, die direkte Nachfahren von diesem Tag sind. Außerdem können Selektoren wie Klassenname oder id benutzt werden, um einzelne Tags individuell anzusprechen. Ebenfalls können statt einzelner Tags auch alle Tags des gesamten HTML-Dokuments angesprochen werden.

```
<article class="Klassenname" id="article">
  <h1>Dies wäre ein child-Element von article</h1>
  <h2>Ein weiteres child-Element</h2>
  <section>
    <h3> Ein descendant-Element von article </h3>
    <ul>
      <li>Dies ist ebenfalls ein descendant-Element von article</li>
    </ul>
  </section>
</article>
```

Für dieses HTML-Dokument können die Stilangaben auf folgende Art definiert werden

```
*{Angabe} /* diese Angabe gilt für alle Elemente des HTML-Dokuments */

h1, h2, h3 {Angabe} /* die Angabe gilt für alle <h1>, <h2> und <h3> Elemente unabhängig davon, wo sie
    stehen */

#article {Angabe} /* diese Angabe gilt nur für das Element mit der id=article */

.Klassenname {Angabe} /* diese Angabe gilt für alle Elemente in dieser Klasse */

article >h1 {Angabe} /* diese Angabe gilt nur für <h1>-Elemente, die direkt von <article> abstammen.
    */

article h1 {Angabe} /* diese Angabe gilt für alle <h1>-Elemente, die auf irgendeiner Ebene von <
    article> umschlossen werden */
```

Eine ausführliche und aktuelle Liste aller CSS Selektoren stellt zum Beispiel das W3C auf seiner Website CSS Selektoren zur Verfügung.

Somit kann das in Programmlisting 1 vorgestellte simple HTML-Dokument mit Hilfe eines CSS gestaltet werden. Dazu wird eine entsprechende Verlinkung zu dem CSS an eine beliebige Stelle im <head>-Abschnitt gesetzt.

```
<link rel="Stylist" href="htmlSimpel.css">
```

Da das HTML-Dokument und das zugehörige Stylesheet im gleichen Ordner liegen, reicht es aus, lediglich den Namen des CSS einzutragen. Nun können mit dessen Hilfe die verschiedenen Elemente des HTML-Dokuments positioniert und optisch aufgewertet werden.

```
<!DOCTYPE html>
<html lang="de"> <!-- Legt die Sprache des Dokumentes fest. -->

    <!-- Pflichtelement. Hier werden die allgemeinen Angaben zum HTML-Dokument wie Titel,
        Zeichenkodierung, Beschreibung, Verwendetes CSS, Viewport etc. festgelegt. Ist nicht fuer
        den Nutzer sichtbar.-->
     <head>
        <meta charset="utf-8">
        <title>Titel der Webpraesenz</title>
        <meta name="description" content="Beschreibung der Website">
        <meta name="keywords" content="Beispiel, HTML, Kopfbereich, Schluesselwort">
        <link rel="stylesheet" href="htmlSimpel.css">
    </head>

<!-- Pflichtelement. Beinhaltet den sichtbaren Teil der Website. Kopfbereich, Seiteninhalt,
    Fussbereich und Alles, was der Nutzer sehen soll befindet sich hier.-->
    <body>
        <header>
            <p>Hier ist der header</p>
        </header>
        <main>
            <h1>Inhalt</h1>
```

```
            <p>Hier ist der Seiteninhalt zu sehen</p>
        </main>
        <footer>
            <p>Hier ist der footer</p>
        </footer>
    </body>
</html>
```

Listing 6: Minimales HTML-Dokument mit eingebundenem CSS

```
/* Stilangaben fuer alle Elemente, welche im <html> und im <body> Tag liegen.
    Diese Angaben werden auf unterliegende Ebenen angewendet, sofern nichts widerspruechliches
        deklariert wird.

    width - Angabe zur Breite des Elements. 100% = immer so breit wie die gesamte Seite.
    font-family - Angabe zur Schriftart und -familie. Verdana = Schriftart, sans-serif = Familie
        serifenloser Schriften.
*/
html body {
    width: 100%;
    font-family: Verdana, sans-serif;
}

/* Stilangaben fuer alle Elemente im <header> Tag.
    width: 90% - ueberschreibt die Breitenangabe im html body Abschnitt
    height - Angabe zur Hoehe des Elements. Kann in festen Massen mit px(pixel) angegeben werden
        oder in relativen Werten mit %.
    background - Fasst alle Angaben fuer den Hintergrund zusammen. Sowohl Farbe als auch
        Hintergrundbild(er) und dessen Groesse, Position etc.
    color - Angabe zur Schriftfarbe fest. Angabe kann in fuer einfache Farben in Klartext, fuer
        andere in HEX, RGB, RGBA, HSL oder HSLA.
    margin - Angabe zum aeusseren Abstand des Elements zum Elternelement. Hier zum <body>-Element.
        10px = oben und unten, 10px = links und rechts.
*/
header {
    width: 90%;
    height: 80px;
    background: blue;
    color: floralwhite;
    margin: 10px 10px;
}

/* Stilangaben fuer alle <p> Elemente im <header> Tag.
    padding - Angabe zum inneren Abstand des Textes zum Rand des Elements. 10px=Angabe fuer oben und
         unten, 12px=Angabe fuer links und rechts.
    font-size - Angabe zur Schriftgroesse. 200% = doppelt so gross wie die Standardschriftgroesse
        der Seite.
    font-weight - Angabe zur Schriftdicke. Kann in % oder als bold(fett), bolder(fetter), normal,
        lighter(duenner) oder inherit(vererbt vom Elternelement) angegeben werden.
*/
header p {
```

```
33      padding: 10px 10px;
34      font-size: 200%;
35      font-weight: bold;
36  }
37  /* Stilangaben fuer alle Elemente im <main> Tag
38      background-color - Angabe nur zur Hintergrundfarbe. Angabe als HEX-Code.
39      font-family - Ueberschreibt die Angabe zur Schriftart im html und body Tag.
40      color - Angabe zur Schriftfarbe. Angabe als RGB-Code.
41  */
42  main {
43      background-color: #e8d4cf;
44      font-family: Arial, sans-serif;
45      color: rgb(20, 20, 20);;
46  }
47
48  /* Stilangaben fuer alle <h1> Elemente im <main> Tag
49      font-style - Angabe zum Schriftstil(Schraege). italic = kursiv.
50      text-decoration - Angabe zur Dekoration fuer den sichtbaren Text im Element. overline = Linie
            uber dem Text.
51  */
52  main h1 {
53      font-style: italic;
54      text-decoration: overline;
55  }
56
57  /* Stileangaben fuer alle Elemente im <footer> Tag
58      font-size - Angabe zur Schriftgroesse. Feste Angabe in Pixel.
59      text-alignment - Angabe zur horizontalen Ausrichtung des Textes im Element
60      padding-left - Angabe zum inneren Abstand zum linken Rand des Elements.
61      padding-top - Angabe zum inneren Abstand zum oberen Rand des Elements.
62  */
63  footer {
64      font-size: 20px;
65      text-align:center;
66      padding-left: 10px;
67      padding-top: 15px;
68  }
```

Listing 7: CSS welches im Minimalen HTML-Dokument referenziert wird

Abbildung 16: Darstellung der visuellen Veränderungen durch CSS im Browser

Da bereits eine erste Auswahl zu Layout (3.1), Farbe (3.5) und Schriftart (3.6) getroffen wurden, kann jetzt ein erstes CSS für die Startseite der Bachelorarbeit geschrieben werden:

```
1  <!DOCTYPE html>
2  <html lang="de">
3      <head>
4          <meta charset="utf-8">
5          <title>Willkommen</title>
6          <meta name="description" content="Startseite des Bachelorprojektes von Fabian Klautke">
7          <meta name="keywords" content="Website, HTML, CSS, Tutorial, Anfaenger">
8          <meta name="viewport" content="width=device-width, initial-scale=1.0">
9           <!-- Zeit in Sekunden, wie lange diese Seite aus dem lokalen Zwischenspeicher geladen wird,
                bevor sie neu von Server abgerufen wird. Hier 1209600 sek = 14 Tage. -->
10         <meta http-equiv="Cache-Control" content="max-age=1209600">
11
12         <!-- Pfad zu Stylesheet.-->
13         <link rel="stylesheet" href="stylesheet.css">
14         <!-- Pfad zur Schriftart auf den Google Servern und Verweis aufs Stylesheet. -->
15         <link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
16         <!-- Wenn JQuery verwendet werden soll, kann es so von Googles CDN geladen werden. Hier
               JQuery-Version 3.2.1. -->
17         <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
18         <!-- Hier wird die Javascriptdatei mit den spaeter verwendeten Funktionen eingebunden. -->
19         <script src="funktionen.js"></script>
20     </head>
21
22     <!-- onload und onresize rufen die Funktion inhaltUnterNavi() jeweils beim ersten Laden und wenn
            das Fenster die Groesse aendert auf, um den Inhalt neu zu platzieren. -->
23     <body onload="inhaltUnterNavi()" onresize="inhaltUnterNavi()">
24         <!-- Die Liste und Listenelemente wurden weggelassen, um ein schlankeren HTML-Code bei
               gleichen Positionierungsmoeglichkeiten zu ermoeglichen. -->
25         <nav class="globalNavi" id=navi>
26
27             <!-- Seitentitel und gleichzeitiger Home-Link. -->
28             <a href="startseite.html" id=home>Bachelorarbeit Fabian Klautke</a>
29
30             <a href="Grundlagen/Grundlagen.html" id=Gru>Don'ts der Website Erstellung</a>
31             <a href="Konzept/Konzept.html" id=Konz>Zuerst das Konzept</a>
32             <a href="Umsetzung/Umsetzung.html" id=Ums>Dann die Umsetzung</a>
33             <a href="Evaluierung/Evaluierung.html" id=Eva>Testen: Funktioniert alles?</a>
34
35             <!-- Menu-Icon, dass bei kleinen Bildschirmgroessen statt der Navigation angezeigt wird.
                   &#9776; = Unicode-Symbol. Drei Striche untereinander. -->
36             <a href="javascript:void(0);" class="menu" onclick="navbarToggle()">&#9776;</a>
37
38         </nav>
39          <noscript id="keinJS" class="keinJS"><div>Diese Website benÃ¶tigt Javascript, um korrekt
               dargestellt zu werden. Auf dieser Seite kÃ¶nnen Sie nachlesen, wie Javascript in ihrem Browser aktiviert
               wird: <a href="https://www.java.com/de/download/help/enable_browser.xml">Javascript
```

```
            aktivieren</a></div>
40      </noscript>
41      <main id=cont class="inhalt">
42        <h1 id=ue>Willkommen!</h1>
43         <p id=beg>
44                Auf dieser Seite koennen Sie hilfreiche Tipps und Hinweise zur Erstellung Ihrer ersten Website
                  finden. Dabei ist die Website im ersten Sinne fuer Laien gedacht, welche noch nicht viel mit
                  Webentwicklung oder Design zu tun hatten. In den folgenden Seiten finden Sie kurze
                  Erklaerungen zu den wichtigsten Dingen, die bei der Erstellung einer Website zu beachten sind
                  und wie Sie am Ende selbst testen koennen, ob alles so laeuft wie es soll. Die Website wurde
                  im Rahmen der Bachelorarbeit von Fabian Klautke an der Hochschule Anhalt, im Fachbereich
                  Informatik und Sprachen erstellt.
45         </p>
46      </main>
47
48      <!-- Die Liste und Listenelemente wurden weggelassen, um ein schlankeren HTML-Code bei
              gleichen Positionierungsmoeglichkeiten zu ermoeglichen. -->
49      <footer id=fuss>
50         <a href="Meta/About.html" id=about>About</a>
51         <a href="Meta/Impressum.html" id=impress>Impressum</a>
52      </footer>
53    </body>
54 </html>
```

Listing 8: HTML-Code für die Startseite der Bachelorwebsite mit eingebundenem CSS

```
1  /*
2  .color-primary-0 { color: #383276 } Main Primary color
3  .color-primary-1 { color: #837EB1 }
4  .color-primary-2 { color: #5A5494 }
5  .color-primary-3 { color: #1F1959 }
6  .color-primary-4 { color: #0D083B }
7
8  .color-secondary-1-0 { color: #7A296A } Main Secondary color (1)
9  .color-secondary-1-1 { color: #B77AAB }
10 .color-secondary-1-2 { color: #984C8A }
11 .color-secondary-1-3 { color: #5B0F4D }
12 .color-secondary-1-4 { color: #3D0031 }
13
14 .color-secondary-2-0 { color: #25705A } Main Secondary color (2)
15 .color-secondary-2-1 { color: #70A897 }
16 .color-secondary-2-2 { color: #468C77 }
17 .color-secondary-2-3 { color: #0E543F }
18 .color-secondary-2-4 { color: #003827 } */
19
20 /* Stilangaben fuer alle Elemente im body Tag
21
22    margin - Angabe zum aeusseren Abstand des Elements zum Elternelement. 0 = kein Abstand.
          Ueberschreibt die Standardabstaende von Elementen.
```

```
23      font-family - Schriftart und Familie 'Open Sans' = Webfont von Google, Verdana =
            Systemschriftart als Ausweichmoeglichkeit, sans-serif = Serifenlose Schriftfamilie
24  */
25  body {
26      margin:0;
27      font-family: 'OpenSans', Verdana, sans-serif;
28  }
29
30  /* Stilangaben fuer das Element mit der id "home" aka. den Seitentitel
31      color - Angabe zur Schriftfarbe. Angabe in Hexcode oder RGB, RGBA, HSL, HSLA
32      background-color - Angabe zur Hintergrundfarbe.
33      border-bottom - Angabe zum Rahmen nur am unteren Rand des Elements. solid = normale Linie, 2px =
            Breite, #837EB1 = Farbe
34      border-right - Angabe zum Rahmen nur am rechten Rand.
35  */
36  #home {
37      color: #837EB1;
38      background-color: #383276;
39      border-bottom: solid 2px #837EB1;
40      border-right: solid 2px #837EB1;
41  }
42
43  /* Stilangaben fuer alle Elemente in der Klasse ".globalNavi"
44
45      overflow - Angabe, was passieren soll, wenn der Inhalt groesser als das Element wird. hidden =
            Ueberfluss wird versteckt.
46      position - Angabe zur Positionierung der Elemente. fixed = fest und scroll nicht mit.
47      top: Angabe, wo die obere Begrenzung des Elements beginnt. 0 = direkt am oberen Rand der Seite.
48      height - Angabe zur Hoehe des Elements. auto = Passt sich automatisch an die Hoehe der
            Kinderelemente an.
49      widht - Angabe zur Breite des Elements. 100% = gesamte Breite des Bildschirms.
50      box-shadow - Angabe zum Schlagschatten des Elements. 0px = horizontale Verschiebung(noetig), 1px
            = vertikale Verschiebung(noetig), 1px = Verschwommenheit des Schattens(optional), 2px =
            Ausbreitung des Schattens ueber die Masse des Elements(optional), #585858 = Farbe(noetig).
51      -webkit - Angabe fuer box-shadow in Safari, Chrome, Opera.
52      -moz - Angabe fuer box-shadow in Firefox
53  */
54  .globalNavi {
55      overflow: hidden;
56      position: fixed;
57      background-color: #B77AAB;
58      top: 0;
59      height: auto;
60      width: 100%;
61      border-bottom: solid 2px #3D0031;
62      box-shadow: 0px 1px 1px 2px #585858;
63      -webkit-box-shadow: 0px 1px 1px 2px #585858;
64      -moz-box-shadow: 0px 1px 1px 2px #585858;
65  }
66
```

```
/* Stilangaben fuer alle a-Elemente in der Klasse ".globalNavi"

    float - Angabe, die festlegt an welchen Rand des Elternelments das Element platziert ist. Andere Inhalte umfliessen das gefloetete Element.
    text-align - Angabe, die den Text in der Mitte des Elements zentriert.
    padding - Angabe welche den inneren Abstand des Inhalts des Elements zu den Elementraendern festlegt. 14px = Abstand oben und unten, 16px = Abstand links und rechts.
    text-decoration - Angabe, welche Dekorationen der Text im Element bekommt(z.B. Unterstriche, Oberstriche, Durchstriche). none = keine Dekoration, entfernt den automatischen Unterstrich von den Links.
    font-size - Angabe zur Schriftgroesse. 17px = 17 pixel.
    font-weight - Angabe zur Schriftdicke. bold = fett.
*/
.globalNavi a {
    float: left;
    color: #3D0031;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
    font-weight: bold;
}

/* Stilangaben fuer den Fall, dass der Mauszeiger ueber dem Element a schwebt

    background - Angabe zum Hintergrund. Faesst alle Einzelangaben wie Farbe, Bild, Groesse usw. zusammen.
*/
.globalNavi a:hover {
    background: #984C8A;
}

/* Stilangaben fuer das Elemente mit der Klasse .globalNavi und .menu aka. dem Menu-Icon.

    display - Angabe, die festlegt, wie das Element angezeigt wird. none = wird nicht angezeigt.
*/
.globalNavi .menu {
    display: none;
}

/* Stilangaben fuer das Element mit der Klasse ".inhalt".

    overflow-y - Angabe, die festlegt, was mit dem vertikalen Uberfluss geschehen soll. auto = wechselt automatisch zwischen hidden und scroll
    font-size - Angabe zur Schriftgroesse. 150% = das 1.5fache der normalen Schriftgroesse.
*/
.inhalt {
    text-align: center;
    font-size: 150%;
    overflow-y: auto;
```

```
111     width: 80%;
112     margin: auto;
113 }
114 
115 /* Stilangaben fuer das Element footer
116 
117     position - Angabe zur Positionierung des Elements. absolute = Wird vom restlichen Inhalt
            losgeloest.
118     bottom - Angabe, wo das untere Ende des Elements beginnt. 0 = am unteren Rand der Seite.
119 */
120 footer {
121     position: absolute;
122     background-color: #70A897;
123     width: 100%;
124     height: auo;
125     bottom: 0;
126     text-align: center;
127 }
128 
129 /* Stilangaben fuer die a-Elemente im footer.
130 
131     display - Angabe, die festlegt, wie das Element angezeigt wird. inline-block = als Bloecke
            nebeneinander.
132 */
133 footer a {
134     display: inline-block;
135     margin: 0 auto;
136     color: #003827;
137     text-decoration: none;
138     padding: 7px 8px;
139 }
```

Listing 9: Das CSS der Startseite ohne Media Queries

Abbildung 17: Darstellung der Startseite mit Veränderungen durch CSS

4.3.1 Responsive Webdesign mit Mediaqueries

Damit diese erste Webseite auch auf kleineren Endgeräten gut dargestellt wird, also responsive gemacht wird, kommen zusätzlich zu dem viewport im HTML-Dokument die sogenannten Media Queries, zu deutsch Medienabfragen, im CSS zum Einsatz. Mit diesen Media Queries kann die Breite des Endgerätes abgefragt werden. Sollten sich die Inhalte der Website bei dieser Breite nicht mehr gut und leserlich darstellen lassen, werden im CSS neue Styleangaben für diese Breite definiert. Die Bildschirmbreiten, an denen die Darstellung der Seiteninhalte sich verschiebt, sozusagen „bricht“, werden als breakpoints, also Bruchpunkte, bezeichnet. Um diese breakpoints zu finden, gibt es zwei Herangehensweisen. Bei der Ersten werden mit Hilfe von Browsertools oder Softwareprodukten die herkömmlichen Abmessungen der möglichen Endgeräte, wie zum Beispiel Smartphones oder Tablets, simuliert. Kommt es bei einer dieser Abmessungen zu einer fehlerhaften Darstellung der Inhalte, ist dies ein breakpoint. Bei der zweiten Herangehensweise wird das Browserfenster einfach manuell soweit verkleinert, bis die ersten Darstellungsfehler auftreten. Mittlerweile unterstützen herkömmliche Browser wie Chrome, Opera und Firefox die Darstellung des Bildschirminhalts in verschiedenen Größen. Diese Ansicht kann in allen drei Browsern mit der Tastenkombination Strg+Shift+M aufgerufen werden, oder über die Option „Element untersuchen“ im Rechtsklickmenü. In Opera und Chrome findet sich dort die Option „Toogle Device Toolbar“ rechts oberhalb der geöffneten Leiste – in Firefox links oberhalb. Zusätzlich kann in dieser Ansicht gut der zugrunde liegende HTML-Code und die CSS der untersuchten Seite betrachtet werden.

Da fast jeden Monat neue Mobilgeräte auf den Markt kommen, wird für die Bachelorwebsite die zweite Methode, also das freie Verkleinern des Browserfensters, gewählt, um die nötigen breakpoints zu bestimmen. Erweitert mit diesen breakpoints, sieht das CSS zur Website dann so aus:

```
1  /*
2  .color-primary-0 { color: #383276 } Main Primary color
3  .color-primary-1 { color: #837EB1 }
4  .color-primary-2 { color: #5A5494 }
5  .color-primary-3 { color: #1F1959 }
6  .color-primary-4 { color: #0D083B }
7
8  .color-secondary-1-0 { color: #7A296A } Main Secondary color (1)
9  .color-secondary-1-1 { color: #B77AAB }
10 .color-secondary-1-2 { color: #984C8A }
11 .color-secondary-1-3 { color: #5B0F4D }
12 .color-secondary-1-4 { color: #3D0031 }
13
14 .color-secondary-2-0 { color: #25705A } Main Secondary color (2)
15 .color-secondary-2-1 { color: #70A897 }
16 .color-secondary-2-2 { color: #468C77 }
17 .color-secondary-2-3 { color: #0E543F }
18 .color-secondary-2-4 { color: #003827 } */
19
20 /* Stilangaben fuer alle Elemente im body Tag
21
22    margin - Angabe zum aeusseren Abstand des Elements zum Elternelement. 0 = kein Abstand.
       Ueberschreibt die Standardabstaende von Elementen.
23    font-family - Schriftart und Familie 'Open Sans' = Webfont von Google, Verdana =
       Systemschriftart als Ausweichmoeglichkeit, sans-serif = Serifenlose Schriftfamilie
```

```
24 */
25 body {
26     margin:0;
27     font-family: 'OpenSans', Verdana, sans-serif;
28 }
29
30 /* Stilangaben fuer das Element mit der id "home" aka. den Seitentitel
31     color - Angabe zur Schriftfarbe. Angabe in Hexcode oder RGB, RGBA, HSL, HSLA
32     background-color - Angabe zur Hintergrundfarbe.
33     border-bottom - Angabe zum Rahmen nur am unteren Rand des Elements. solid = normale Linie, 2px =
           Breite, #837EB1 = Farbe
34     border-right - Angabe zum Rahmen nur am rechten Rand.
35 */
36 #home {
37     color: #837EB1;
38     background-color: #383276;
39     border-bottom: solid 2px #837EB1;
40     border-right: solid 2px #837EB1;
41 }
42
43 /* Stilangaben fuer alle Elemente in der Klasse ".globalNavi"
44
45     overflow - Angabe, was passieren soll, wenn der Inhalt groesser als das Element wird. hidden =
           Ueberfluss wird versteckt.
46     position - Angabe zur Positionierung der Elemente. fixed = fest und scroll nicht mit.
47     top: Angabe, wo die obere Begrenzung des Elements beginnt. 0 = direkt am oberen Rand der Seite.
48     height - Angabe zur Hoehe des Elements. auto = Passt sich automatisch an die Hoehe der
           Kinderelemente an.
49     widht - Angabe zur Breite des Elements. 100% = gesamte Breite des Bildschirms.
50     box-shadow - Angabe zum Schlagschatten des Elements. 0px = horizontale Verschiebung(noetig), 1px
           = vertikale Verschiebung(noetig), 1px = Verschwommenheit des Schattens(optional), 2px =
           Ausbreitung des Schattens ueber die Masse des Elements(optional), #585858 = Farbe(noetig).
51     -webkit - Angabe fuer box-shadow in Safari, Chrome, Opera.
52     -moz - Angabe fuer box-shadow in Firefox
53 */
54 .globalNavi {
55     overflow: hidden;
56     position: fixed;
57     background-color: #B77AAB;
58     top: 0;
59     height: auto;
60     width: 100%;
61     border-bottom: solid 2px #3D0031;
62     box-shadow: 0px 1px 1px 2px #585858;
63     -webkit-box-shadow: 0px 1px 1px 2px #585858;
64     -moz-box-shadow: 0px 1px 1px 2px #585858;
65 }
66
67 /* Stilangabenn fuer alle a-Elemente in der Klasse ".globalNavi"
68
```

```
69      float - Angabe, die festlegt an welchen Rand des Elternelments das Element platziert ist. Andere
            Inhalte umfliessen das gefloetete Element.
70      text-align - Angabe, die den Text in der Mitte des Elements zentriert.
71      padding - Angabe welche den inneren Abstand des Inhalts des Elements zu den Elementraendern
            festlegt. 14px = Abstand oben und unten, 16px = Abstand links und rechts.
72      text-decoration - Angabe, welche Dekorationen der Text im Element bekommt(z.B. Unterstriche,
            Oberstriche, Durchstriche). none = keine Dekoration, entfernt den automatischen Unterstrich
            von den Links.
73      font-size - Angabe zur Schriftgroesse. 17px = 17 pixel.
74      font-weight - Angabe zur Schriftdicke. bold = fett.
75  */
76  .globalNavi a {
77      float: left;
78      color: #3D0031;
79      text-align: center;
80      padding: 14px 16px;
81      text-decoration: none;
82      font-size: 17px;
83      font-weight: bold;
84  }
85
86  /* Stilangaben fuer den Fall, dass der Mauszeiger ueber dem Element a schwebt
87
88      background - Angabe zum Hintergrund. Faesst alle Einzelangaben wie Farbe, Bild, Groesse usw.
            zusammen.
89  */
90  .globalNavi a:hover {
91      background: #984C8A;
92  }
93
94  /* Stilangaben fuer das Elemente mit der Klasse .globalNavi und .menu aka. dem Menu-Icon.
95
96      display - Angabe, die festlegt, wie das Element angezeigt wird. none = wird nicht angezeigt.
97  */
98  .globalNavi .menu {
99      display: none;
100 }
101
102 /* Stilangaben fuer das Element mit der Klasse ".inhalt".
103
104     overflow-y - Angabe, die festlegt, was mit dem vertikalen Uberfluss geschehen soll. auto =
            wechselt automatisch zwischen hidden und scroll
105     font-size - Angabe zur Schriftgroesse. 150% = das 1.5fache der normalen Schriftgroesse.
106 */
107 .inhalt {
108     text-align: center;
109     font-size: 150%;
110     overflow-y: auto;
111     width: 80%;
112     margin: auto;
```

```
113 }
114
115 /* Stilangaben fuer das Element footer
116
117     position - Angabe zur Positionierung des Elements. absolute = Wird vom restlichen Inhalt
        losgeloest.
118     bottom - Angabe, wo das untere Ende des Elements beginnt. 0 = am unteren Rand der Seite.
119 */
120 footer {
121     position: absolute;
122     background-color: #70A897;
123     width: 100%;
124     height: auo;
125     bottom: 0;
126     text-align: center;
127 }
128
129 /* Stilangaben fuer die a-Elemente im footer.
130
131     display - Angabe, die festlegt, wie das Element angezeigt wird. inline-block = als Bloecke
        nebeneinander.
132 */
133 footer a {
134     display: inline-block;
135     margin: 0 auto;
136     color: #003827;
137     text-decoration: none;
138     padding: 7px 8px;
139 }
140
141
142 /*------------------------ Erster Breakpoint -------------------------------*/
143
144 /* Media Query, welches das CSS manipuliert, sobald das Browserfenster 1200px oder schmaler ist.
145
146     @media - beginnt das Media Query
147     only - Query wird nur von Browsern verarbeitet, die Media Queries auch verstehen und umsetzen
        koennen.
148     sceen - Das Ausgabemedium ist ein Bildschirm.
149     and - Verbindungswort um zwei Abfragen zusammenzufuehren
150     max-width = maximale Breite, die der Bildschirm haben kann, damit das Media Querie wirksam wird.
151 */
152 @media only screen and (max-width: 1200px) {
153
154     /* Neue Stylangabe fuer die a-Elemente in der Klasse ".globalNavigation". Alle vorherigen Angaben die Hier nicht
        explizit ueberschrieben werden bleiben gueltig. Auch Angaben aus vorherigen Media Queries.
155
156     font-size - Angabe zur Schriftgroesse. 1.2vw = 1vw ist gleich 1% der viewport width, also Bildschirmbreite. 1.2vw
        sind also 1.2% der Bildschirmbreite
157     */
```

```
158     .globalNavi a {
159       font-size: 1.2vw;
160         margin: auto;
161     }
162 }
163
164
165 /*------------------------ Zweiter Breakpoint ------------------------------*/
166
167 /* Media Query, welches das CSS manipuliert, sobald das Browserfenster 600px oder schmaler ist.
168 @media only screen and (max-width: 700px) {
169
170     /* Der blaue Hintergrund des Titels streckt sich auf die gesamte Navigationsleiste aus. */
171     .globalNavi {
172         background-color: #383276;
173     }
174
175     .globalNavi a {
176         font-size: 100%;
177     }
178
179     /* Alle a-Elemente, ausser dem ersten, werden ausgeblendet */
180     .globalNavi a:not(:first-child) {
181         display: none;
182     }
183
184     /* Das Menu-icon wird an den rechten Rand der Navigationsleiste gesetzt, als Block angezeigt und
            in einer sekundaer Farbe eingefaerbt. */
185     .globalNavi a.menu {
186         float: right;
187         display: block;
188         color: #B77AAB;
189     }
190
191     /* Beim Klick aufs Menu-icon wird die globale Navigation, durch die Javascript-Funktion, um die
            Klasse ".responsive" erweitert.
192
193     /* Stilangaben fuer Elemente der ".globalNavi.responsive" Klasse
194         z-index - Angabe zur Ebene des Elements. Die hoehere Zahl repraesentiert die vordere Ebene.
195     */
196     .globalNavi.responsive {
197         position: fixed;
198         background-color: #B77AAB;
199         z-index: 1;
200     }
201
202     .globalNavi.responsive .menu {
203         position: absolute;
204         right: 0;
205         top: 0;
```

```
206     }
207
208     .globalNavi.responsive a {
209         float: none;
210         display: block;
211         text-align: left;
212     }
213 }
```

Listing 10: Das CSS der Startseite erweitert um Media Queries

Wenn eine bestimmte Bildschirmbreite unterschritten wird, kommen dann automatisch die Angaben des dazugehörigen Media Queries zum Einsatz. Dabei sollte beachtet werden, dass frühere Angaben gültig bleiben, insofern sie nicht explizit in dem Media Query überschrieben werden. Dies gilt auch für Angaben vorheriger Media Queries. Soll bei sehr kleinen Bildschirmen statt der Navigation einen Button eingeblendet werden, mit dem sich die Navigation ausklappen lässt, muss zusätzlich zu HTML und CSS noch eine Programmiersprache wie Javascript zum Einsatz kommen. Javascript wurde in den 90iger Jahren von Netscape entwickelt um Websiten um einfache interaktive Funktionalitäten zu erweitern. Javascript und andere Programmiersprachen, welche eigens für solche einfachen Aufgaben entwickelt wurden, werden auch Skriptsprachen genannt [Vgl. Bewersdorff 2014, S. 42]. Javascript kann in Form von selbsterstellten scripts eingesetzt werden, oder durch die Verwendung von Angeboten wie Bootstrap [Bootstrap Website], welches Code-Vorlagen für allerlei Gebiete der Webentwicklung bereitstellt. Soll Javascript in seiner nativen Form verwendet werden, muss der Code lediglich in einen <script>-Tag geschrieben werden. Wird eine HTML-Version älter als HTML5 verwendet, ist es zudem nötig, den MIME-Type des folgenden Scripts mit

```
1 <script type="text/javascript">...Anweisungen...</script>
```

zu bestimmen. Der MIME-Type, kurz für Multipurpose Internet Mail Extension, gibt dabei an, welche Daten als nächstes gesendet werden. In HTML5 kann dieser Zusatz weggelassen werden.

```
1 <script>...Anweisungen...</script>
```

Soll der Einsatz von Javascript etwas erleichtert werden oder Frameworks, wie das schon erwähnte Bootstrap, verwendet werden, kommt JQuery [JQuery Website] zum Einsatz. JQuery ist eine Javascript-Bibliothek, welche Funktionen zur Selektion und Manipulation von HTML-Dokumenten enthält. Diese sind im Vergleich zum nativen Javascript verkürzt und einfacher zu handhaben. Soll JQuery verwendet werden, muss zu erst die entsprechende JQuery-Datei eingebunden werden. Dabei gibt es zwei Möglichkeiten. Man kann diese Bibliothek entweder auf den eigenen Server stellen, oder über Content Delivery Networks (CDN), also Netzwerke, welche die gewünschten Inhalte auf ihren Servern bereitstellen, nutzen. Danach muss JQuery in das HTML-Dokument eingebunden werden. Dies geschieht durch eine der folgenden Angaben im <head> des Dokuments:

```
1 <script src="Pfad/Zur/Datei/jquery-Dateiname.js"></script>
```

Hier wird davon ausgegangen, dass die JQuery-Datei auf dem gleichen Server wie die Website liegt und unter dem angegebenen Pfad zu finden ist.

```
1 <script src="https://code.jquery.com/jquery-latest.js"></script>
```

Diese Zeile bindet die aktuellste Version direkt von der JQuery-Website ein.

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

```
1 <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.1.min.js"></script>
```

Hier wird die angegebene Version von JQuery von den CDNs von Google bzw. Microsoft eingebunden. Eine der Angaben ist ausreichend.

Da es bei JQuery nötig ist, immer die gesamte Funktionsbibliothek einzubinden, was zu einer weiteren HTTP-Anfrage und damit Ladezeitverlängerung führt, ist die Verwendung nur empfohlen, wenn die Website viele oder relativ komplexe Javascript-Funktionen erfordert. Wenn, wie bei der Bachelorseite, nur eine kurze, einfache Funktion zum Ausklappen der Navigation bei kleinen Bildschirmen zum Einsatz kommt, reicht natives Javascript vollkommen aus und schont die Performance.

Die komplette Startseite im responsive Design wird in den folgenden Abbildungen dargestellt:

Abbildung 18: Bildschirme größer als 1.200 Pixel

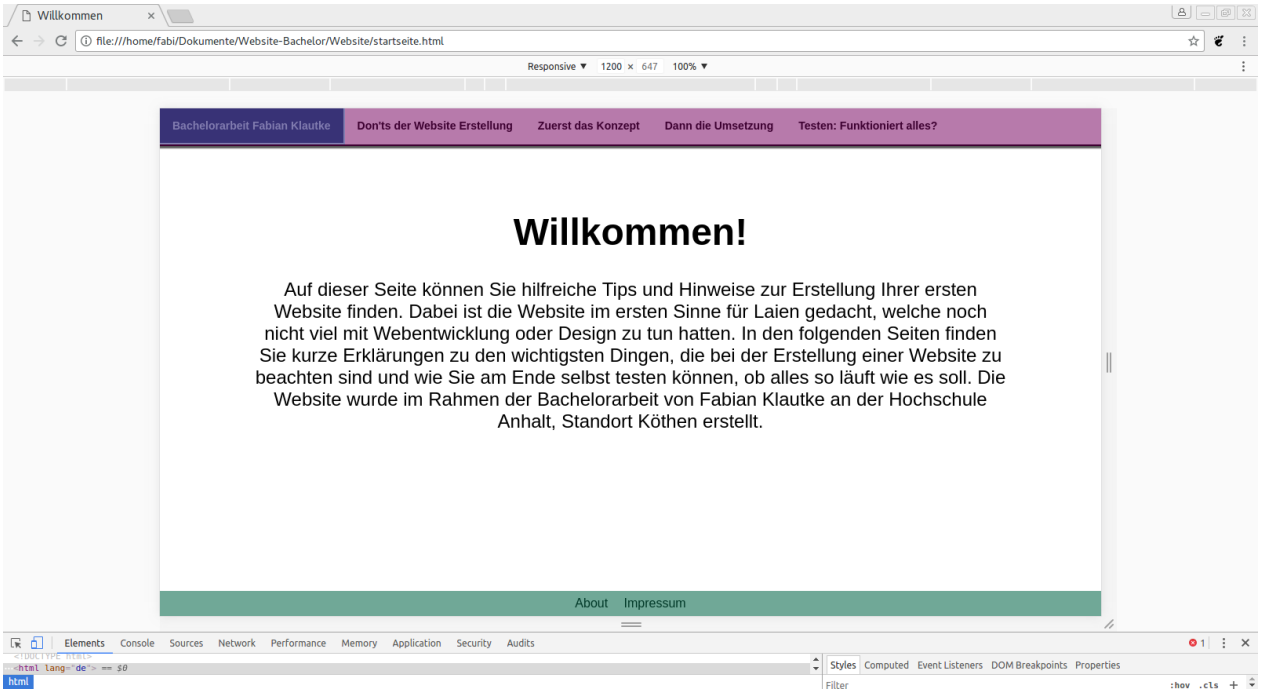

Abbildung 19: Bildschirme mit 1.200 Pixel Breite oder geringer Breite.

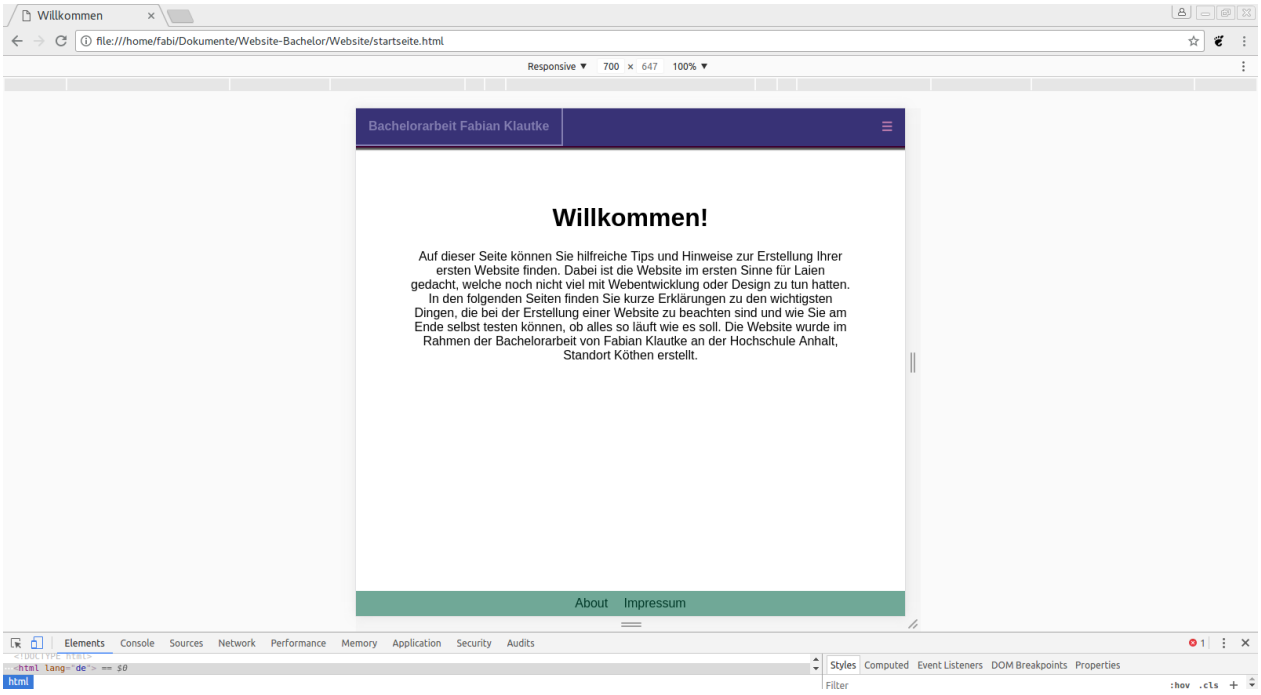

Abbildung 20: Bildschirme mit 700 Pixel Breite oder geringerer Breite.

Ab 1.200 Pixel Breite wird die Schriftgröße der Navigationselemente auf 1.3% der Bildschirmbreite angepasst, um zu vermeiden, dass die einzelnen Elemente sich untereinander schieben. Ab 700 Pixel wird dies allerdings zu klein und unleserlich. Stattdessen werden ab dieser Breite die Navigationselemente bis auf den Seitentitel ausgeblendet und mit einem Menü-Icon ersetzt, welches die restliche Navigation ausklappt, sobald darauf geklickt wird. Diese Media Queries gelten nun für alle Webseiten, welche das stylesheet.css der Startseite referenzieren. Es muss also für alle anderen Seiten ebenfalls getestet werden, ob diese zwei breakpoints ausreichen, um den Inhalt auf allen Bildschirmgrößen optisch ansprechend darzustellen, oder ob neue definiert werden müssen.

5 Evaluation

Wenn die Website fertiggestellt ist, sollten vor der in Inbetriebnahme, dem „live gehen“ der Website, einige Tests durchgeführt werden, um sicherzustellen, dass die Funktionalität gegeben ist und die Webpräsenz auch auf anderen Betriebssystemen und in unterschiedlichen Browsern korrekt dargestellt wird. Der folgende Abschnitt erklärt zuerst einige Aspekte von Funktionstests und Oberflächentests, welche dann auf die Website, welche im Rahmen dieser Arbeit entwickelt wurde, angewandt werden. So wird überprüft ob die gesetzten Ziele der Bachelorarbeit erfüllt wurden. Zudem wird ein kurzer Überblick über weitere Testbereiche gegeben.

5.1 Funktionstests

Durch Funktionstests werden hauptsächlich die interaktiven Elemente einer Website intensiv getestet. Im Einzelnen werden dabei in Sachen Navigation folgende Dinge getestet:

- Funktioniert die globale und lokale Navigation?
- Funktioniert das Hin- und Herspringen zwischen den unterschiedlichen Webseiten?
- Funktioniert das hin und her springen über interne Verlinkungen auf der gleichen Webseite
- Sind alle ausgehenden Verlinkungen aktuell und führen zu den richtigen Orten?

Kommen auf der Website zusätzlich noch ausfüllbare Formulare zum Einsatz, sollte weiterhin getestet werden:

- Sind die Textfelder der Formulare ausfüllbar?
- Werden nur zulässige Eingaben akzeptiert?

Zusätzlich zu diesen Punkten sollten auch HTML- und CSS-Dokumente auf Syntaxfehler und Kompatibilität mit den Indexierungsprogrammen der verschiedenen Suchmaschinenanbieter geprüft werden.

Während all diese Funktionen durchaus per Hand getestet werden können, gibt es verschiedene Software, Browsererweiterungen und Webangebote, welche die Arbeit sehr erleichtern.

5.1.1 Nützliche Software

Eine Möglichkeit das Testen der Navigation und Verlinkungen zu erleichtern, ist der Einsatz von Browser Automatisierungstools. Diese Tools orientieren sich an Selektoren wie id, ClassName, name um die Elemente der Website, auch DOM-Elemente genannt, zu identifizieren und mit ihnen zu interagieren. So lässt sich das Vorgehen eines Nutzers auf der Website simulieren. Viele dieser Automatisierungswerkzeuge zeichnen dabei zuerst eine manuelle Interaktion mit der Website auf und speichern diese in Programmen ab, die danach immer wieder abgerufen werden können. So können Nutzerinteraktionen wie die Navigation über verschiedene Webseiten, der Login, oder das Ausfüllen von Formularen simuliert und mit einer weit höheren Geschwindigkeit, als es einem Menschen möglich wäre, immer wieder abgespielt werden. Mit mehr spezialisierten Automatisierungswerkzeugen können komplexere Tests auch direkt in einer Entwicklungsumgebung programmiert werden. Diese erfordern allerdings zumeist Vorkenntnisse in Programmierung und sind daher für die Zielgruppe dieser Arbeit weniger geeignet.

Daher wird das Augenmerk auf Automatisierungswerkzeuge wie Chromium Browser Automation von Google exklusiv für den Chrome Browser bzw. die freie Version von iMakros von Ipswitch, Inc., welches sowohl für Firefox als auch für Chrome angeboten wird, gelegt. Beide Werkzeuge werden als Add-Ons für die jeweiligen unterstützten Browser angeboten, was die Installation sehr unkompliziert macht.

Zum validieren von HTML und CSS bietet das W3C einen kostenfreien Dienst unter https://validator.w3.org an. Hier kann der HTML-Code oder das CSS entweder per URL, Datei Upload oder direkter Eingabe zugeführt werden.

Die Bachelorwebsite wird nun also zuerst einmal auf korrekte Funktionsweise überprüft. Zuerst wird mit Hilfe des Validators des W3C überprüft, ob die erstellten HTML-Dokumente und das Stylesheet fehlerfrei sind und den üblichen Konventionen entsprechen. Am Beispiel der Startseite wird gezeigt, wie ein Ergebnis des Validators aussieht. Dazu wird zuerst das HTML-Dokument startseite.html hochgeladen und validiert. Des ergab folgendes Ergebnis:

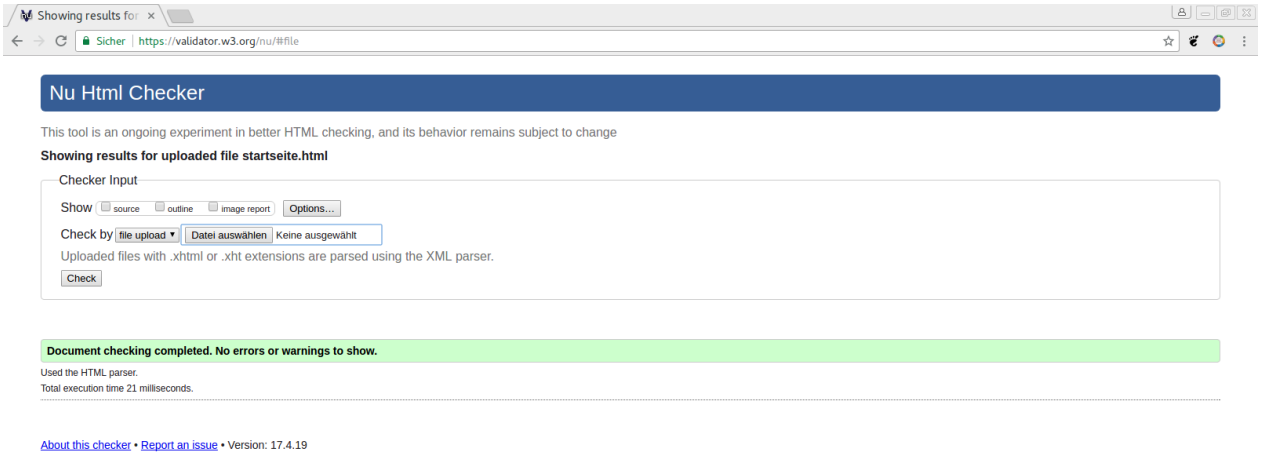

Abbildung 21: W3C Validator: HTML

Der Validator hat keinerlei Fehler im HTML-Dokument gefunden. Somit kann es ohne weiteres in der Website verwendet werden. Dabei ist zu beachten, dass der Validator lediglich die Struktur des HTML-Dokuments überprüft, ob z.B. Tags korrekt geöffnet und geschlossen wurden. Schreibfehler werden nicht überprüft und muss vom Nutzer selbst korrigiert werden. Die Validierung der verbleibenden HTML-Dokumente der Website ergab ebenfalls keine Fehler.

Nachdem das HTML-Dokument der Startseite keine Fehler aufweist, kann nun das CSS geprüft werden. Dazu wird das Dokument stylesheet.css hochgeladen. Der Validator kommt zu folgendem Ergebnis.

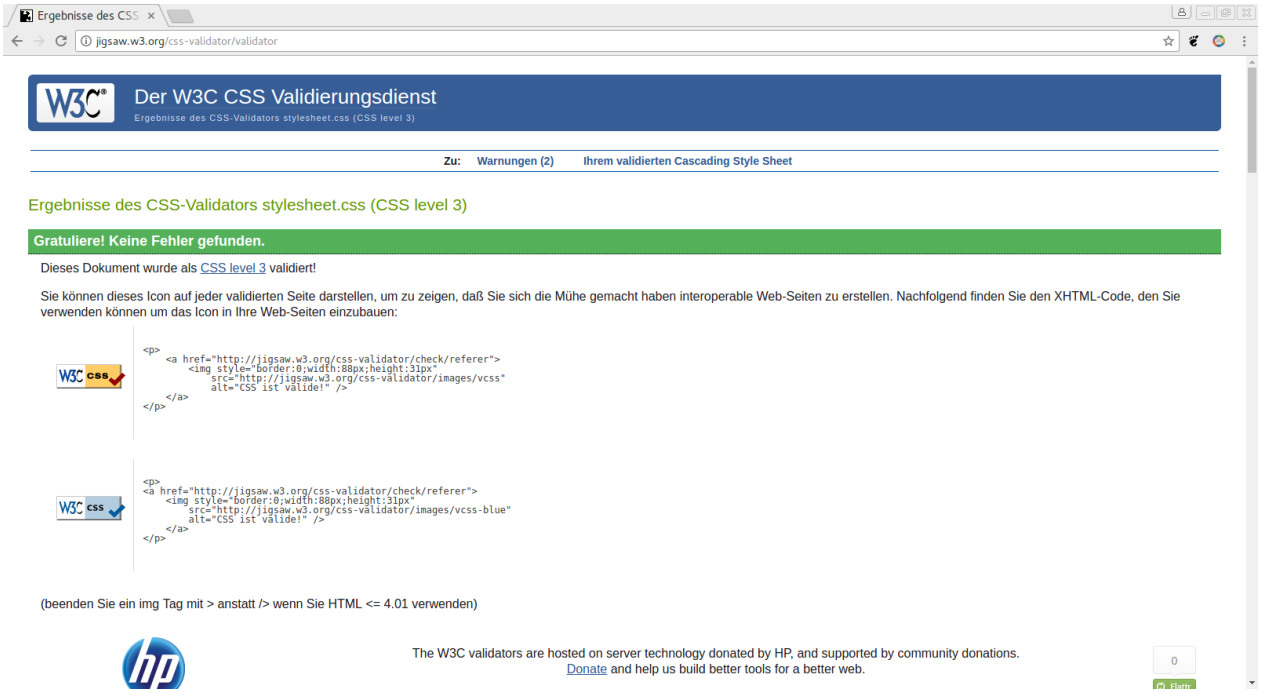

Abbildung 22: W3C Validator: CSS

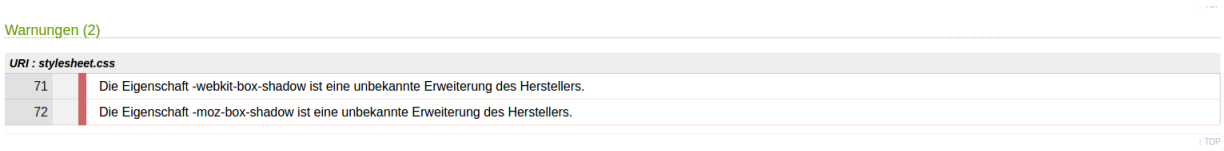

Abbildung 23: W3C Validator: CSS Warnungen

Das CSS ist also korrekt bis auf zwei Warnungen, welche sich auf die browserspezifischen Angaben zum box-shadow für Firefox und Webkit-Browser beziehen. Somit kann also auch das CSS ohne Bedenken verwendet werden. Auch hier zeigt sich der Vorteil eines externen CSS, da nur eine Datei validiert werden muss.

Bei der Bachelorwebsite boten sich aufgrund der geringen Komplexität manuelle Funktionstests an. Dazu wurde die Website einer Gruppe von 10 Probanden im Alter von 20-60 Jahren zur Verfügung gestellt. Die Gruppe setzte sich aus Personen mit unterschiedlichen Vorkenntnissen in den Gebieten Informatik und Webdesign bzw. Webentwicklung zusammen. 6 Probanden besaßen keinerlei Vorkenntnisse in den Gebieten, zwei Probanden hatten ein Informatikstudium abgeschlossen, in dessen Rahmen sie in zwei Modulen mit den Themen Webdesign und -entwicklung in Berührung kamen. Die verbleibenden zwei Probanden waren ausgebildete Webdesigner bzw. Webentwickler mit mehrjähriger Berufserfahrung. Die Probanden benutzten dabei ihre eigenen Hard- und Softwarekonstellationen. So wurde die Website in den Browsern Firefox, Chrome, Opera, Edge und Internet Explorer getestet. 6 Probanden benutzten dabei das Betriebssystem Windows 10, zwei Probanden benutzten Windows 7 und zwei Probanden unterschiedliche Linux-Distributionen basierend auf Ubuntu. Auf der Hardwareseite kamen drei Laptops und 6 Flachbildschirme verschiedener Auflösungen und Preisklassen, sowie ein Tablet und ein Röhrenmonitor zum Einsatz.

Um die Funktionstüchtigkeit der Website zu testen, erhielten die Probanden folgende Anweisungen:

- Navigieren Sie zum Abschnitt mit der Überschrift "Die richtige Farbwahl". Wie lange haben Sie gebraucht? Was wurde angeklickt?
- Navigieren Sie zum Abschnitt mit der Überschrift "technische Probleme". Wie lange haben Sie gebraucht? Was wurde angeklickt?
- Navigieren Sie zum Abschnitt mit der Überschrift "Das gute Aussehen mit CSS". Wie lange haben Sie gebraucht? Was wurde angeklickt?
- Navigieren Sie mit Hilfe der globalen und lokalen Navigation über die Webseiten? Wird das erwartet Navigationsziel erreicht?

Die globale Navigation konnte von allen Probanden ohne Probleme benutzt werden. Bei der lokalen Navigation fiel auf, dass die Überschrift des angeklickten Abschnitts von der globalen Navigation verdeckt wurde und somit nicht lesbar war. Dies führte bei den Probanden zu Orientierungslosigkeit. Ebenfalls wurde angemerkt, dass die lokale Navigation des Kapitels „Zuerst das Konzept“ einen Link zum Abschnitt „Responsive Webdesign“ enthielt. Da dieser Abschnitt allerdings bei einer früheren Überarbeitung des Seiteninhalts verworfen wurde, geschah bei der Benutzung des entsprechenden Links nichts, was die Probanden verwirrte.

Obwohl die Frage, wie lange die Probanden brauchten, um einen bestimmten Abschnitt zu finden, eher bei den Oberflächentests einzuordnen ist, wurde sie bereits im Rahmen der Funktionstests gestellt. So konnte sicher gestellt werden, dass die Probanden nicht schon beim Testen der Navigation mit der Website vertraut wurden, da dies ansonsten die Ergebnisse verfälscht hätte.

5.2 Oberflächentests

Neben den Funktionen der Website sollte auch die Oberfläche überprüft werden. Dazu gehören sowohl die Suche nach Rechtschreibfehlern als auch die Suche nach unkolorierten oder falsch kolorierten Elementen, unformatierten oder falsch formatierten Textpassagen und Ähnlichem. Weiterhin gehört es zu Oberflächentests dazu, die Darstellung der Website auf verschiedenen Kombinationen aus Betriebssystemen und Browsern zu überprüfen. Am einfachsten lässt sich dies bewerkstelligen, indem Probanden der Link zur Website geschickt wird bzw. ihnen die Dateien zur Verfügung gestellt wird, damit sie die Website direkt auf ihrem Rechner aufrufen können.

Eine andere Möglichkeit sind Webseiten, welche sogenanntes Cross-Browser-Testing anbieten. Die Anbieter dieser Webseiten setzen oft virtuelle Maschinen ein, um eine Vielzahl an Kombinationen aus Betriebssystem, Browser und Auflösung zu simulieren. Wird einem dieser Dienste die URL der eigenen Webpräsenz zugeführt, wird sie in den verschiedenen virtuellen Umgebungen aufgerufen und ein Screenshot der Website in der jeweiligen simulierten Umgebung gemacht. Diese Screenshots können dann vom Nutzer auf eventuelle Darstellungsfehler überprüft werden. Mit der Nutzung dieser Dienste sind allerdings so gut wie immer Kosten verbunden.

Für die Oberflächentests wurde die Website wieder den gleichen 10 Probanden zur Verfügung gestellt. Diesmal bekamen sie folgende Anweisungen:

- Beurteilen Sie die Farbwahl. Sind alle Elemente konstant koloriert? Ist alles gut lesbar?
- Ist der Inhalt der Website angenehm lesbar und verständlich? Werden alle Buchstaben korrekt dargestellt?
- Werden die Bilder auf der Website korrekt dargestellt. Vergrößern sich die Bilder beim darauf klicken?

Zudem wurde hier die Dauer der Navigation zu den Abschnitten ausgewertet, welche vorher ihm Rahmen der Funktionstests abgefragt wurden. Die Abschnitte „Die richtige Farbwahl“ und „Das gute Aussehen mit CSS“ wurden von den Probanden sehr schnell gefunden. Sie brauchten dazu durchschnittlich drei Klicks und ca. eine Minute. Ebenfalls wurden die Überschriften der beiden Abschnitte und ihre übergeordneten Kapitel als passend bewertet. Beim Abschnitt „Technische Probleme“ brauchten die Probanden hingegen entschieden länger. Durchschnittlich 7 Klicks. Einige Probanden mussten sogar die kompletten Navigationspunkte durchsuchen, um den Abschnitt zu finden. Zudem wurde hier die Überschrift des Abschnitts als verwirrend und unpassend empfunden.

Die Farbgebung der Website wurde von Probanden grundsätzlich positiv aufgenommen. Die Elemente waren über die gesamte Website hinweg konstant koloriert. Die älteren Probanden hatten beim Lesen der Navigationspunkte leichte Schwierigkeiten und wünschten sich eine hellere Schriftfarbe bzw. einen höheren Kontrast.

Den Inhalt beurteilten die Probanden als gut lesbar und auch für Einsteiger leicht verständlich. Die Schrift wurde auf allen Betriebssystemen korrekt dargestellt und die Schriftgröße von Probanden als angenehm groß empfunden. Allerdings mussten drei Probanden darauf hingewiesen werden, dass die Webpräsenz Javascript benötigt, damit der Inhalt korrekt dargestellt wird. Einige Probanden wiesen auch darauf hin, dass es unwahrscheinlich ist, dass Laien den Quellenangaben im Text Beachtung schenken. Stattdessen wurde vorgeschlagen, Fußnoten oder ähnlich unauffällige Verweise zu benutzen. Außerdem wurde bemängelt, dass die Überschriften der verschiedenen Abschnitte mal mit Kleinbuchstaben und mal mit Großbuchstaben begannen.

Die Bilder wurden bei allen Probanden zügig geladen und korrekt angezeigt. Ebenfalls funktionierte die Vergrößerung der Bilder bei allen Probanden ohne Probleme. Allerdings verwirrte die Probanden, dass sich auch bei Bildern, welche nicht zur Vergrößerung gedacht waren, der Mauszeiger eine Vergrößerungsmöglichkeit indizierte. Dies führte bei den Probanden zu der fälschlichen Annahme, dass die Funktion nicht richtig arbeitete.

5.3 Weitere Tests

Für ein eigenes Websiteprojekt sind die ersten beiden Punkte Funktionstests (5.1) und Oberflächentests (5.1) meist ausreichend. Allerdings können auch noch andere Sachen getestet werden.

5.3.1 Stresstests

Stresstests werden genutzt, um die Belastungsgrenzen der Website auszutesten. Dazu wird eine große Anzahl an Anfragen auf die Seite an sich, oder auf einzelne Funktionen simuliert. So kann sichergestellt werden, dass die Webpräsenz auch bei hoher Belastung weiter zuverlässig funktioniert, oder sich von auftretenden Fehlern schnell wieder erholt. Oft werden auch Funktionen wie die Anmeldung bzw. Registrierung getestet, um sicherzustellen, dass die Verbindung auch bei einem großen Nutzeraufkommen stabil bleibt. Bei kleinen privaten Websites sind Stresstests eher ungewöhnlich.

5.3.2 Sicherheitstests

Die Sicherheit einer Website ist äußerst wichtig, nicht nur, um das Vertrauen der Nutzer zu sichern, als auch, um vor rechtlichen Konsequenzen im Falle eines erfolgreichen Angriffs geschützt zu sein. Im Rahmen von Sicherheitstests sollte zum Beispiel geprüft werden, ob die Eingabefelder der Website unzulässige Zeichen abfangen. Somit kann verhindert werden, dass Datenbankabfragen in diese Felder eingetragen werden, welche dann an die Datenbank weitergeleitet werden, wodurch ungewollt Informationen freigegeben werden. Zudem sollte getestet werden, ob durch die Manipulation der Adressleiste ohne Anmeldung auf geschützte Seiten zugegriffen werden kann oder ob angemeldete Nutzer auf das Profil anderer zugreifen können. Ein weiterer wichtiger Punkt ist das Testen, ob sensible Daten verschlüsselt gesendet werden.

6 Fazit und Aussicht

Durch die Evaluierung hat sich herausgestellt, dass sich mit dem im Rahmen der Bachelorarbeit erarbeiteten Wissen eine Website konzipieren und entwickeln lässt, welche von sehr unterschiedlichen Nutzern positiv aufgenommen wird. Dabei setzte sich die Gruppe von Probanden bewusst aus Personen unterschiedlichen Alters und mit unterschiedlichen Hintergründen zusammen. Es wurde darauf geachtet das ein Großteil der Probanden mit der Zielgruppe der Webpräsenz übereinstimmt, d.h. Menschen mit keinerlei Vorkenntnissen in Sachen Webdesign und Webentwicklung. Allerdings wurden auch Menschen mit einem fundierten Wissen und viel Erfahrung in den beiden Gebieten mit einbezogen, um so etwas wie eine Expertenmeinung zu gewinnen.

Die Funktions- und Oberflächentests durch die Probanden zeigten einige Mängel an der Bachelorwebsite auf. Dies bestätigte Aussagen, welche in den Abschnitten zu den konzeptionellen und technischen Problemen bei der Website-Erstellung getätigt wurden. So hatten die Probanden zum Beispiel Probleme den Abschnitt „Technische Probleme“ zu finden, da sich anhand der Überschrift, nicht die Thematik erkennen lies. Dies bestätigte eine Aussage des Abschnitts „Undurchdachte Navigation“ (2.2.1), nachdem eine unklare Benennung von Navigationspunkten den Nutzern Probleme bereiten, auch wenn sie dem Entwickler der Webpräsenz sinnvoll erscheinen. Außerdem befürworteten die älteren Probanden bei der globalen und lokalen Navigation einen höheren Kontrast zwischen Schrift und Hintergrund. Dies bestätigt eine Aussage im Abschnitt „Inkonsequentes Farbschema“ (2.2.3), welche besagt, dass die Fähigkeit Kontraste wahrzunehmen mit steigendem Alter abnimmt. Zudem führte die inkorrekte Funktionsweise der lokalen Navigation zur Orientierungslosigkeit bei den Probanden. Dies bestätigte die, im Abschnitt „Kein einheitliches Layout“ (2.2.2) getroffene Aussage, nach der Nutzer sich auf der Website nicht mehr zurechtfinden, wenn Elemente, in dem Falle die Überschrift des Abschnitts, nicht an ihrer erwarteten Position zu finden sind.

Um die bestehenden Probleme mit der Website zu adressieren mussten also noch einige Änderungen an der Website vorgenommen werden. Zuerst wurde ein Hinweis eingefügt, welcher Nutzer mit deaktiviertem Javascript darauf hinweist, dass die Webpräsenz Javascript braucht, um korrekt zu funktionieren. Die Probleme mit der Navigation wurden behoben, indem der Schrift der einzelnen Navigationspunkte ein Textschatten hinzugefügt wurde, um so den Kontrast zu erhöhen. Außerdem wurden im HTML-Dokument leere Elemente vor den einzelnen lokalen Navigationspunkten platziert. Diesen leeren Elementen wurden dann im CSS Abstände hinzugefügt. So wird der obere Abstand des Textes zur globalen Navigation gewahrt, ohne dass ungewollte Leerräume zwischen den einzelnen Abschnitten entstehen. Die Abschnitte unter „Don’ts der Website Erstellung“ wurden umbenannt in „Konzeptionelle Verfehlungen“ und „Technische Fauxpas“, um ihren Inhalt besser widerzuspiegeln. Ebenfalls wurden die anderen Überschriften so bearbeitet, dass nun alle mit Großbuchstaben beginnen. Weiterhin wurde das CSS so bearbeitet, dass nur noch Bilder, welche auch tatsächlich eine Vergrößerung zulassen, durch eine entsprechende Änderung des Mauszeigers indiziert werden.

In der Zukunft wird die Website um zusätzliche Inhalte erweitert. Mit der Weiterentwicklung der Fähigkeiten des Autors wird sich auch die Website weiterentwickeln. Es werden neue Inhalte für fortgeschrittene Nutzer hinzufügt und die bestehenden Inhalte weiter überarbeitet und aktualisiert. Die gelernten Techniken werden nicht nur auf der Website beschrieben, sondern auch direkt angewandt, um die Website weiter zu verbessern. Außerdem wird nach einer geeigneten Methode gesucht, um die Website nachhaltig zu hosten und es werden Möglichkeiten analysiert, wie mit Hilfe von Suchmaschinen und Social Media die Bekanntheit der Webpräsenz gesteigert werden kann.

7 Quellen

[Fahl 2010] Fahl, Constantin, Die Bilder- und Nachrichtensuche im Internet: Urheber-, persönlichkeits- und wettbewerbsrechtliche Aspekte, V&R unipress GmbH, 2010

[CERN] http://home.cern/topics/birth-web, zuletzt besucht am 05.05.2017

[Raggett 1998] Raggett, Dave; Lam, Jenny; Alexander, Ian; Kmiec, Michael, Raggett on HTML 4, Addison Wesley Longman 1998

[WHATWG] https://whatwg.org, zuletzt besucht am 05.05.2017

[Rupf 2014] Rupf, Jannes, Web-Usability: Die benutzerfreundliche Gestaltung von Webseiten am Beispiel der Webseite der Baden-Württembergischen Übersetzertage 2013, Diplomica Verlag, 06.03.2014

[Nielsen 2006] Nielsen, Jakob; Loranger, Hoa, Prioritizing Web Usability, Pearson Education, 20.04.2006

[Burmester 2007] Burmester, Michael, Kompendium Medieninformatik - Medienpraxis, Springer-Verlag Berlin Heidelberg, 2007

[Balzert 2004] Balzert, Heide, Webdesign und Web-Ergonomie, W3l GmbH, 2004

[SelfHTML] https://wiki.selfhtml.org/wiki/SELFHTML:Verein/Protokolle/Gründungsprotokoll_2004, zuletzt besucht am 19.05.2017

[heise online] 28.01.2007, https://www.heise.de/newsticker/meldung/SelfHTML-Gruender-Muenz-kehrt-seinem-Projekt-den-Ruecken-139434.html, zuletzt besucht am 19.05.2017

[Münz 2006] Münz, Stefan, Professionelle Websites, Addison-Wesley Verlag, 2006

[Hochschule Anhalt] http://www.inf.hs-anhalt.de/fachbereich/kontakt/, zuletzt besucht am 19.05.2017

[Wikipedia] https://de.wikipedia.org/wiki/Wikipedia:Hauptseite, zuletzt besucht am 19.05.2017

[Zeit Online] http://www.zeit.de/index, zuletzt besucht am 19.05.2017

[Ewert 2001] Ewert, Birgit; Christoffer, Kerstin; Christoffer, Uwe; Ünlü, Saban, Web-Usability, Galileo Press GmbH, Bonn 2002 1. Auflage 2001

[Weniger Adobe Flash in Firefox 2016] Weniger Adobe Flash in Firefox, https://blog.mozilla.org/press-de/2016/07/20/weniger-adobe-flash-in-firefox, 20.06.2016, zuletzt aufgerufen am 02.04.2017

[LaForge 2016] LaForge, Anthony, Flash and Chrome, https://blog.google/products/chrome/flash-and-chrome, 09.08.2016, zuletzt aufgerufen am 02.04.2017

[Rohr 2015] Rohr, Matthias, Sicherheit von Webanwendungen in der Praxis: Wie sich Unternehmen schützen können – Hintergründe, Maßnahmen, Prüfverfahren und Prozesse, Springer-Verlag, 03.03.2015

[Krug 2014] Krug, Steve, Don't make me think: Web Usability: Das intuitive Web, MITP-Verlags GmbH & Co. KG, 19.11.2014

[Florin 2015] Florin, Alexander, User - Interface - Design: Usability in Web- und Software-Projekten, BoD – Books on Demand, 22.06.2015

[Bollwage 2005] Bollwage, Max, Typografie kompakt, Springer Berlin Heidelberg, 2005

[Hammer 2011] Hammer, Norbert; Bensmann, Karen, Webdesign für Studium und Beruf, Springer Berlin Heidelberg, 2011

[Böhringer 2014] Böhringer, Joachim; Bühler, Peter; Schlaich Patrick; Sinner, Dominik, Kompendium der Mediengestaltung, Springer Berlin Heidelberg, 2014

[Kalbach 2008] Kalbach, James, Handbuch der Webnavigation, O'Reilly Germany, 2008

[Stapelkamp 2007] Stapelkamp, Torsten, Screen- und Interfacedesign: Gestaltung und Usability für Hard- und Software, Springer Berlin Heidelberg, 2008

[Thesmann 2016] Thesmann, Stefan, Interface Design - Usability, User Experience und Accessibility im Web gestalten, 2., aktualisierte und erweiterte Auflage, Springer Fachmedien Wiesbaden, 2016

[CSS Selektoren] https://www.w3schools.com/cssref/css_selectors.asp, zuletzt aufgerufen am 10.06.2017

[W3C Arbeitsentwurf] https://www.w3.org/TR/2017/WD-html-aria-20170323/, zuletzt besucht am 10.06.2017

[Bootstrap Website] https://getbootstrap.com/, zuletzt aufgerufen am 10.06.2017

[Bewersdorff 2014] Bewersdorff, Jörg, Objektorientierte Programmierung mit JavaScript - Direktstart für Einsteiger, Springer Fachmedien Wiesbaden, 2014

[Walter 2008] Walter, Thomas, Kompendium der Web-Programmierung - PHP, Springer Berlin Heidelberg, 2008

8 Softwareverzeichnis

Chromium Browser Automation: Browser Automatisierung für den Chrome Browser. Weitere Informationen unter www.chrome-automation.com
IMacros: Browser Automatisierung für verschiedene Browser. Weitere Informationen unter www.imacros.net
Adobe Brackets: Entwicklungsumgebung von Adobe. Weitere Informationen unter www.brackets.io

9 Programmverzeichnis

Programm-Listings

10 Abbildungsverzeichnis

Abbildungsverzeichnis

Diese Arbeit wurde von mir selbständig verfasst und in gleicher oder ähnlicher Fassung noch nicht in einem anderen Studiengang als Prüfungsleistung vorgelegt. Ich habe keine anderen als die angegebenen Hilfsmittel und Quellen, einschließlich der angegebenen oder beschriebenen Software, verwendet.

Ort, Datum	Fabian Klautke