Fachhochschule Köln
Cologne University of Applied Sciences

Campus Gummersbach

Fakultät für Informatik und Ingenieurwissenschaften

Verbundstudiengang Wirtschaftsinformatik

Abschlussarbeit zur Erlangung des Diplomgrades

Diplom-Informatiker (FH)

in der Fachrichtung Informatik

Effiziente Implementierung von Responsive Web-Applikationen

Erstprüfer:	Prof. Dr. Heide Faeskorn-Woyke
Zweitprüfer:	Prof. Dr. Birgit Bertelsmeier
vorgelegt am:	03.02.2014
von cand.:	Bianca Schleich
aus:	Kaufmannstr. 53 53115 Bonn
Tel.:	0177/2748173
eMail:	bianca.schleich@gmx.de
Matrikel-Nr.:	11026074

Abstract (deutsch)

Ziel dieser Diplomarbeit ist es zu evaluieren, ob eine effiziente Implementierung von responsiven Webapplikationen zum Zeitpunkt der Erstellung der Arbeit möglich ist. Als technische Grundlage wird hierzu die HTML5-Spezifikation mit dem darin enthaltenen CSS3 und den JavaScript-Programmierschnittstellen herangezogen.

Es wird erläutert, dass unter responsivem Design die Reaktionsfähigkeit des Designs auf die Abrufumgebung, wie zum Beispiel die Größe der Anzeigefläche, zu verstehen ist und mit Hilfe welcher Techniken ein solches Design für Webapplikationen realisiert werden kann. Des Weiteren werden Möglichkeiten zur Performance-Optimierung aufgeführt, wobei festgestellt wird, dass für die Nutzung einer Webanwendung auf mobilen Geräten die Anzahl der Dateien das größte Potenzial zur Optimierung besitzt. Die Möglichkeiten der JavaScript-Programmierschnittstellen in HTML5 zur Umsetzung von Funktionalitäten für Webapplikationen, wie sie bei lokal installierten Anwendungen gebräuchlich sind, werden ebenso erläutert.

Das Fazit dieser Arbeit ist, dass ausreichend Techniken zur Erstellung von responsiven Webapplikationen in HTML5 definiert sind. Lediglich die zum Teil ausstehende Umsetzung dieser Techniken in den einzelnen Browsern verursacht Einschränkungen. Dies wirkt sich gegebenenfalls negativ auf die Effizienz des Umsetzungsprozesses aus. Ebenso kann die übermäßige Optimierung des Layouts und der Performance zu unverhältnismäßigem Aufwand führen.

Abstract (englisch)

The aim of this diploma thesis is to evaluate whether an efficient implementation of responsive web applications is possible at the time of creation of the thesis. As technical base the HTML5 specification containing also CSS3 and JavaScript programming interfaces is used.

It will be explained that Responsive Design means the responsiveness of the design regarding the execution environment, e.g. the size of the display. The techniques which can be used to realize such a design are described. Also options for performance optimization are listed. It becomes clear that the biggest opportunity for optimization of web applications that run on mobile devices is the number of files. The possibilities of JavaScript programming interfaces of HTML5 for implementing functionality for web applications as they are used for locally installed programs will also be explained.

The conclusion of this thesis is that HTML5 specifies sufficient techniques for creating a responsive web application. It is just the partial implementation of these features in the different browsers that causes limitations. This may negatively affect the efficiency of the implementation process. Also an excessive optimization of the layout or performance would result in a disproportional effort.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BLOB	Binary Large Object
CDN	Content Distribution Network/Content Delivery Network
CMS	Content Management System
CRM	Customer Relationship Management
CSS	Cascading Style Sheets
CSS3	CSS, Version 3
DNS	Domain Name System
DOM	Document Object Model
DSL	Digital Subscriber Line
GPRS	General Packet Radio Service
HTML	Hypertext Markup Language
HTML5	HTML, Version 5
HTTP	Hyper Text Transfer Protocol
ID	Identifier, eindeutiges Kennzeichen
IndexedDB	Indexed-Database
IT	Informationstechnologie
JSON	JavaScript Object Notation
kB	Kilobyte
MB	Megabyte
PC	Personal Computer
PHP	PHP Hypertext Preprocessor
RTT	Round Trip Time
RWA	Responsive Web Application
RWD	Responsive Webdesign
SEO	Search Engine Optimization
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WebGL	Web Graphics Library
WHATWG	Web Hypertext Application Technology Working Group
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

Einleitung

Die Informationstechnologie befindet sich permanent im Wandel. Die Weiterentwicklung der Technik bewirkt, dass Personen, die in diesem Bereich arbeiten, sich ebenfalls weiterentwickeln müssen. Betrachtet man die Entwicklung des World Wide Webs, so gibt es zurzeit zwei große Herausforderungen, mit denen sich Webentwickler befassen sollten. Zum einen ist dies die Erstellung von responsiven Webseiten und zum anderen die Implementierung von Webanwendungen mit einem Funktionsumfang, der dem von lokal installierten Anwendungen entspricht.

Responsives Webdesign

Seit der Einführung von Smartphones und der folgenden extrem schnellen Verbreitung dieser Geräte entsteht eine neue Art der Nutzung der Inhalte des Internets. Neben Desktop-Rechnern, von denen aus Informationen von Internetseiten abgerufen werden, existieren nun auch mobile Geräte, deren Anzeigefläche im Verhältnis zu der von Desktop-Monitoren deutlich begrenzt ist. Ein weiterer Unterschied durch den mobilen Abruf per Smartphone ist die im Vergleich mit der stationären Anbindung an das Internet langsamere Datenübertragung. Um Anwendern die Daten benutzerfreundlich zur Verfügung stellen zu können, müssen Webentwickler Methoden und Techniken entwickeln, die dieser Art der Nutzung des Internets gerecht werden. Das Layout der Webseiten sollte sich auf allen Displaygrößen so darstellen, dass die Informationen komfortabel zu erlangen sind. Kommt eine angemessene Übertragungszeit hinzu, kann man sich der Aufmerksamkeit des Benutzers sicher sein. Somit müssen Webseiten heutzutage auf ihre jeweilige Abrufumgebung angepasst reagieren können. Den Webentwicklern stellt sich die Aufgabe, reaktionsfähige Designs für eine Webseite zu entwerfen. Dies bezeichnet man als responsives Webdesign. Auch wird der feststehende englische Begriff „Responsive Webdesign“ verwendet.

Webapplikationen

Die Ausrichtung des Webs von der reinen Informationsvermittlung hin zur Informationsverarbeitung ist der zweite Punkt, dem Webentwickler Beachtung schenken müssen. Der Trend, Daten in der „Cloud“ zu speichern und diese Daten online verarbeiten und verbreiten zu können, erzeugt einen steigenden Bedarf an Webapplikationen. So stellt zum Beispiel Google seinen Anwendern sowohl online Speicherplatz als auch die Online-Office-Anwendungen „Drive“

kostenlos zur Verfügung. Webapplikationen bieten Vorteile bei der Entwicklung, da nicht für verschiedene Betriebssysteme entwickelt werden muss, und bei der Wartung, da immer nur eine Version auf dem Server aktiv ist und von den Anwendern genutzt wird. Viele Firmen sind von diesen und anderen Vorteilen der zentralen Datenhaltung und Anwendungsbereitstellung überzeugt und benötigen Webentwickler, die sich der Aufgabe Webapplikationen dieser Art zu implementieren, stellen.

Neben den zuvor beschriebenen Herausforderungen gibt es Punkte, die generell bei der Umsetzung von Applikationen oder Webseiten beachtet werden sollten. Ein effizientes Vorgehen ist für die Wirtschaftlichkeit der Umsetzung von Bedeutung und eine performante Implementierung spielt für die Akzeptanz der Anwendung beim Benutzer eine wichtige Rolle.

Effizienz

Bei der Beauftragung einer responsiven Webapplikation wird Webentwicklern von Auftraggebern indirekt eine weitere, nicht technische Herausforderung gestellt. Die Realisierung muss aus wirtschaftlichen Gründen möglichst effizient erfolgen. Um eine reaktionsfähige Webapplikation zu entwickeln, fällt sicherlich mehr Aufwand an als bei einer Webapplikation für eine fixe Bildschirmgröße. Die Vorgehensweise kann jedoch für die Erstellung von responsiven Webapplikationen optimiert werden, sodass der Mehraufwand möglichst gering gehalten werden kann.

Implementierung

Bei der Implementierung der Anwendung sollten Webentwickler die effizientesten zur Verfügung stehenden Techniken verwenden, um so eine Webapplikation mit möglichst hoher Performance zu erstellen. Eine besonders große Bedeutung besitzt eine gute Performance für die Nutzung der Anwendung per mobiler Datenübertragung, zum Beispiel auf einem Smartphone. Werden die Webapplikationen schnell geladen und arbeiten diese verzögerungsfrei, so dient dies der Benutzerfreundlichkeit und vor allem der Akzeptanz beim Anwender.

Betrachtet man die vier zuvor genannten Aspekte, so ergibt sich daraus das Thema dieser Arbeit:

Effiziente Implementierung von Responsive Web-Applikationen

Gliederung der Arbeit

Ziel dieser Arbeit ist es zu klären, ob eine effiziente Implementierung von responsiven Webanwendungen zum Zeitpunkt der Erstellung der Arbeit möglich ist. Hierzu soll betrachtet werden, was unter responsivem Design zu verstehen ist und mit Hilfe welcher Techniken dies realisiert werden kann. Die Möglichkeiten zur Performance-Optimierung und zur Erstellung von Webapplikationen sollen zusammengetragen und evaluiert werden.

Im ersten Kapitel dieser Arbeit wird daher zunächst geklärt, was „responsiv“ bedeutet. Durch einen Rückblick auf die Historie des Webdesigns und der Betrachtung, wie das Web mittlerweile genutzt wird, ergibt sich die Notwendigkeit für eine Anpassung des Webdesigns an die heutige Nutzung des Internets. Webseiten müssen reaktionsfähig bezüglich der Bedingungen sein, unter denen sie aufgerufen werden, und sich unter anderem der Anzeigefläche oder der Übertragungsgeschwindigkeit anpassen. Wie eine Reaktionsfähigkeit realisiert werden kann, wird anhand der ersten Ideen und Techniken der Pioniere des responsiven Webdesigns erläutert. Verwendet man bei der Erstellung von responsiven Webseiten ein Vorgehen analog dem für die Erstellung von Seiten für feste Monitorgrößen üblichen, so führt dies zu Problemen. Diese werden im letzten Abschnitt des ersten Kapitels erläutert.

Die Möglichkeiten für ein effizientes Vorgehen bei der Erstellung von responsiven Webseiten oder –anwendungen werden zu Anfang des zweiten Kapitels dargestellt. Darauf folgend werden die Verfahren und Techniken erläutert, die sich bis heute hinsichtlich des responsiven Designs entwickelt haben. Ein Hauptproblem für die Erstellung von Webseiten und -anwendungen ist die minimale Anzeigefläche der kleinen mobilen Geräte. Wie flexibel mit den Inhalten auf den unterschiedlichen Geräten umgegangen werden kann, wird in dem Kapitel ebenfalls beschrieben. Möglichkeiten werden aufgezeigt, wie Inhalte oder Navigationselemente ausgeblendet werden können. Eine semantische Auszeichnung der darzustellenden Elemente mit Hilfe von HTML5 hat den Vorteil, dass die Bedeutung der einzelnen Elemente für die Vermittlung der Inhalte der Seite gewichtet wird. Dies und weitere Vorteile werden im zweiten Kapitel ebenfalls erläutert. Auch wird beschrieben, warum keine festen, sondern flexible Raster für eine responsive Seite verwendet werden sollten. Umbruchpunkte werden benötigt,

wenn ein solches flexibles Raster ab einer bestimmten Anzeigefläche kein akzeptables Layout mehr liefert und ein anderes Layout zur Gestaltung der Seite definiert und herangezogen werden muss. Das zweite Kapitel schließt mit der Beschreibung der CSS Technik Mediaqueries ab. Mit ihr lässt sich die Auswahl der unterschiedlichen Gestaltungsmerkmale realisieren.

Der Performance-Optimierung wird sich im dritten Kapitel gewidmet. Es wird erläutert, warum die Reduzierung der Anzahl der Dateien und somit der benötigten HTTP-Requests von enormer Bedeutung ist, um die Zeit von der Anfrage bis zur ersten Darstellung einer Webseite oder –anwendung möglichst gering zu halten. Eine solche Performance-Optimierung wirkt sich besonders bei mobilen Datenübertragungstechniken positiv aus. Die verschiedenen Einsparmöglichkeiten bezüglich der Dateianzahl, aber auch des Dateivolumens werden im dritten Kapitel erläutert. Eine längere Vorhaltezeit der Dateien im Cache des Browsers verhindert, dass die Daten bei jedem Aufruf der Seite erneut übertragen werden müssen. Zügiges Anzeigen erster Inhalte kann auch durch die optimale Positionierung von JavaScript- und CSS-Aufrufen erreicht werden. Dies ist ebenso im dritten Kapitel beschrieben wie auch ein nachrangiges Laden von Daten, die für eine erste Anzeige nicht benötigt werden. Dadurch wird zusätzlich die Zeit bis zur ersten Anzeige der Seite optimiert. Mit welchen Tools und Internetseiten sich die Performance analysieren lässt, beschreibt der letzte Abschnitt des dritten Kapitels.

Im vierten Kapitel werden die Techniken dargestellt, die speziell zur Erstellung von Webapplikationen im HTML5-Standard, der auch JavaScript beinhaltet, definiert wurden. So gibt es seit HTML5 mehr Möglichkeiten Formulare zu gestalten. Mit nativem HTML der Version 5 lässt sich für einzelne Formularfelder eine Prüfung der Eingaben realisieren. Eine automatische Vervollständigung der Benutzereingabe oder Vorbelegung der Felder mit Inhalten ist möglich. Durch die Angabe von spezifischen Datentypen zu den Formularfeldern lassen sich auf Geräten mit eingeblendeter Tastatur auf den Datentyp angepasste Tastaturen oder Eingabemechanismen anzeigen, wie das vierte Kapitel beschreibt. Darauf folgend werden die JavaScript-Programmierschnittstellen zur Umsetzung von Webanwendungen erläutert. Diese wurden im HTML5-Standard festgelegt und sollen von den Browsern implementiert werden. Jedoch ist dies bisher nur teilweise geschehen, wie das vierte Kapitel zeigen wird. Diese APIs ermöglichen es, Funktionen, wie sie von

den bisherigen, lokal installierten Anwendungen bekannt sind, auch für Webanwendungen umzusetzen. So lässt sich mit den in HTML5 definierten Mitteln clientseitig zeichnen, um zum Beispiel Grafiken anzuzeigen. Auch das Verschieben von Elementen innerhalb einer Webseite oder aus Bereichen außerhalb des Browsers wie Dateien aus einem Explorer in die Seite hinein, lässt sich nun realisieren. Eine clientseitige Speicherung von Schlüssel-Wert-Paaren für die Dauer einer Session oder darüber hinaus ist ebenso möglich. Datenbanken für die Speicherung von Objekten unterschiedlicher Ausprägung lassen sich über eine in HTML5 definierte Programmierschnittstelle erstellen und manipulieren. Zudem wird im vierten Kapitel erläutert, wie Webanwendungen nutzbar gemacht werden können, wenn keine Datenverbindung vorhanden ist. Ein Zugriff auf einen Bereich des Dateisystems des aufrufenden Systems ist nun auch möglich. Verzeichnisse und Dateien können angelegt, gelesen und gelöscht werden. Des Weiteren ist ein Einlesen des Dateiinhalts umsetzbar. Für JavaScript existiert erstmalig die Möglichkeit, mehrere Threads zu erstellen. So können Funktionen, die für den aktiven Teil der Anwendung nicht relevant sind, in einem nachrangigen Thread abgearbeitet werden. Da das Aufführen aller in der Spezifikation des HTML5 enthaltenen APIs für diese Arbeit zu umfangreich wäre, folgt als letztes Kapitel eine kurze Übersicht über weitere APIs.

Generell werden in dieser Arbeit lediglich die Techniken aufgeführt, die mit HTML5 zu realisieren sind. Unter HTML5 versteht man jedoch nicht nur das reine HTML, sondern auch CSS3 und JavaScript. Somit werden im weiteren Verlauf Techniken und Methoden erläutert, die im Browser ausgeführt werden. Auf serverseitige Techniken und Einstellungen wird lediglich in Bezug auf die Performance-Optimierung, zum Beispiel für eine komprimierte Datenübertragung, eingegangen. Ebenso wird einige wenige Male auf kleine PHP-Skripte hingewiesen, da sie in den genannten Fällen die Performance-Optimierung für den Entwickler vereinfachen. Zusätzliche Arbeitshilfen wie CSS-Präprozessoren, -Frameworks und JavaScript-Bibliotheken sind in dieser Arbeit nicht berücksichtigt, da dies den Umfang übersteigen würde.

1 Responsive Webdesign

Responsive Webdesign (RWD) bedeutet, wenn auch stark vereinfacht, Webseiten reaktionsfähig zu gestalten.

„Reaktionsfähig“ ist die wörtliche Übersetzung des englischen Begriffs „responsive“. Da „Responsive Web Design“ ein feststehender Begriff in der Informationstechnologie (IT) ist, wird im weiteren Verlauf dieser Arbeit das eingedeutschte Wort „responsiv“ verwendet, auch wenn dieses so nicht im deutschen Duden vorhanden ist.

Responsives Webdesign bedeutet somit Webseiten reaktionsfähig zu gestalten. Da stellt sich die Frage, worauf diese Webseiten reagieren sollen? Oft wird diesbezüglich nur das Layout einer Webseite, das flexibel auf verschiedene Bildschirmgrößen und –ausrichtungen reagieren soll, betrachtet. Es gehört jedoch weit mehr zu einem responsiven Webdesign, wie man an der nachfolgenden Definition von Design erkennen wird. Wenn einzelne Webseiten und Webapplikationen auch in Zukunft eine hohe Akzeptanz durch den Nutzer erfahren sollen, müssen diese nicht nur in der Gestaltung, sondern auch bezüglich der zu übertragenden Datenmenge und der technischen Möglichkeiten des aufrufenden Systems reagieren können.

Hierzu klären wir erst einmal den Begriff Design:

Im deutschen Sprachgebrauch wird allgemein unter Designen ein gestalterisch-kreativer Prozess verstanden. Im Englischen und Französischen ist es ähnlich. Design wird mit „Gestaltung“ oder „Entwurf“ übersetzt. Im angelsächsischen Bereich beinhaltet das Design auch den technischen bis hin zum konzeptionellen Anteil der Gestaltung.[1]

Für den Begriff „Responsive Webdesign“ trifft diese umfassende Definition von Design ebenfalls zu. Es geht nicht nur darum, ein optisches Layout, sondern auch eine technische Umsetzung zu entwerfen.

Von der folgenden, bei Wikipedia zu lesenden Beschreibung zu Design lässt

[1] *Vgl. Wikipedia - Design, 20.01.2014*

sich ableiten, warum Webdesign fast schon zwangsläufig responsiv sein muss:

„*Design orientiert sich am Menschen und seinen vielfältigen Bedürfnissen. Diese Bedürfnisse reichen von körperlichen und psychischen Bedürfnissen bis hin zu Anforderungen des menschlichen Verstands an die gegenständliche Umwelt. Design folgt dabei nicht allein selbst gesetzten Regeln und Intentionen, sondern muss sich vor allem mit den Interessen jener (…) Personen auseinandersetzen, denen das Design dienlich sein soll. Dadurch ist Design und sind die Entwürfe vor allem zweckorientiert.*“[2]

Überträgt man dies zum Beispiel auf Webseiten mit Präsentationscharakter, so müssen diese Seiten die Interessen des Nutzers bedienen und somit die gewünschten Informationen zur Verfügung stellen. Dies muss in einer Form geschehen, die es dem Benutzer mit seinen körperlichen und psychischen Fähigkeiten erlaubt, die Informationen aufzufinden und zu erkennen.

Somit wird deutlich, dass bei der Erstellung des Designs einer Webseite der Mensch als Benutzer der Seite im Vordergrund steht. Die technischen Hilfsmittel wie zum Beispiel die Geräte, die zum Aufrufen der Seite dienen, dürfen für den Nutzer keine Einschränkung der Erfüllung seiner Interessen verursachen. Die verschiedensten Anzeigeformate dieser Geräte, deren unterschiedliche Bedienmechanismen oder die Browser mit ihrem nicht einheitlichen Funktionsumfang stellen die Designer vor die große Herausforderung reaktionsfähige Webseiten zu gestalten, die möglichst alle diese Bedingungen berücksichtigen. Hinzu kommt, dass nicht alle Menschen mit den gleichen Fähigkeiten ausgestattet sind. Auch sehbehinderte oder blinde Personen sollten ihren Bedarf an Informationen auf Webseiten stillen können. Barrierefreie oder zumindest barrierearme Seiten können mit Techniken, die im Rahmen des responsiven Webdesigns zur Verfügung stehen, erstellt werden.

Zunächst jedoch folgen Unterkapitel, in denen beschrieben wird, wie Webseiten bisher designt wurden, welche Probleme mit diesen Internetpräsenzen oder -anwendungen heutzutage konkret auftreten und wie zukunftsträchtige, responsive Designs erstellt werden können.

---

[2] *Wikipedia - Design, 20.01.2014*

1.1 Die Historie des Webdesigns

Beim Designen von Webseiten greift man bisher auf Techniken zurück, die von der Gestaltung der Printmedien übernommen wurden. Grafikdesigner teilen die für den Druck zu gestaltende Fläche in Raster ein, die als Hilfslinien für die Anordnung von grafischen und textuellen Elementen dienen. Innerhalb dieser Raster werden dann zum Beispiel Bilder und Texte angeordnet. Werden Designs für den Druck entworfen, so steht die Größe der zur Verfügung stehenden Fläche in eindeutigen Einheiten fest.

Da sich ein Browser im Vollbildmodus der Größe des Displays anpasst, steht die Größe der Anzeigefläche einer Webseite nicht fest. Die Ursprünge des Webdesigns stammen von der Gestaltung bei den Printmedien ab. Von dort wurde die Einteilung der Fläche in Raster übernommen. Als nutzbare Fläche definiert man einfach die zu dem Zeitpunkt der Erstellung des Designs gängige Monitorgröße. So hat sich in den letzten Jahren eine statische Breite von 960 Pixeln etabliert, da diese Breite von fast allen üblichen Monitoren abgedeckt wird. Die Höhe der Webseite ist abhängig von den auf der Seite anzuzeigenden Daten oder gegebenenfalls auch von der Standardhöhe der Monitore, wenn ein Scrollen auf der Seite unerwünscht ist. Diese feststehende Fläche kann so von Designern in Grafikprogrammen, wie zum Beispiel Adobe Photoshop, analog dem Vorgehen bei Printmedien entworfen werden. Dem Auftraggeber kommt dieses Verfahren entgegen. Er kann sich das fertige Design vor Augen führen und abnehmen. Lediglich die interaktiven Elemente können in einer statischen Grafik nicht dargestellt werden.

Meist zeitlich auf die Abnahme des Designs folgend beginnt der Entwickler mit der Umsetzung der Webseite. Da das Design vom Kunden abgenommen wurde, muss der Entwickler es soweit möglich „millimetergenau“ umsetzen. Hierzu war es bei der Implementierung lange Zeit gebräuchlich, Hypertext Markup Language (HTML) - Tabellen mit exakten Pixelangaben für die Größe zu verwenden. Diese Tabellen spiegelten das Hilfsraster des Designs wider und wurden auch in der Umsetzung zur Anordnung der Elemente genutzt.

Nachdem Cascading Style Sheets (CSS) entwickelt, vom World Wide Web Consortium (W3C) als Standard definiert und von vielen Browsern implementiert wurden, begannen die Webentwickler die Designs mit Hilfe von CSS umzusetzen. Im CSS wird die Darstellung einzelner HTML-Elemente, wie zum Beispiel die Größe und die Farbe, festgelegt. Das HTML kann somit

weitestgehend frei von Design-Informationen bleiben. Durch die Gestaltungsmöglichkeiten per CSS wurde die Aufteilung einer Webseite in Raster per HTML-Tabellen überflüssig.

Dennoch war das Umsetzen der auf 960 Pixeln basierenden Designs eine mühsame Arbeit. Darum entwickelten sich verschiedene Frameworks, die ein CSS-Grundgerüst für 960 Pixel breite Webseiten zur Verfügung stellen und die Arbeit der Entwickler vereinfachen. Das wahrscheinlich bekannteste Grid-Framework (Grid = englisch für Raster) ist das „960 Grid System“ von Nathan Smith.[3] Auf der Startseite des 960 Grid Systems werden verschiedene Beispielseiten, die das Framework verwendet haben, gezeigt. Über jeden vorhandenen Screenshot kann zur Verdeutlichung das Raster, welches für die Webseite Anwendung fand, eingeblendet werden (Abb. 1-1).

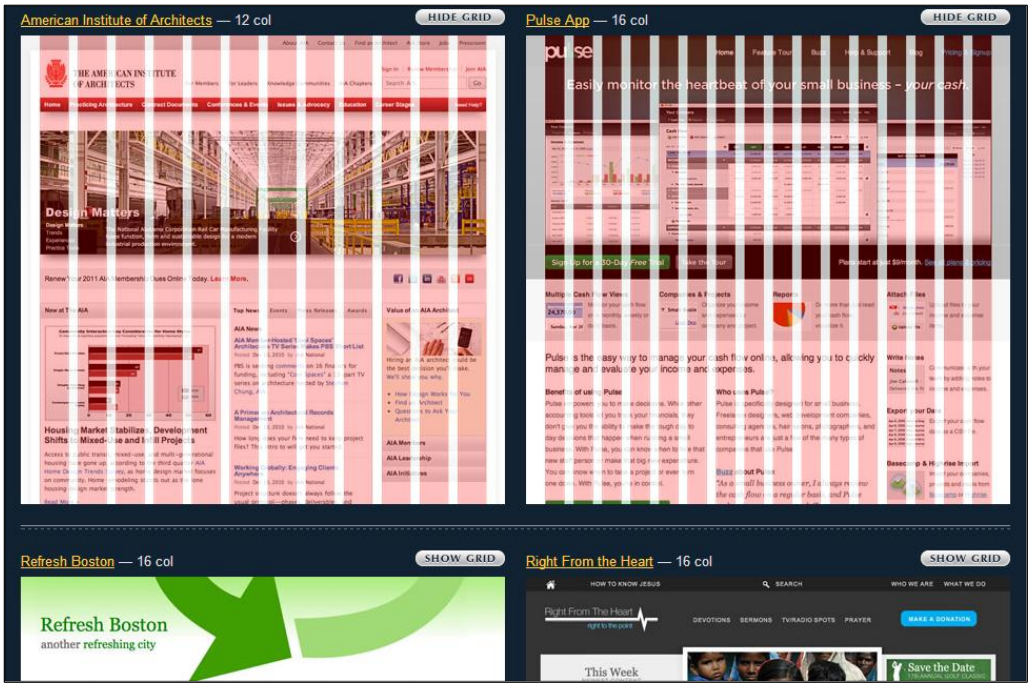

Abb. 1-1 960 Grid System – Beispielseite 12 und 16 spaltig

1.2 Die Entwicklung der Nutzung des Webs

Die größte Veränderung hinsichtlich der Nutzung des Internets in den letzten Jahren ist, dass der Nutzer einer Webseite oder Webapplikation nicht mehr zwangsläufig zu Hause oder auf der Arbeit an einem Schreibtisch vor einem

[3] *Vgl. Smith, 960 Grid System, 20.01.2014*

Monitor sitzt. Seit der Einführung von Smartphones und vor allem von Smartphones mit Touch-Bedienoberfläche hat sich das Bild des typischen Internet-Nutzers gewandelt.

Oft versuchen Webdesigner und Entwickler den kleinen Displays der Smartphones Rechnung zu tragen, indem sie spezielle Internet-Seiten für diese, im Verhältnis zu den bisher üblichen Dimensionen winzigen Geräte erstellen. Es wird parallel zu den für Desktops entworfenen Seiten ein zweites Design angefertigt und gegebenenfalls der anzuzeigende Inhalt angepasst. Die Umsetzung wird auf einer Unterdomain veröffentlicht. Ruft man eine Internetseite, für die eine „mobile“ Version besteht, mit dem Smartphone auf, so wird man meist automatisch auf die speziell entwickelte Webseite weitergeleitet. Dieses Vorgehen hat einige Nachteile. So entsteht doppelter Entwicklungsaufwand und sowohl technisch als auch inhaltlich müssen zwei Seiten gepflegt werden. Bei der Vielzahl der heutigen Geräte darf zudem bezweifelt werden, dass eine dieser beiden Seiten immer eine akzeptable Darstellung auf dem aufrufenden Gerät bietet und somit den Bedürfnissen des Nutzers gerecht wird.

Folgende Liste führt die heutzutage gängigsten Geräte zur Anzeige von Webseiten auf:

- Desktop-PCs
- Laptops
- Tablets
- Smartphones
- Fernsehgeräte
- Spielekonsolen

Durch die unterschiedlichen Gerätekategorien ergeben sich für Webseiten noch vielfältigere Systemumgebungen:

- Ausgabe per Anzeige auf Geräten in unterschiedlichsten Größen und mit verschiedensten Auflösungen oder Ausgabe per Sprache
- Bedienung per Maus, Trackpoint, Touchpad, Touch-Bedienoberfläche, TV-Fernbedienung, Spielekonsolencontroller, Gestik oder Sprache
- Texteingabe per Tastatur, Touch-Bedienoberfläche mit eingeblendeter Tastatur, TV-Fernbedienung, Spielekonsolencontroller oder Sprache

- Internetzugang, der sich gegebenenfalls auf die Bandbreite auswirkt, per „Festnetz“, Kabelfernsehnetz, Mobilfunknetze der unterschiedlichsten Kategorien, Satellit oder öffentliches WLAN
- Rechenleistung der Geräte
- Funktionsumfang der verwendeten Browser, zum Beispiel JavaScript, Mediaqueries

Aufgrund der beiden Listen wird deutlich, dass ein statisches Webdesign für ein Ausgabemedium, das mindestens 960 Pixel Breite besitzt, heute in keiner Weise mehr zeitgemäß ist und beim Internetnutzer wenig Akzeptanz finden wird.

Im Jahr 2013 verwenden ca. 18 Prozent der Internetnutzer schon ein Display, das weniger als 960 Pixel breit ist. Darunter fallen allerdings auch ca. 7 Prozent Tablet-Nutzer mit einer Auflösung von 768 x 1024 Pixeln, die ihr Gerät vertikal ausgerichtet nutzen und prinzipiell auf eine horizontale, 960 Pixel abdeckende Anzeige wechseln könnten.[4]

Oft wird aufgrund der Geräte wie Smartphones oder Tablets vom „Mobilen Internet“ gesprochen. Eine „mobile“ Nutzung des Internets wird auch mit einem kleinen Display, einer geringeren Datenbandbreite und einem kurz auf der Seite verweilenden Nutzer, der schnell nur etwas nachsieht und sein Gerät per Berührung bedient, verbunden. Viele Menschen benutzen diese kleinen Geräte jedoch zu Hause per WLAN über ihre schnelle Festnetz-Internetanbindung, um bequem von der Couch aus ausgiebig Informationen aus dem Internet abzurufen oder Online-Applikationen zu benutzen. Der Begriff „Mobiles Internet“ verleitet dazu, viele Gegebenheiten bezüglich der Internetnutzung in die Systemumgebung hinein zu interpretieren, obwohl diese nicht zutreffend sein müssen. Deswegen sollte der Begriff nicht dazu verwendet werden, die Art der Nutzung des Internets bezüglich Größe der Anzeige (englisch: display), der Geschwindigkeit der Internetanbindung, des Aufenthaltsorts und der „Ausdauer“ des Nutzers zu beschreiben. Denn eigentlich gibt es nur ein Internet, das von verschiedenen Geräten, über vielartige Datenverbindungen und von Nutzern an unterschiedlichsten Orten aufgerufen wird.

[4] *Vgl. Siegel, My Webcheck – Seitenbreite von Websites 2013, 20.01.2014*

Blicken wir jetzt noch einmal auf die einleitende Definition von Design zurück, stellen wir fest, dass statische Webseiten nicht die Interessen des Benutzers bedienen. Er möchte heutzutage sein Bedürfnis Informationen zu erhalten und Internet-Geschäfte zu tätigen auch mit anderen Geräten als dem PC befriedigen und dabei nicht durch die Wahl des Gerätes an die Grenzen seiner körperlichen und psychischen Leistungsfähigkeit geführt werden.

1.3 Reaktionsfähige Webseiten

Schon im Jahr 2000 schrieb John Allsopp, der sich intensiv mit Webdesign und Webdevelopment beschäftigt, einen auf alistapart.com veröffentlichten Artikel namens „A Dao of Web Design“. Er sagt, dass das Web prinzipiell flexibel ist und diese Eigenschaft bei der Umsetzung von Webseiten genutzt werden soll, um diese flexibel und dadurch zugänglich für jeden zu gestalten:

„*It is the nature of the web to be flexible, and it should be our role as designers and developers to embrace this flexibility, and produce pages which, by being flexible, are accessible to all.*“[5]

John Allsopp sagt weiter, dass dies erreicht werden kann, indem man sich beim Entwerfen einer Webseite nicht erst Gedanken um die Optik macht, sondern sich auf den Inhalt und den Service, den man dem Nutzer bieten möchte, konzentriert. Auf technischer Seite schlägt er vor, HTML ausschließlich zur Strukturierung des Inhaltes zu nutzen und das Design per CSS zu gestalten. Er fordert auf, die vom Nutzer im Browser eingestellte Schriftgröße zu akzeptieren und einen prozentualen Wert hiervon für die Angabe der Größe von Überschriften, des Abstandes vom Text und anderen Elementen vom Rand und für Einrückungen zu verwenden. Die Größe einer Webseite und deren Layout soll ebenfalls nur in Prozentwerten definiert werden, sodass sich die Seiten an verschiedene Geräte und Auflösungen anpassen können. All diese Gedanken entwickelte er im Jahr 2000 und das, obwohl die Bandbreite der Größe der Betrachtungsgeräte für Webseiten noch sehr gering war. Webseiten wurden bis dahin auf Desktop-Monitoren angezeigt, Smartphones gab es noch nicht.

[5] *Allsopp, A List Apart - A Dao of Web Design, 20.01.2014*

Erst im Jahr 2009 beschreibt Ethan Marcotte in dem ebenfalls bei alistapart.com erschienenen Artikel „Fluid Grids“, wie er sich für einen Kunden, der eine fließende, sich der Browsergröße auf dem Monitor anpassende Webseite beauftragt, Gedanken um eine mögliche Umsetzung macht.[6] Dabei ist er sicherlich auf den Aufsatz „A Dao of Web Design“ gestoßen, denn die technische Lösung, die er aufführt, spiegelt die Grundidee von John Allsopp wider. Alle verwendeten Schriftgrößen definiert Ethan Marcotte prozentual von der im Browser eingestellten Standardgröße. Entsprechend verfährt er mit der Breitenangabe für die verwendeten Elemente und zeigt, wie man aus Pixelangaben eines Designs prozentuale Werte für ein flüssiges Layout berechnet.

Schon ein Jahr später (2010) veröffentlichte Ethan Marcotte einen weiteren Artikel auf der Internetseite „A List Apart“. Er merkt an, dass immer mehr Firmen eine „iPhone website“ beauftragen.[7] Dieser Bedarf ist auf den Start des Verkaufs des ersten Smartphones mit Touch-Bedienoberfläche, dem iPhone, 2007 zurückzuführen. Die Verkaufszahlen stiegen im Juli 2008 mit Verkaufsstart der Nachfolgegeneration, dem iPhone 3G, drastisch an.[8] Zu diesem Zeitpunkt wurden auch weitere Smartphones mit den Betriebssystemen der Hersteller Symbian, RIM (heute BlackBerry), Android und andere angeboten, sodass diese Gerätekategorie immer größere Verbreitung fand. Ethan Marcotte stellt bei seiner Arbeit fest, dass die Verwendung von „Fluid Grids“ auf sehr kleinen Geräten, aber auch auf Breitbildschirmen an seine Grenzen stößt. Auch das Entwickeln separater Webseiten für Smartphones betrachtet er nicht als zukunftsweisende Lösung, da die Anzahl der unterschiedlichen Geräte zur Nutzung des Internets immer weiter steigen wird. Durch einen Begriff aus der Architektur inspiriert, macht er darauf aufmerksam, dass in Zukunft responsives Webdesign betrieben werden muss. Somit überschrieb er einen Artikel bei alistapart.com mit dem Titel „Responsive Web Design“ und prägt damit diesen mittlerweile feststehenden Begriff.[9]

---

[6] *Vgl. Marcotte, A List Apart - Fluid Grids, 20.01.2014*

[7] *Vgl. Marcotte, A List Apart – Responsive Web Design, 20.01.2014*

[8] *Vgl. Wikipedia - Apple iPhone, 20.01.2014*

[9] *Vgl. Marcotte, A List Apart - Responsive Web Design, 20.01.2014*

Ethan Marcotte nennt als Zutaten für „Responsive Web Design“ folgende drei Techniken, zu denen bis heute allerdings noch weitere hinzugekommen sind:

- Raster mit prozentual angegebener Größe (Fluid Grids)
- Mediaqueries
- Flexible Bilder

Die mit prozentualen Größen definierten Raster wurden zuvor schon erläutert. Mediaqueries sind Teil von CSS und wurden mit der Version 2 eingeführt. Es können verschiedene Medientypen direkt im Browser abgefragt werden. Häufig verwendet wird die Abfrage auf „screen“ und „print“, worauf mit unterschiedlichen Styles reagiert werden kann. Mit CSS3 ist es möglich, zusätzlich noch ein Gerät auf eine Mindest- oder Höchstbreite oder –höhe zu prüfen. Auch andere Abfragen sind möglich und werden im weiteren Verlauf noch beschrieben. Unter flexiblen Bildern versteht Ethan Marcotte, ähnlich wie bei der Schriftgröße, dass auch Bilder sich anpassen und dazu mit prozentualen Größen versehen werden. Auch die Anordnung kann je nach Bildschirmgröße angepasst werden.

Die Möglichkeit, das Medium und die Anzeigegröße abfragen zu können, sollte nicht genutzt werden, um für spezielle, weit verbreitete Geräte oder für jede Gerätekategorie ein separates Layout zu definieren. Im Sinne von responsivem Webdesign sollte dies verwendet werden, um bei für das optische Design kritischen Größen auf ein anderes in CSS definiertes Design umzuspringen. Webdesign muss heutzutage den Anspruch haben, flexibel auf die Umgebung reagieren zu können. Die Reaktionsfähigkeit der heute erstellten Seiten sollte auch in Zukunft gegeben sein. Anpassungen auf die Maße von Geräten, die zum Zeitpunkt der Erstellung existieren, werden nicht nachhaltig sein.

Im Internet liest man des Öfteren den Begriff „Adaptive Webdesign“. Das Design ist hierbei nicht ganz so flexibel wie beim „Responsive Webdesign“. „Adaptive Webdesign“ bedeutet anpassungsfähiges Design. Es werden nicht die fließenden Raster mit prozentualen Größen zur Gestaltung verwendet, sondern lediglich, abhängig von der per Mediaqueries geprüften Systemumgebung, unterschiedliche CSS mit darin jeweils fest in Pixel definierte Größen herangezogen. Dadurch entsteht ein sich in Stufen veränderndes Raster beziehungsweise Layout.

In den folgenden Kapiteln werden die von Ethan Marcotte verwendeten Techniken und weitere Verfahren aufgezeigt, die nach der Veröffentlichung seines Artikels zur Erfüllung des Grundgedankens von responsivem Design von Webdesignern und Webentwicklern zusammengetragen und neu entwickelt wurden.

Zu diesen Techniken und Verfahren zählen die hier zunächst kurz erwähnten und im Verlauf ausführlicher behandelten Punkte.

Progressive Enhancement

Die Grundidee des „Progressive Enhancement“ ist es, Webseiten so zu gestalten, dass jeder Besucher, unabhängig von seiner technischen Ausstattung, alle grundlegenden Inhalte und Funktionen abrufen kann. Wird auf einer Seite zum Beispiel JavaScript verwendet, so muss der Inhalt dennoch für Anwender ohne JavaScript erreichbar bleiben. Wird der Inhalt semantisch per HTML ausgezeichnet, so können zum Beispiel auch Screenreader die Inhalte dem darauf angewiesenen Benutzer zugänglich machen. Das HTML enthält den Inhalt einer Seite. JavaScript und CSS sollen getrennt davon in eigenen Dateien gespeichert werden.

Flexible Inhalte

Inhalte können flexibel gestaltet werden, indem unterschiedliche Inhalte für verschieden große Display-Größen angezeigt werden. Besser ist es jedoch, Inhalte auf kleinen Geräten zunächst auszublenden und per Benutzerinteraktion sichtbar zu machen. Hierzu existieren Verfahren wie „Off Canvas“, „Akkordeons“ und „Karussells“.

Responsive Navigation

Da ein Navigationsmenü auf Smartphones leicht große Teile des Displays einnehmen kann, würde der eigentliche Inhalt der Seite untergehen, vor allem, wenn die Navigation im Kopf der Startseite untergebracht ist. Darum haben sich Verfahren entwickelt, mit denen das Menü auf unterschiedliche Art und Weise eingeblendet werden kann. Meist ist im Kopf der Webseite nur ein Schalter zum Einblenden des Menüs vorhanden, wodurch viel Platz für den Inhalt der Seiten zur Verfügung steht.

Semantisches HTML

Um den Inhalt einer HTML-Seite reaktionsfähig zu gestalten, ist es notwendig, die Bedeutung und den Inhalt einzelner Bereiche beurteilen zu können. Nur wenn bekannt ist, was die Überschrift der Seite, was der Hauptinhalt und was zusätzliche, für das Verständnis nicht notwendige Informationen sind, ist es möglich, das Layout einer Seite responsiv an die unterschiedlichen Geräte anzupassen. HTML in der Version 5 (HTML5) bietet hierzu Auszeichnungselemente wie „header“, „main“, „nav“ und „aside“.

Flexible Raster (englisch: Fluid Grids)

Um eine Anpassung des Layouts an die zur Verfügung stehende Anzeigefläche zu erzielen, müssen die Breitenangaben für eine responsive Webseite beziehungsweise Webanwendung prozentual angegeben werden. „Cascading Style Sheets“ in der Version 3 (CSS3) bietet hierfür mit dem Boxmodell „border-box“ eine komfortable Möglichkeit.

Umbruchpunkte

Selbst bei der Verwendung von flexiblen Rastern gibt es Grenzen hinsichtlich der Flexibilität des Layouts. An bestimmten Punkten, zum Beispiel ab einer bestimmten Breite, kann ein Layout nicht mehr korrekt oder benutzerfreundlich angezeigt werden. An solchen Grenzen kann per Mediaquery auf ein anderes Layout, definiert per CSS, gewechselt werden. Diese Grenzen werden Umbruchpunkte genannt.

Mediaqueries

Mediaqueries werden im Rahmen der Entwicklung einer responsiven Webseite hauptsächlich benötigt, um zwischen verschiedenen Layouts, abhängig von der Größe der Anzeigefläche, wechseln zu können. So ermöglichen Mediaqueries die Abfrage, ob ein Display kleiner oder größer einer bestimmten Pixel-Anzahl ist. Die Abfrage kann im CSS-Code wie folgt aussehen:

```
@media only screen and (max-width:799px) {
  /*CSS Befehle*/
}
```

Performance-Optimierung

Eine gute Performance ist vor allem für die mobile Nutzung des Internets von großer Bedeutung. Durch geringere Datenübertragungsraten und weitere Performance-Nachteile ist eine Optimierung wichtig, um eine Akzeptanz beim Anwender zu erreichen. Hauptfaktoren, die zur Optimierung der Ladezeit berücksichtigt werden sollten, sind die Anzahl und die Größe der zu ladenden Dateien. Eine geringe Anzahl an Dateien kann für CSS, JavaScript und Bilder durch das Zusammenfügen von einzelnen Dateien in eine Datei erreicht werden. Auch die Größe der Dateien selbst, egal ob es Bild-, CSS- oder JavaScript-Dateien sind, sollte optimiert werden. Eine komprimierte Übertragung bringt weitere Vorteile für die Performance

1.4 Probleme eines linearen Vorgehensmodells

In der Software-Entwicklung, wie bei der Erstellung von Webseiten, wird meist eine lineare Vorgehensweise für den gesamten Entwicklungszyklus verwendet. Zunächst werden die Vorgaben vom Auftraggeber analysiert und an die umsetzende Firma beziehungsweise Abteilung weiter gegeben. Diese erstellt eine Planung, zum Beispiel in Form eines Lastenheftes. Anschließend wird ein Software-Entwurf erstellt und danach die Applikation implementiert. Nach all diesen einzelnen Phasen erfolgt in der Regel eine Abnahme des Ergebnisses durch den Auftraggeber.

Ein solches Vorgehen, kombiniert mit der Erstellung des statischen Designs in einem Grafikprogramm, führt bei der Entwicklung von responsiven Webseiten zu Problemen. Wird der Entwurf einer Weboberfläche dem Auftraggeber in Form eines Design-Dokuments vorgelegt, so wird ihm in den darin enthaltenen Bildern beziehungsweise Grafiken ein statisches Bild einer Webseite vermittelt. Wird auf eine dynamische Anpassung an die Bildschirmgröße im Text hingewiesen, so bleibt es der Vorstellungskraft des Lesers überlassen, wie sich die Anpassung darstellt. Dass diese Vorstellung jedoch der später umgesetzten Realität entspricht, ist eher unwahrscheinlich. Wie sich das Design zum Beispiel auf kleineren Geräten ändert, kann theoretisch mit weiteren Bildern in dem Dokument verdeutlicht werden. Aber damit sind gegebenenfalls noch nicht alle möglichen Layouts einer responsiven Webseite abgedeckt. Nimmt der Kunde ein solches Design ab und wird die Seite anschließend implementiert, so kann es sein, dass der

Auftraggeber mit der fertigen Webseite nicht zufrieden ist, da sich andere Layouts als im Dokument beschrieben, ergeben.

Eine von der Planung abweichende inhaltliche Gewichtung einzelner Bereiche der Seite kann beispielsweise durch ein sich veränderndes Größenverhältnis des Textbereiches zum angezeigten Bild entstehen und somit ein fachliches Problem darstellen. Eine Überarbeitung, die in einem späten Stadium des Entwicklungsprozesses hohen Aufwand bedeutet, würde notwendig. Ähnliche Probleme können bezüglich animierter oder interaktiver Elemente entstehen, da ihre Aktionen in einem Dokument zwar beschrieben, aber nicht visuell verdeutlicht werden können.

Ein Nachteil dieses Vorgehens ist auch, dass auf den Grafiken das Design des kompletten Inhalts einer einzelnen Seite visualisiert wird. Ist dieser Inhalt jedoch zu mächtig für eine Bildschirmhöhe, egal ob auf einem Desktop-Monitor oder einem Tablet, so wird Scrollen notwendig. Auf einen Blick ist auf diesen Geräten nur ein Teil des Layouts sichtbar. Dies kann gegebenenfalls die Wirkung des in der Designgrafik ansprechenden Layouts auf den Anzeigegeräten beeinträchtigen. Ein anderes Problem sind Textbereiche, deren Größe auf unterschiedliche Displaygrößen reagiert. Wird auf einem Monitor eine Box mit dem anzuzeigenden Text komplett ausgefüllt, so kann es sein, dass auf Fernsehern die Box sehr breit wird und der Text nur noch wenige Zeilen in der Box einnimmt und damit Leerraum im unteren Bereich entsteht. Dies wird nicht in einem statischen Design erkennbar. Dasselbe Problem kann bei dynamischen Texten auftreten. Werden lange und kurze Texte für entsprechende Bildschirmgrößen verwendet, kann auch ungeplanter Leerraum verursacht werden. Zusätzlich steht der tatsächlich anzuzeigende Text zum Zeitpunkt des Designs oft noch nicht fest, da der Auftraggeber sich darüber noch nicht im Klaren ist. So kann es sein, dass ein Design entwickelt wird, das nicht zum Inhalt, beziehungsweise zum Umfang des Inhalts, passt.

Eine weitere Gefahr für das Projekt sind technische Hindernisse. Es kann vorkommen, dass ein im Grafikprogramm entwickeltes Design technisch nicht umgesetzt werden kann, da notwendige Techniken zum Beispiel in bestimmten Browsern nicht funktionsfähig sind.

Dies alles bedeutet, dass ein lineares Vorgehen bei der Entwicklung von responsiven Webseiten, beziehungsweise Webanwendungen, zu erhöhtem

Überarbeitungsaufwand und somit zu höheren Kosten führt und nicht effektiv ist.

Viele dieser möglichen Probleme treten so oder ähnlich auch bei der Entwicklung von nicht responsiven Webseiten auf. Dies zeigt, dass nicht nur die Art der technischen Umsetzung von Webseiten, sondern das gesamte Vorgehen einer Überarbeitung bedarf. Besonders trifft dies beim responsiven Webdesign zu. Vielleicht war es falsch oder einfach nur die bequemste Möglichkeit, das Verfahren zur Entwicklung von Designs eins zu eins von den Printmedien auf das Webdesign zu übertragen. Alle Phasen des Entwicklungsprozesses müssen für die Erstellung von responsiven Webseiten überdacht und überarbeitet werden, inklusive dem Vorgehensmodell, in dem die Phasen eingebettet sind. Ein lineares Vorgehen ist, wie oben gezeigt, nicht optimal zur Entwicklung von Webseiten und erst recht nicht effizient für die Entwicklung von reaktionsfähigen Webseiten.

2 Grundlagen des Responsive Webdesign

2.1 Effiziente Vorgehensweise

Zur effizienten Erstellung und Umsetzung eines responsiven Webdesigns ist eine dynamische Vorgehensweise im Projekt notwendig. Um die oben beschriebenen Probleme eines linearen Vorgehensmodells zu minimieren oder bestenfalls auszuschließen, geht man dazu über, Projekte nicht fest in die üblichen Phasen einzuteilen, in denen das abgenommene Ergebnis einer Phase die Grundlage der nachfolgenden Phase darstellt. Der Auftraggeber und die realisierenden Personen müssen von Beginn an zusammenarbeiten, damit in keinem Arbeitsschritt Dinge festgelegt werden, die entweder nicht mit den Vorstellungen des Auftraggebers vereinbar sind oder bei der Umsetzung der Reaktionsfähigkeit zu Problemen führen. Differenzen müssen frühestmöglich auffallen, damit nicht in eine falsche Richtung weitergearbeitet wird und dieser Aufwand somit vergebens wäre.

Kerstin Hahlbohm, eine angesehene Art-Direktorin, beschreibt in einem im Web veröffentlichten Artikel, dass der Arbeitsprozess für ein effektives Responsive Design Projekt interdisziplinär gestaltet werden müsse. Alle Personen, die an Konzept, Design und Programmierung arbeiten, müssten von Beginn an ihre Ergebnisse zusammen erarbeiten und besprechen.[10] Ähnlich sieht dies auch Christoph Zillgens in seinem Buch „Responsive Webdesign“:

„*Eine strikte Rollentrennung ist im reaktionsfähigen Kontext zu langwierig.*“[11]

2.1.1 Planung mit Hilfe von Content First

Zu Beginn der Planung einer responsiven Webseite ist es wichtig, dass die Inhalte der Seite festgelegt werden. Dies klingt zunächst logisch, jedoch kommt es häufig vor, dass beim Start des Projektes zunächst das Design in den Vordergrund gestellt wird. Ursache ist meist, dass beim Auftraggeber

[10] *Vgl. Hahlbohm, Responsive Webdesign Workflow – Sechs Praxistipps, 20.01.2014*

[11] *Zillgens, Responsive Webdesign, Seite 71*

noch nicht beschlossen wurde, welche Inhalte und Funktionen vorhanden sein sollen.

Für reaktionsfähige Webseiten oder –anwendungen ist es jedoch wichtig, dass die zu verwendenden Texte und Grafikelemente zu Beginn des Projektes feststehen. Eine Fokussierung auf die Inhalte bewirkt, dass man nicht in Versuchung kommt, dem Layout der Seite mehr Gewichtung zu erteilen als dem Inhalt.

Jeffrey Zeldman, ebenfalls ein „A List Apart"-Autor, bezeichnete Design ohne Inhalt sogar als Dekoration:

„Content precedes design. Design in the absence of content is not design, it's decoration."[12]

Eine Webseite mit gutem responsiven Design soll, wie schon zu Beginn dieser Arbeit festgestellt wurde, dem Menschen dazu dienen, seine Bedürfnisse hinsichtlich der Seite zu befriedigen. Dies ist am ehesten zu erreichen, wenn die Planung von den Inhalten und Funktionen ausgeht, die dem Zweck beziehungsweise den geschäftlichen Anforderungen der Seite dienen. Ein solches Vorgehen trägt den Namen „Content First". Dabei werden die anzuzeigenden Inhalte vom Auftraggeber in einer nach Priorität sortierten Liste eingefügt. Durch die Priorisierung erhält man wichtige Hinweise, wie die Seite bezüglich des Layouts für unterschiedliche Anzeigeflächen zu gestalten ist. Inhalte am Ende der Liste können gegebenenfalls auf kleinen Geräten nicht angezeigt oder in nicht direkt sichtbaren Bereichen platziert werden.

2.1.2 Mobile First

Sind die Inhalte priorisiert und aufgelistet, kann man in die detailliertere Planung einsteigen. Dabei ist ein Vorgehen nach dem Prinzip „Mobile First" sinnvoll. Das heißt, man beginnt die Inhalte für ein Gerät mit begrenzter Systemumgebung, heutzutage meist Smartphones, zu planen. Die sortierte Liste stellt hierzu eine gute Vorarbeit dar, wobei das Prinzip „Content First" auch innerhalb „Mobile First" Anwendung findet. Denn auch bei „Mobile First" konzentriert man sich auf die Inhalte, die zur Erfüllung des Zwecks der

[12] *Zeldman, Twitter – Zeldman, 20.01.2014*

Webseite vorhanden sein müssen. Es werden die Mindestanforderungen an eine Seite herausgestellt, die trotz der Anzeige auf kleinen Displays oder einer geringen Datenübertragungsrate verfügbar sein müssen.
Wie anfangs schon erwähnt, darf man aus dem Begriff „mobil“ nicht zwangsläufig auf kleine Displays und langsame Internetanbindung schließen. Somit ist „Mobile First“ ein nicht ganz zutreffender Titel. Geht man jedoch vom limitiertesten Fall aus, so trifft die Annahme zu.

Geprägt wurde der Begriff „Mobile First“ durch einen Artikel des amerikanischen Webdesigners Luke Wroblewski im Jahr 2009.[13] In seinem gleichnamigen bei „A Book Apart“ erschienenen Buch schreibt er, dass bei der Entwicklung für mobile Geräte zunächst an Einschränkungen gedacht werde. Das Display sei klein, die Datenverbindung langsam und der Nutzer schenke der Seite nicht die volle Aufmerksamkeit. Weiter sagt er, dass dieses Vorgehen auch Vorteile bringe, da man durch kleine Bildschirmgrößen dazu gezwungen werde festzulegen, was wirklich wichtig für den Auftraggeber und sein Geschäft sei. Für Darüberhinausgehendes sei einfach kein Platz vorhanden. Langsame Datenverbindungen und geringer Speicherplatz verlangen ebenfalls, dass man auf die Effizienz (englisch: performance) achte. Ein Ergebnis seien elegante, überall schnell ladende Webseiten.[14]

Geht man also von der inhaltlichen Seite an die Planung einer Webseite heran und achtet darauf, nur die notwendigen Elemente und Texte zu verwenden, beziehungsweise diese hoch zu priorisieren, so hat dies auch Vorteile für eine größere Darstellung. Die Seiten erhalten eine klare, übersichtliche Struktur und sind ebenfalls auf den Zweck der Webseite fokussiert. Für die Planung vom unteren Ende der Bandbreite an spricht, dass Geräte wie Smartphones und Tablets immer häufiger auch für die kommerzielle Nutzung des Internets herangezogen werden. Hierdurch erlangt die korrekte Darstellung von Webseiten und eine komfortable Bedienbarkeit von Webapplikationen auf kleinen Geräten eine immer größer werdende wirtschaftliche Bedeutung. So stellt der Bundesverband des Deutschen Versandhandels e.V. (bvh) in einer Studie fest, dass im Jahr 2013 schon vierzig Prozent aller Smartphone-

[13] *Vgl. Wroblewski, Mobile First, 20.01.2014*

[14] *Vgl. Wroblewski, Mobile First, Seite 28*

Besitzer über diese Geräte online einkaufen.[15] Der Anwender geht dabei davon aus, dass Online-Shops oder -Applikationen auch auf kleinen Geräten funktionieren und bedienbar sind. Erfüllen sie diese Erwartung nicht, werden gegebenenfalls Geschäfte nicht getätigt und es geht möglicher Umsatz verloren. Somit ist es von großer Bedeutung, dass die Seiten auf kleinen Geräten nutzbar sind. Werden zum Beispiel Eingaben durch den Benutzer auf einer Webseite erwartet, so ist zu berücksichtigen, dass Smartphone- oder Tabletbenutzer keine vollständige oder zumindest keine im Zehn-Finger-System zu bedienende Tastatur besitzen. Die Eingabe von großen Mengen Text oder Sonderzeichen ist mühsam. Beachtet man dies, so werden keine unnötigen Eingaben vom Anwender verlangt. Davon profitieren auch die Nutzer eines Desktop-PCs oder Laptops. Es werden auch von ihnen nur die nötigen Eingaben gefordert.

Der Ansatz „Mobile First“ passt wunderbar zu einem weiteren Vorgehen bezüglich der Erstellung von reaktionsfähigen Webseiten, dem „Progressive Enhancement“. Hierbei geht man ebenfalls so vor, dass zunächst nur Funktionalitäten genutzt werden, die jedes Gerät und jeder Browser umsetzen kann. Darauf aufbauend werden Schritt für Schritt Techniken und Funktionalitäten ergänzt, die leistungsfähigere Geräte und Browser verarbeiten können. So wird eine simple Version der Webseite auf einfachen Geräten angezeigt. Besucht man die Seite mit der neuesten Technik, so werden zusätzliche Features genutzt. Mehr dazu wird in den folgenden Kapiteln erläutert.

Durch die Priorisierung von Inhalten kann es auf unterschiedlichen Anzeigeformaten oder unterschiedlichen Geräten dazu kommen, dass differenzierte Inhalte und Funktionen zur Verfügung stehen. Zum Beispiel kann es sein, dass ein ausschweifender Text auf Smartphones nicht angezeigt werden soll. Oder der Text wird auf kleinen Geräten zunächst nicht komplett, sondern nur der Beginn des Textes angezeigt. Erst durch einen Link oder durch einen aufklappbaren Bereich wird der gesamte Text sichtbar. Weitere abweichende Darstellungen oder Funktionsweisen können geplant werden, um spezielle Gerätefunktionalitäten ausnutzen zu können, sobald diese zur Verfügung stehen. Auf Smartphones kann es je nach Funktion einer

[15] *Vgl. bvh, … Mobiler Einkauf legt 2013 noch einmal deutlich zu … , 20.01.2014*

Webseite sinnvoll sein, den Bewegungssensor, die Ortsbestimmung oder die Kamera zu verwenden.

Während der Konzeption einer Webseite werden Skizzen angefertigt, die verdeutlichen, wie die Webseite aufgebaut und welche Elemente in welcher Form dargestellt werden sollen. Diese Skizzen werden Wireframes oder auch Mockups genannt und können handgemalt oder digital erstellt sein. Hierzu gibt es verschiedene Tools, wie zum Beispiel „Balsamiq Mockups“ (Balsamiq) oder „Axure RP“ (Axure). In beiden Tools ist es möglich, die Skizzen wie handgemalt aussehen zu lassen (Abb. 2-1).

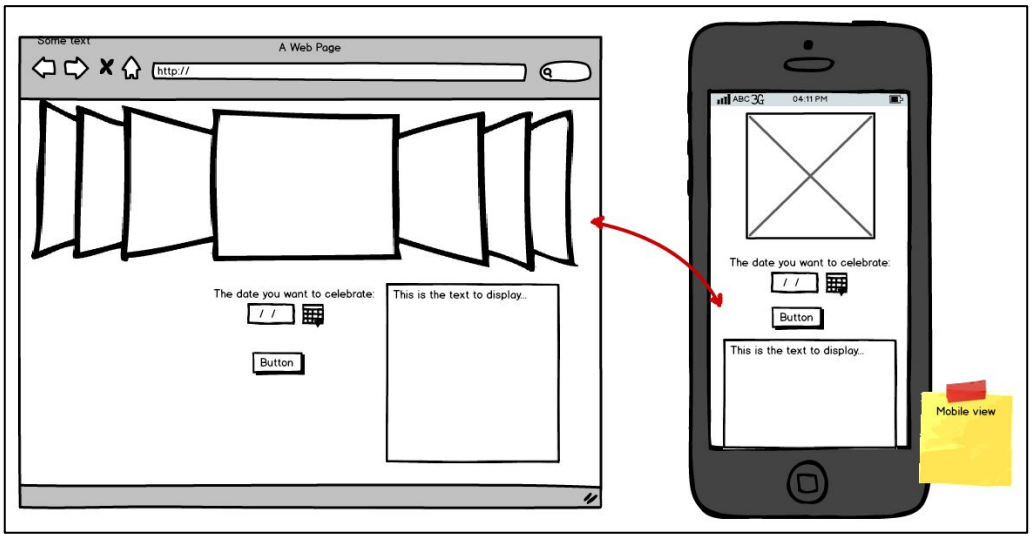

Abb. 2-1 Skizze erstellt mit Balsamiq Mockups

Diese Darstellungsweise hat den Vorteil, dass der Betrachter nicht dazu verleitet wird anzunehmen, es könne sich um das tatsächliche Aussehen beziehungsweise Design der späteren Webseite handeln. Die Wireframes sollen lediglich das Grundgerüst der Webseite und deren Bedienelemente verdeutlichen. Dies gilt meiner Meinung nach zumindest für einen sehr frühen Zeitpunkt im Projekt. Detailliertere Wireframes können sinnvoll sein, um im Rahmen des Designs die Platzverteilung der einzelnen Texte und Inhalte auf der Seite zu verdeutlichen. Dies wird nachfolgend erläutert.

Nach dem Prinzip „Mobile First“ startet man bei der Konzeption von reaktionsfähigen Webseiten und –applikationen bei einem Wireframe für ein kleines Display und entwickelt anschließend einen für eine Desktop-Ansicht. Natürlich können gegebenenfalls auch weitere Formate notwendig sein. Um die Planung effektiv zu gestalten, sollte auch die Erstellung der Wireframes in enger Zusammenarbeit der einzelnen umsetzenden Disziplinen und des Auftraggebers erfolgen.

2.1.3 Design in the Browser

Durch die Planung liegen Wireframes, die das grobe Layout der Seite auf verschieden großen Anzeigegeräten verdeutlichen, und die Inhalte in priorisierter Reihenfolge vor. Da bei der Erstellung des detaillierten Layouts weiterhin vorranging auf den Inhalt zu achten ist, bietet es sich an, diesen nun zu strukturieren. Hierzu kann der Text in ein Markdown-Dokument überführt werden, wie es zum Beispiel der Webentwickler Krasimir Tsonev bevorzugt.[16] Markdown ist eine Auszeichnungssprache, mit der man Texte strukturiert und formatiert. So können beispielsweise Überschriften verschiedener Ebenen, Aufzählungspunkte und Hyperlinks definiert werden. Auch können Textpassagen markiert werden, die fett, kursiv oder unterstrichen sein sollen. Für ein solches Markdown-Dokument ist es notwendig, dass die Inhalte feststehen und ein Verständnis für die Aussagen der zu entwickelnden Webseite vorhanden ist. Nur dann können zu den einzelnen Texten Titel, Untertitel und Abschnitte definiert werden. Ein so erstelltes Markdown-Dokument kann anschließend in HTML konvertiert und in einen ersten Prototyp eingefügt werden.

Andere Webdesigner bevorzugen es, den Inhalt der Seite sofort in HTML zu strukturieren. Bei der Auswahl des Vorgehens spielt es sicherlich eine Rolle, wie vertraut der Ersteller und auch der Auftraggeber mit den Techniken sind. Da beide Seiten eng zusammenarbeiten sollten, ist es sinnvoll, dass auch der Auftraggeber die Form der Strukturierung versteht.

Wendet man sich nun der Erstellung eines Designs zu, so stellt sich die Frage, wie hierzu bei responsivem Webdesign am sinnvollsten und effektivsten zu verfahren ist. Wie schon beschrieben, ist ein statisches, in einem Grafikprogramm erstelltes Bild nicht geeignet, der Reaktionsfähigkeit von Webseiten Rechnung zu tragen. Wie kann einem Auftraggeber ein Design für eine sich an die Systemumgebung anpassende Webseite vorgestellt werden? Am besten in einem Browser, dem Tool, in dem auch die spätere Seite ausgeführt werden wird.

---

[16] Vgl. Tsonev, How to Design Responsively, 20.01.2014

Über das Vorgehen, das Design direkt im Browser zu erstellen, hielt Andy Clarke im Jahr 2009 bei „An Event Apart“ einen Vortrag.[17] Seitdem gibt es viele begeisterte Anhänger des Vorgehens „Design in the Browser“, wie zum Beispiel die schon erwähnte Kerstin Hahlbohm[18] und Chris Thelwell[19], ein Webdesigner mit fünfzehnjähriger Erfahrung.

Vorteil der Erstellung eines Prototyps, der gleichzeitig das Layout darstellt und die in HTML vorbereiteten Inhalte enthält, ist nicht nur, dass die Reaktionsfähigkeit für den Auftraggeber sichtbar wird. Ein solcher Prototyp macht auch deutlich, dass die entstehende Webseite kein fixes, sondern ein variables Layout besitzen wird. Denn bei Anwendung des Prinzips „Progressive Enhancement“ unterscheidet sich die Webseite nicht nur bezüglich der unterschiedlichen Positionierung der Daten, sondern auch im Funktionsumfang. Auf ein und demselben Desktopsystem können sich die Seiten in verschiedenen Browsern aufgrund der jeweils darin implementierten Funktionalitäten unterscheiden.

Eine überflüssige Überarbeitung des Designs oder der Implementierung entfällt, wenn Auftraggeber und Umsetzer, wie es bei der Erstellung von responsiven Webseiten am effektivsten ist, eng zusammenarbeiten. Wird zuerst ein Design in einem Grafikprogramm entworfen und vom Kunden abgenommen, so besteht die Gefahr, dass aufgrund von Problemen bei der responsiven Umsetzung ein Design überarbeitet werden muss.

Ebenso kann es vorkommen, dass eine Implementierung eines Designs anders wirkt als ein grafisches Design-Dokument und dadurch eine Überarbeitung durch den Kunden gewünscht wird. Wird das Design direkt im Browser erstellt und kommunizieren Designer, Entwickler und der Auftraggeber iterativ, so werden zu keinem Zeitpunkt Erwartungen an die Webseite verfestigt, die nicht einzuhalten sind.

Bei dieser Art der Erstellung des Designs, nämlich in Form eines Prototyps, stellt sich noch die Frage, für welche Systemumgebung begonnen werden soll. Orientiert man sich wieder am kleinen Display eines Smartphones und

---

[17] *Vgl. Clarke, Walls Come Tumbling Down, 20.01.2014*

[18] *Vgl. Hahlbohm, Responsive Webdesign Workflow – Sechs Praxistips, 20.01.2014*

[19] *Vgl. Thelwell, Design-in-browser, are you ready?, 20.01.2014*

beachtet „Content First“, so ergibt sich hieraus ein übersichtlich designter Prototyp, in dem die Vermittlung des Inhalts an den Nutzer im Vordergrund steht. Eine Navigation sollte zum Beispiel keinen zu großen Raum im Verhältnis zum eigentlichen Inhalt einnehmen. Oft wird jedoch kritisiert, dass ein Design nach dem Prinzip „Mobile First“, bei dem zunächst für kleine Geräte entworfen und anschließend auf ein Design für die Anzeige auf größeren Monitoren erweitert wird, zu wenig gestalterische Mittel aufweist. Solche Designs wirken sehr schlicht und wecken keine Emotionen beim Betrachter. Deswegen ist bei diesem Vorgehen besonders darauf zu achten, dass für alle Formate einer Seite ausreichende Gestaltungselemente verwendet werden und die Seite ausdrucksstark gestaltet wird.

Entwirft man das Design nach dem Prinzip „Desktop First“, so hat man den Vorteil, dass mehrere zu verwendende Elemente gleichzeitig auf dem Bildschirm angezeigt werden. Hierdurch entsteht ein besserer Gesamteindruck der Seite. Dieser lässt mehr gestalterische Kreativität zu.

Oft wird eine maximale Breite von 960 Pixeln für Webseiten festgelegt, da es ansonsten schwierig wird, den über eine sehr große Breite angezeigten Text zu lesen. Ist eine Seite jedoch klar strukturiert und in Spalten aufgeteilt, so kommt es auch auf responsiven Webseiten nicht zu sehr breit dargestelltem Text. Für Seiten, die sehr viel Inhalt vermitteln sollen, kann somit eine reaktive Ausnutzung der kompletten Anzeigefläche den Vorteil einer besseren Übersicht über alle Inhalte bieten.

Ist es für eine Webseite essentiell, dass technische Möglichkeiten, wie sie nur Smartphones und Tablets bieten, genutzt werden, so sollte mit der Gestaltung per „Mobile First“ gearbeitet werden.[20] Solche gegebenenfalls notwendigen Funktionen können beispielsweise der Bewegungssensor, die Kamera, die Lokalisierung oder eine Touch-Steuerung sein. Bei der Planung ist dann eine Betrachtung, wie die Webseite auf Geräten ohne diese Funktionen reagieren soll, notwendig.

Nimmt man den in HTML strukturierten Inhalt als Grundlage und arbeitet nach dem Prinzip „Mobile First“, so erreicht man mit dem Design im Browser eine

[20] *Vgl. Zillgens, Responsive Webdesign, Seite 65*

fortgeführte Fokussierung auf die Inhalte und wird nicht dazu verleitet, das Design über den Inhalt zu stellen.

Um ein reaktionsfähiges Design zu erhalten, muss mit den vorangehend vorgestellten fließenden Rastern gearbeitet werden.

Technische Hilfsmittel zur Erstellung des Designs im Browser sind HTML, CSS und JavaScript, wodurch auch Funktionalität entworfen werden kann. Bei dieser Vorgehensweise ist es notwendig, dass der Designer mit diesen Techniken umgehen kann und er bereit ist, sich von dem bisher üblichen Grafikprogramm zur Erstellung eines Designs zu verabschieden.

Es gibt des Weiteren Tools, die bei der Erstellung im Browser die Arbeit erleichtern.

Das Tool „960 Gridder“ (http://peol.github.io/960gridder/, 20.01.2014) ermöglicht, im Browser ein Raster mit 960 Pixel Breite über eine Webseite zu legen. Die Anzahl der Spalten und deren Breite kann flexibel eingestellt werden, leider jedoch nicht die Gesamtbreite des Rasters. Dies ist, wie der Name schon andeutet, fest auf 960 Pixeln eingestellt. Allerdings darf man sich durch ein solches Raster nicht verleiten lassen, feste Pixelwerte zur Strukturierung der Webseite zu verwenden. Auf „Fluid Grids“ ist zu achten.

Ob es allerdings sinnvoll ist, nach der Aufstellung der Inhalte und einiger Wireframes, die das grobe Layout der zu erstellenden Webseite auf verschiedenen Geräten darstellen, direkt zum Design im Browser über zu gehen, wird unter Webdesignern differenziert gesehen. Manchen sind Wireframes, die lediglich das Grundgerüst der Webseite und deren Bedienelemente verdeutlichen, nicht ausreichend, um mit dem Design zu beginnen. Sie sind der Meinung, dass die Wireframes auf Basis der zuvor erarbeiteten Hierarchie der Inhalte, die Anordnung und den Platzbedarf der Texte und sonstiger Elemente verdeutlichen sollen. Solche Wireframes stellen eine wesentlich detailliertere Ausarbeitung des Layouts, als in dem vorangehend beschriebenen Verfahren, dar. So ist der Autor des Buches „Responsive Webdesign“, Christoph Zillgens, der Meinung, dass Wireframes, die die tatsächlichen Inhalte der Seite enthalten, einen guten Eindruck

vermitteln „*wie die Inhalte vom Umfang und von der Platzierung her auf den verschiedenen Geräten wirken*“[21].

Die Inhalte, die es in einem Projekt umzusetzen gilt, oder auch die gewünschte Intensität der Zusammenarbeit zwischen Auftraggeber und Webdesigner beeinflussen sicherlich die bevorzugte Vorgehensweise. Doch sollte man im Rahmen des responsiven Designs so lange wie möglich flexibel bleiben und nicht zu früh Details ausarbeiten, da ansonsten immer wieder Überarbeitungen notwendig werden. Dies spricht meiner Meinung nach für Wireframes, die das Layout grob andeuten, und ein anschließendes Design im Browser, in das die Inhalte eingefügt werden. Eine ständige Einbeziehung des Auftraggebers verhindert aufwändige Überarbeitungen. Durch ein solches Verfahren kann das reaktionsfähige Design von Anfang an von Designern, Entwicklern und dem Kunden im Browser, der realen Umgebung, betrachtet und getestet werden. Hierbei sollten immer wieder unterschiedliche Browser und Geräte verwendet werden, sodass spezifische Probleme frühestmöglich festgestellt werden.

2.1.4 Style Tiles

Parallel zum Layout im Browser sollten Auftraggeber und Designer ein gemeinsames Verständnis über das Design bezogen auf Farben, Texturen, Bilder und Schriftarten erarbeiten. Als geeignetes Mittel haben sich in letzter Zeit „Style Tiles“ hervorgetan.[22] Die Designerin Samantha Warren hat das Konzept der „Style Tiles“ entwickelt und 2012 bei „A List Apart“ vorgestellt.[23]

Auf den beiden folgenden „Style Tiles“ Abb. 2-2 (http://styletil.es/ - Examples, 20.01.2014) ist erkennbar, dass mit ihrer Hilfe ein detaillierter Stil einer Seite entworfen werden kann. Dabei wird nur das Layout unterschiedlicher, immer wieder zu verwendender Elemente festgelegt, wie zum Beispiel Schriftstil, -größe und -farbe, generell verwendete Farben, Hintergrundmuster und das Aussehen der Links und Schaltflächen (englisch: buttons).

---

[21] *Zillgens, Responsive Webdesign, Seite 86*

[22] *Vgl. Warren, Style Tiles, 20.01.2014*

[23] *Vgl. Warren, A List Apart – Style Tiles and How They Work, 20.01.2014*

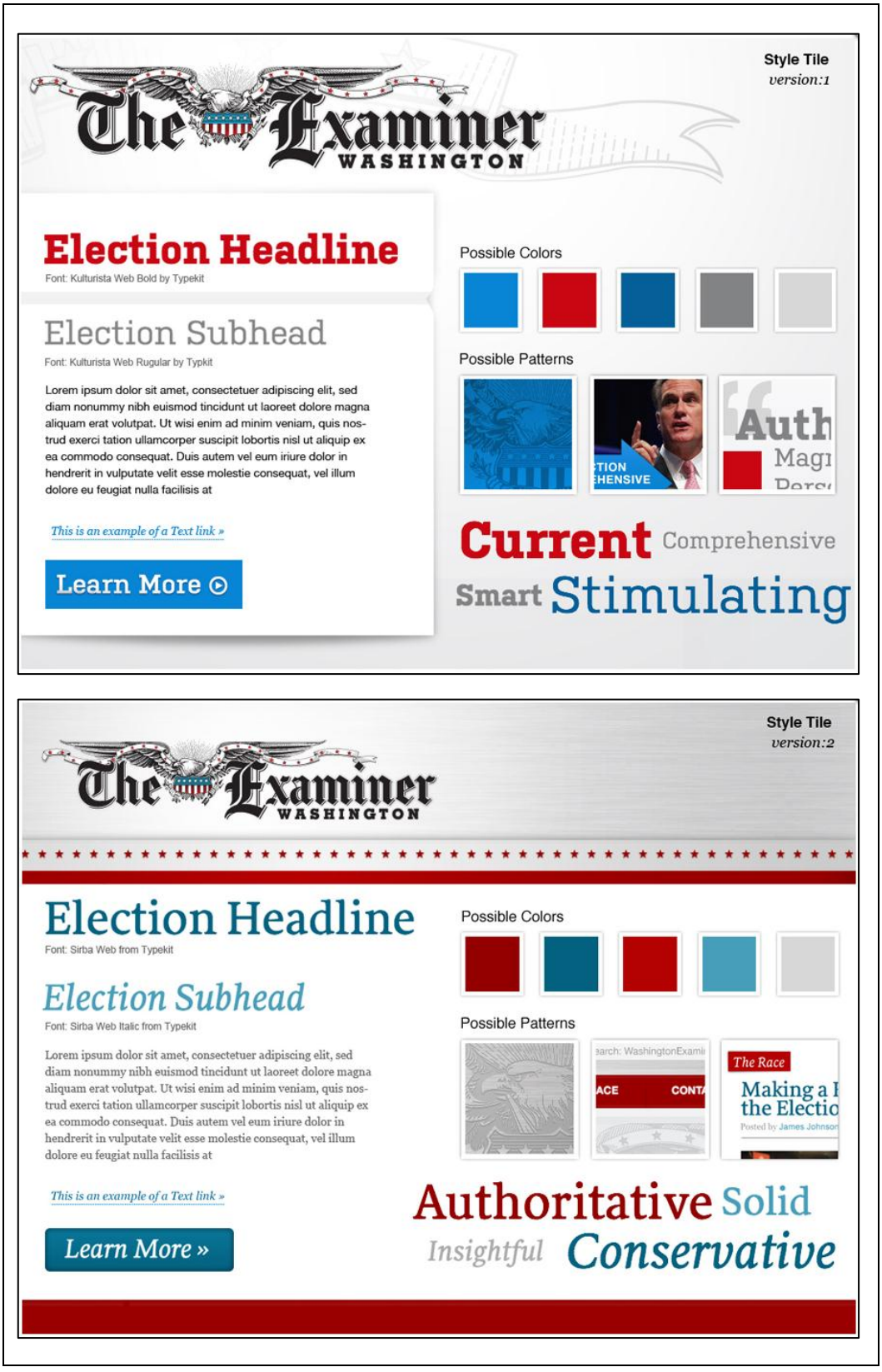

Abb. 2-2 Zwei „Style Tiles“ Beispiele einer Webseite

Meiner Meinung nach vermittelt eine solche Grafik sehr gut die Wirkung einer Webseite, die mit diesen Elementen gestaltet werden soll. Ebenso lassen sich für Firmen mit einem feststehenden Corporate Design mittels „Style Tiles“ die

Elemente herausarbeiten, die für eine Webseite oder –applikation verwendet werden sollen. Der Auftraggeber erhält ein oder mehrere visuelle Konzepte, anhand derer er sich für einen Stil der beauftragten Webseite entscheiden kann. Trotz des Detailgrades besteht bei „Style Tiles“ nicht die Gefahr, dass man aufgrund einer zu frühen Festlegung das Konzept überarbeiten muss. Die per Style Tiles beschriebenen Elemente haben zunächst keinen Einfluss auf das Gesamtdesign einer Webseite, welches zunächst nur die Anordnung der einzelnen Elemente für unterschiedliche Anzeigeformate festlegt.

Wurde ein Layout der Inhalte fertiggestellt, kann nun das in den „Style Tiles“ festgelegte Design der einzelnen Elemente in den Prototypen mittels HTML und CSS eingearbeitet werden.

2.1.5 Progressive Enhancement

Die vorangehend beschriebene Vorgehensweise zeigt, dass Design und Entwicklung keine voneinander zu trennenden Phasen bei der Entwicklung und Umsetzung von responsiven Designs sind. Für eine effiziente Durchführung eines Projektes ist es notwendig, dass Designer mit HTML, CSS und JavaScript umgehen können. Ebenso müssen Entwickler ein Verständnis für Design besitzen. Nur so kann in enger Zusammenarbeit der unterschiedlichen Disziplinen effizient gearbeitet werden.

Ist ein umzusetzendes Design einer Webseite erarbeitet worden, so können nun die Frontend-Entwickler mit der eigentlichen Umsetzung beginnen. Hierzu bildet der Design-Prototyp die Basis. Der darin enthaltene Code kann von den Entwicklern wiederverwendet werden.

Bei der Implementierung ist es wieder sinnvoll nach dem Prinzip „Mobile First“, ergänzt um „Progressive Enhancement“, zu arbeiten. Es wird mit einer einfachen Form der Webseite begonnen und zunächst lediglich HTML und CSS verwendet. Danach sollten in allen Browsern die grundlegenden Inhalte und Funktionen der Webseite verfügbar sein. Der Inhalt wird mit semantischen Elementen des HTML ausgezeichnet, sodass die eine Struktur der Inhalte im Code ersichtlich ist. Ob hierzu schon die neuen Elemente des HTML5 oder die älteren „div“-Elemente, versehen mit einer Klasse, verwendet werden sollen, wird im Verlauf der Arbeit betrachtet. Diese Auszeichnung hilft nicht nur dem Entwickler, sondern auch Suchmaschinen und Screenreadern, die Seite zu interpretieren. Das HTML wird möglichst frei von Formatierungen gehalten, sodass es den Inhalt der Seite widerspiegelt. Die Formatierungen werden in

externen CSS-Dateien platziert. Im Sinne von „Progressive Enhancement“ wird diese rudimentäre Seite nun schrittweise verbessert, indem ihr aktuellere Funktionen hinzugefügt werden. Da HTML und CSS fehlertolerant sind, ignorieren ältere Browser ihnen unbekannte Befehle und zeigen die Webseite, definiert durch die restlichen Befehle, korrekt an. Weitere Erweiterungen können mit Hilfe von spezifischerem CSS und JavaScript realisiert werden. JavaScript wird ebenfalls in separate Dateien ausgelagert.

2.2 Flexible Inhalte

Während der Planung einer responsiven Webseite sollte eine Liste der darzustellenden Inhalte, die nach der Wichtigkeit der Informationen und Elemente sortiert ist, erstellt werden. Eine Möglichkeit, die Inhalte einer Webseite responsiv zu gestalten, ist, die anzuzeigenden Daten flexibel auf das abrufende Gerät reagieren zu lassen. Die Liste dient hierzu als Grundlage.

2.2.1 Inhalte auslassen

Ein einfaches Verfahren wäre, auf kleinen Geräten wie Smartphones nur die wichtigsten Inhalte, die sich oben in der sortierten Liste wiederfinden, anzuzeigen. Auf Geräten mit größerer Anzeigefläche können zusätzliche Texte angezeigt werden, wenn der Platz auf dem Display es zulässt. Dieses Vorgehen behandelt die einzelnen Nutzer jedoch nicht gleich. Es werden den Smartphone-Nutzern Daten vorenthalten. Sind die Daten von Relevanz, so sollten sie auch den Anwendern mit kleinen Displays zugänglich sein. Besitzen die Daten keinerlei Bedeutung für die Erfüllung des Zwecks der Webseite, so muss man sich jedoch fragen, ob Nutzern mit großen Displays zugemutet werden soll, diese zusätzlichen Informationen angezeigt zu bekommen. Nach dem vorangehend beschriebenen Prinzip „Content First“ sollte sich darauf bezogen werden, dass der Inhalt das wichtigste Element einer Webseite ist. Überflüssiges sollte weggelassen werden - generell und nicht nur auf kleinen Geräten.

2.2.2 Inhalte ausblenden

Welche anderen Möglichkeiten gibt es, den Inhalt flexibel zu gestalten? Anstatt Text entfallen zu lassen, kann man die Überschrift und die ersten Sätze des Textes anzeigen und per Link auf eine separate Seite mit dem

gesamten Text verweisen. Oder man lässt den Text auf der aktuellen Seite mit Hilfe einer Schaltfläche komplett einblenden. Alternativ zur Anzeige des Beginns des Textes kann ein sogenannter Anreißertext (englisch: teaser), der einen Überblick über den Inhalt des Abschnitts vermitteln und zum Weiterlesen animieren soll, verwendet werden. Bei beiden Verfahren kann sich der Betrachter einen Überblick über die angebotenen Inhalte verschaffen und die für ihn relevanten auswählen.

Aufgrund der Menge der Texte nutzen typischerweise Zeitungen und Zeitschriften auf ihren Webseiten Anreißertexte, um die Leser zu einem Klick auf den Link, der zum gesamten Artikel führt, zu bewegen. Auf der Startseite der „Zeit“, woraus der Ausschnitt in Abb. 2-3 (www.zeit.de, 02.11.2013) entnommen ist, werden sowohl Überschriften mit Teaser-Texten als auch nur Überschriften verwendet. Die Überschrift ist jeweils der Link zum gesamten Artikel.

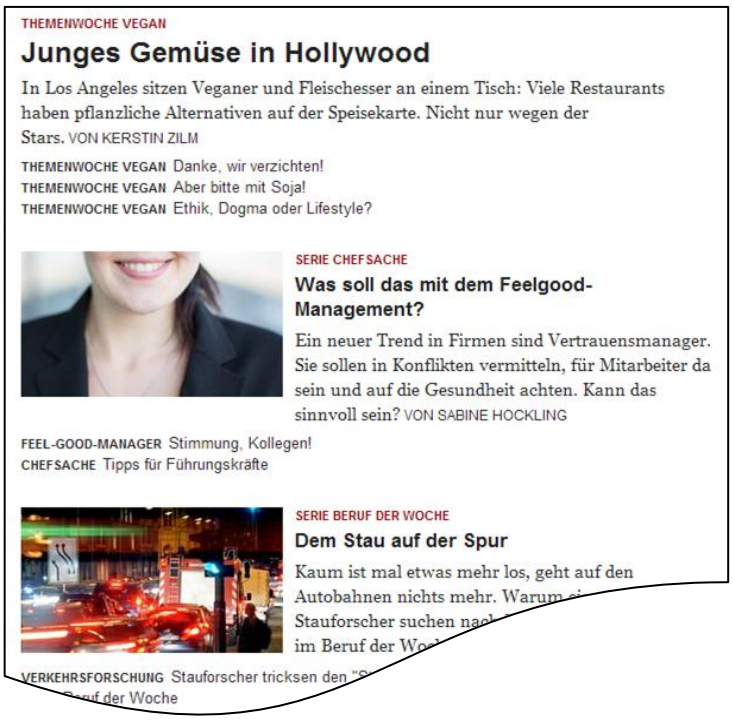

Abb. 2-3 Übersicht über die Artikel der Zeit

Akkordeon

Auf ähnliche Art und Weise verschaffen Akkordeons Übersichtlichkeit, wenn viel Text oder sonstiger Inhalt angezeigt werden soll. Bei Akkordeons ist ohne

Benutzeraktion nur eine Überschrift sichtbar. Durch Anklicken der Überschrift oder eines Symbols davor, meist ein Pfeil, öffnet sich der darunterliegende Bereich und der gesamte Inhalt wird sichtbar.[24] Realisiert werden Akkordeons in HTML und JavaScript. Um sie ansehnlich zu gestalten, wird CSS benötigt. Abb. 2-4 (www.wi.hs-wismar.de, 20.01.2014) und Abb. 2-5 (www.polargold.de, 20.01.2014) zeigen Beispiele für Akkordeons.

Abb. 2-4 Akkordeon mit Pfeil-Bedienelementen

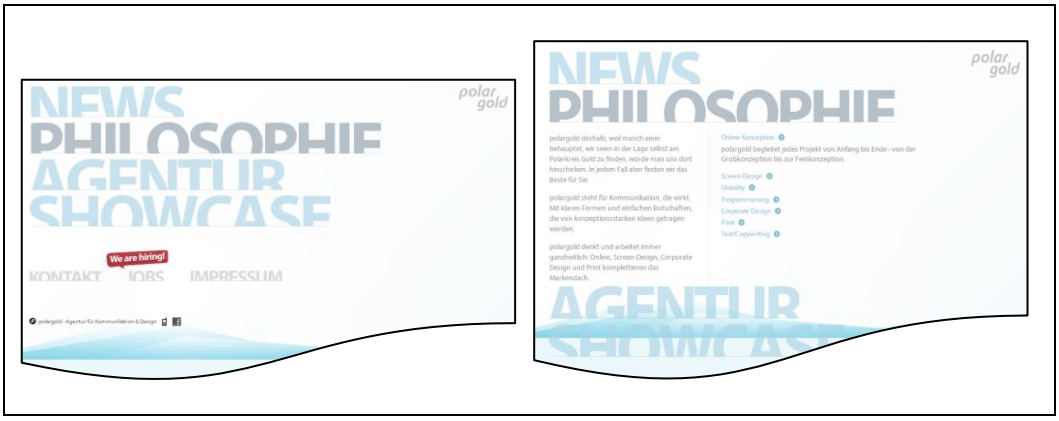

Abb. 2-5 Akkordeon ohne sichtbare Bedienelemente

Karussell

Um Inhalte, die zunächst auf einer Webseite nicht sichtbar sind, einzublenden, können zudem Karussells (englisch: carousel) verwendet werden. Sie werden auch Slider genannt. Karussells sind Bereiche einer Webseite, deren Inhalt sich in festgelegten Zeitintervallen oder durch Benutzerinteraktion ändert. Typischerweise ist für jeden iterierenden Inhalt ein Punkt unterhalb des

[24] *Vgl. Müller, Flexible Boxes, Seite 384*

Bereichs angebracht, über den zu der entsprechenden Seite navigiert werden kann. Auf der Seite des „Bundesministerium der Finanzen“ ist in dem verwendeten Karussellbereich ein Bild und ein anschließender Text enthalten, wie man in Abb. 2-6 erkennen kann (www.bundesfinanzministerium.de, 20.01.2014).

Abb. 2-6 Karussell mit Punkt für jede Inhaltsseite

Anhand der Punkte kann jedoch nicht erkannt werden, welcher Inhalt sich dahinter verbirgt. Deswegen nutzt der „General-Anzeiger“ Reiter-Elemente, in denen ein Schlagwort zu den einzelnen Seiten des Karussells angezeigt wird. Über diese Begriffe, die Abb. 2-7 zeigt, kann gezielt der Inhalt angesteuert werden (www.ga-bonn.de, 01.11.2013).

Abb. 2-7 Karussell mit Reitern

Weit verbreitet ist das Karussell „FlexSlider 2“, ein jQuery-Plugin.[25] Sowohl Christoph Zillgens („Responsive Webdesign“)[26], als auch Peter Müller („Flexible Boxes“)[27] beschreiben es in den genannten Büchern. Beide verwenden „FlexSlider 2“ in der zum Buch erstellten Beispielseite. Zillgens Seite steht online unter http://responsive-webdesign-buch.de/ und Müllers Beispiel kann über den Link http://pmueller.de/flexibleboxes.html unter „Beispieldateien zu Flexible Boxes“ heruntergeladen werden. Die unterschiedlichen zur Verfügung stehenden „FlexSlider 2“-Varianten sind responsiv gestaltet. Das heißt, das Karussell passt sich der Bildschirmgröße an und unterschiedliche Bedienmechanismen, wie Wischen auf dem Smartphone und Mausklicks an einem Desktop, werden unterstützt.

Lightbox

Lightboxes, die Inhalte in einem modalen Dialog über der eigentlichen Webseite anzeigen, werden meist zur Anzeige von Bildern genutzt. In ihnen können technisch gesehen auch andere HTML-Elemente und somit Inhalte oder Formulare angezeigt werden.

Auf der Webseite der JavaScript Library „Lightbox2“ werden Beispiele gezeigt. Ein Beispiel ist in Abb. 2-8 dargestellt (lokeshdhakar.com/projects/lightbox2, 20.01.2014).

Das Beispiel in Abb. 2-9, das von der Seite FancyBox stammt, zeigt einen Text, der in einem modalen Dialog angezeigt wird (fancybox.net, 20.01.2014).

[25] Vgl. WooThemes, FlexSlider 2, 20.01.2014

[26] Vgl. Zillgens, Responsive Webdesign, Seite 214

[27] Vgl. Müller, Flexible Boxes, Seite 396

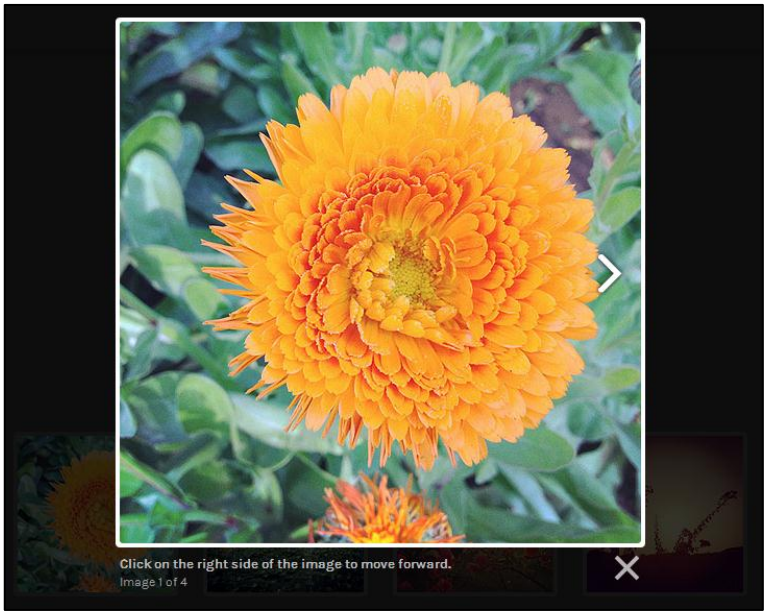

Abb. 2-8 Lightbox mit Bild

Abb. 2-9 Lightbox mit textuellem Inhalt

„Off Canvas“

„Off Canvas“ bedeutet übersetzt „neben der Leinwand“ und im Kontext responsiver Webseiten „außerhalb der Darstellungsfläche“. Nachrangige Elemente einer Seite oder die Navigation können, wie Christoph Zillgens es beschreibt, für kleinere Bildschirme außerhalb der Darstellungsfläche geparkt und per Knopfdruck in den sichtbaren Bereich gerückt werden. Die Layoutfläche ist hierbei größer als das Display. Um bestimmte Bereiche sichtbar zu machen, wird das Layout unterhalb des Bildschirms verschoben.

Die von ihm verwendete Grafik macht deutlich, wie dies zu verstehen ist (Abb. 2-10). [28]

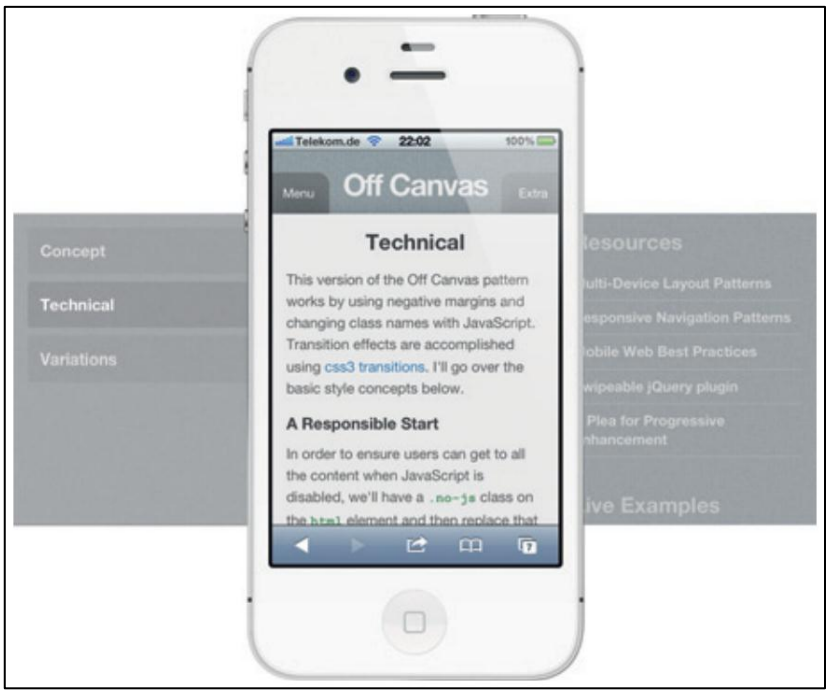

Abb. 2-10 Angedeutete „Off Canvas“ Spalten

Luke Wroblewski beschreibt auf seiner Webseite verschiedenste Möglichkeiten „Off Canvas“ anzuwenden.[29] Bei einem Beispiel wird auf kleinen Geräten die Navigation von oben in den sichtbaren Bereich hineingeschoben und Extras fahren vom linken Rand her in den sichtbaren Bereich (http://jasonweaver.name/lab/lw-slidetopnav/, 20.01.2014). Ein anderes Beispiel zeigt, wie die Navigation und zusätzlicher Inhalt „Off Canvas“ platziert werden kann. Abb. 2-11 stellt hierzu die Desktop-Ansicht dar, bei der die Navigation und die Extras immer sichtbar sind und in Abb. 2-12, der Smartphone-Ansicht, wird deutlich, dass nur der Hauptinhalt ständig sichtbar ist und die Extras samt Navigation in den sichtbaren Bereich geschoben werden können (http://jasonweaver.name/lab/lw-sidedrawer, 20.01.2014).

[28] *Zillgens, Responsive Webdesign, Seite 150*

[29] *Vgl. Wroblewski, Off Canvas Multi-Device Layouts, 20.01.2014*

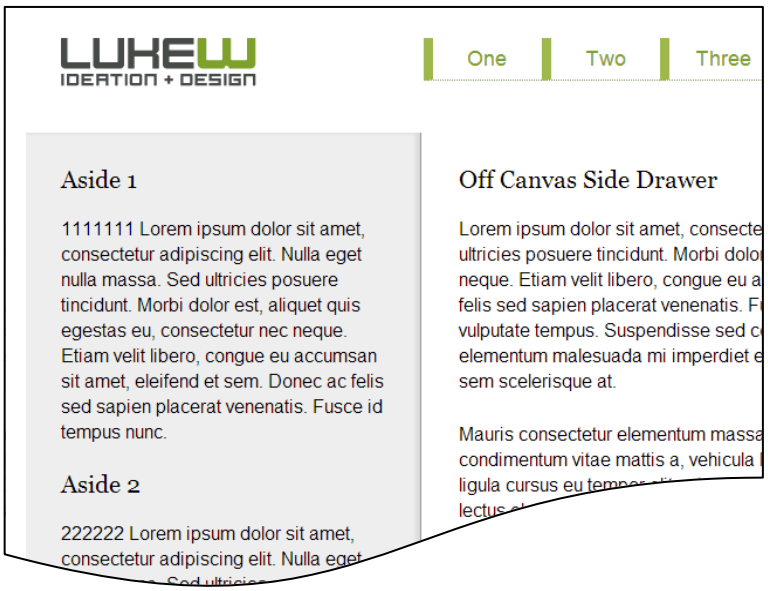

Abb. 2-11 Desktop-Ansicht „Off Canvas" Beispiel

Abb. 2-12 Smartphone-Ansicht mit und ohne „Off Canvas" Bereich

Nachteile

Die oben beschriebenen Verfahren, Inhalte zunächst auszublenden, besitzen jedoch auch Nachteile. Sebastian Preuss beschreibt die Probleme in einem Artikel zu Akkordeons, wie im Folgenden beschrieben.[30] Diese Probleme

[30] *Vgl. Preuss, //SEIBERT/MEDIA-Weblog - Wann sind Akkordeon-Effekte auf Websites sinnvoll?, 20.01.2014*

lassen sich auch auf die anderen zuvor beschriebenen Techniken übertragen. Öffnet der Nutzer den Bereich mit dem gesamten Inhalt nicht, weil er den Mechanismus nicht erkennt oder zu bequem ist, so bleibt der Inhalt unbeachtet. Ein weiteres Problem entsteht, wenn der Nutzer über eine Suche in einer Suchmaschine auf die Seite gelangt ist. Wurde der Begriff durch die Suchmaschine in einem Textbereich, der mit JavaScript ausgeblendet wurde, gefunden, so ist der Begriff zunächst nicht sichtbar. Der Anwender kann das gesuchte Schlagwort mit bloßem Auge und mit der Suchfunktion des Browsers nicht finden. Der Benutzer glaubt, auf eine nicht zu seinem Suchbegriff passende Seite gelangt zu sein. Bei der Optimierung für Suchmaschinen (englisch: Search Engine Optimization (SEO)) führen Seiten, auf denen sehr viel Text zu unterschiedlichsten Themen angeboten wird, zu Nachteilen beim Ranking innerhalb der Treffer. Tritt ein Suchbegriff im Verhältnis zum gesamten Umfang der Wörter selten auf, so wird die Seite nachrangig bewertet. Ein weiterer Nachteil bei Akkordeons ergibt sich für Nutzer mit deaktiviertem JavaScript. Akkordeons werden nicht zusammengeklappt und der gesamte Text ist sichtbar, was die Seite unübersichtlich macht. Laut Sebastian Preuss haben seiner Erfahrung nach fast 10 Prozent der Nutzer einer Seite JavaScript deaktiviert, sodass immerhin jedem zehnten Nutzer die Seite unstrukturiert angezeigt wird.

2.3 Responsive Navigation

Den Nutzer einer Webseite interessieren die Inhalte oder die Funktionen der Seite. Meist erhält man erste wichtige Informationen auf der Startseite. Erst nach der Betrachtung beziehungsweise Nutzung der ersten Seite entsteht bei der aufrufenden Person das Bedürfnis, auf andere Seiten der Webseite zu navigieren. Aus diesem Grund sollte das Layout der Navigation dem eigentlichen Inhalt der Seite nachrangig gestaltet sein. Es wird wieder das Prinzip „Content First“ angewandt. Um der Navigation optisch nicht zu viel Gewicht zu verleihen, sind kurze, aussagekräftige Begriffe für die Menüpunkte sinnvoll. Auch sollte darauf geachtet werden, dass in der Hauptnavigation nur die wichtigsten Seiten verlinkt sind, sodass das Navigationsmenü übersichtlich und für den Anwender schnell erfassbar ist. Luke Wroblewski sagt, dass es wichtig ist, darüber nachzudenken, was der Nutzer standardmäßig auf der Seite macht. Danach sollte die Struktur der Webseite ausgerichtet werden. „Content First“ und eine möglichst geringe Anzahl an Navigationsmöglichkeiten, über die die Hauptinhalte oder Funktionen erreicht

werden können, helfen dem Anwender schnell, die gewünschten Inhalte zu erreichen, auch wenn sie der Seite nicht die volle Aufmerksamkeit schenken, wie das bei Smartphone-Nutzern häufig der Fall ist.[31]

Für ein Layout, das für eine Desktopanzeige konzipiert wird, ist das Anordnen der Navigation nicht besonders herausfordernd. Die einzelnen Menüpunkte können horizontal im Kopfbereich (englisch: header) der Webseite oder vertikal in einer Spalte am linken Rand untergebracht werden. Für kleine Anzeigegeräte gestaltet sich die Planung der Navigation jedoch schwieriger. Ständig eingeblendete Navigationsmenüs lassen auf Smartphones kaum Platz für den eigentlichen Inhalt. Denn da diese Geräte meist per Touch-Oberfläche bedient werden, müssen die einzelnen Menüpunkte eine gewisse Mindestgröße einhalten, damit der Nutzer sie auch ohne Komplikationen auswählen kann.

Möchte man Smartphone-Nutzern eine gute Nutzbarkeit (englisch: usability) bieten, so nimmt eine Navigation sehr viel Platz ein. Folgende Möglichkeiten eine Navigation zu gestalten, haben sich entwickelt und teilweise bewährt.

Navigation auch auf kleinen Displays horizontal im Kopf

Eine leicht und schnell umzusetzende Navigation kann mit horizontal angeordneten Links oder Schaltflächen (englisch: buttons) realisiert werden. Es ist darauf zu achten, dass die Buttons ausreichend groß gestaltet werden, damit sie per Touch-Bedienung gezielt auswählbar sind. Sinnvoll ist diese Möglichkeit jedoch nur, wenn sehr wenige, kurze Navigationspunkte aufgelistet werden sollen, da ansonsten die Navigation zu viel Platz im Kopfbereich einnimmt. Vorteil dieses Verfahrens ist, dass JavaScript nicht benötigt wird.[32] Ein stimmiges Beispiel für eine übersichtliche Navigation im Kopfbereich bietet die Webseite von Ethan Marcotte. In Abb. 2-13 wird links die Smartphone- und rechts die Desktopanzeige dargestellt (ethanmarcotte.com, 20.01.2014).

[31] *Vgl. Wroblewski, Mobile First, Seite 65*

[32] *Vgl. Zillgens, Responsive Webdesign, Seite 236*

Abb. 2-13 Horizontal angeordnete Navigation im Header

Navigation im Fußbereich mit Navigationsbutton im Kopfbereich

Ebenfalls recht effizient und ohne JavaScript zu realisieren ist die Möglichkeit, die Navigation ans Ende der Webseite zu stellen. Da im Kopfbereich lediglich ein Navigationsbutton in Form eines Anker-Links (englisch: anchor link), der auf die Navigation referenziert, platziert wird, bleibt auf kleinen Anzeigegeräten viel Platz für den eigentlichen Inhalt. Dieses Vorgehen bietet sich an, wenn Menüs mit vielen Einträgen oder mit Untermenüs für eine effiziente Navigation notwendig sind. Im Fußbereich (englisch: footer) ist viel Platz vorhanden und es muss kein Inhalt zusätzlich angezeigt werden.[33] Brad Frost erwähnt in einem Artikel, in dem er Vor- und Nachteile unterschiedlicher Navigationsmenüs beschreibt, dass man durch einen schnellen Sprung ans Ende einer Seite leicht die Orientierung verlieren kann. Seiner Meinung nach gibt es elegantere Methoden eine Navigation zu gestalten.[34] Die Webseite „Webdesigntuts+“ bietet eine Anleitung zur Erstellung einer Navigation im Fußbereich (http://webdesign.tutsplus.com/tutorials/htmlcss-tutorials/a-simple-responsive-mobile-first-navigation, 20.01.2014). Umgesetzt wurde eine solche Navigation auf der Seite „Grey Goose“ (http://greygoose.com, 18.11.2013). Im oberen Teil der Abbildung Abb. 2-14 sieht man die Navigation im Fußbereich einer Desktopanzeige. Die unteren Bilder zeigen eine Smartphonedarstellung,

[33] *Vgl. Müller, Flexible Boxes, Seite 271*

[34] *Vgl. Frost, brad frost web – Responsive Navigation Patterns, 20.01.2014*

in der per „MENU“-Button zur Navigation am Ende der Seite gesprungen werden kann.

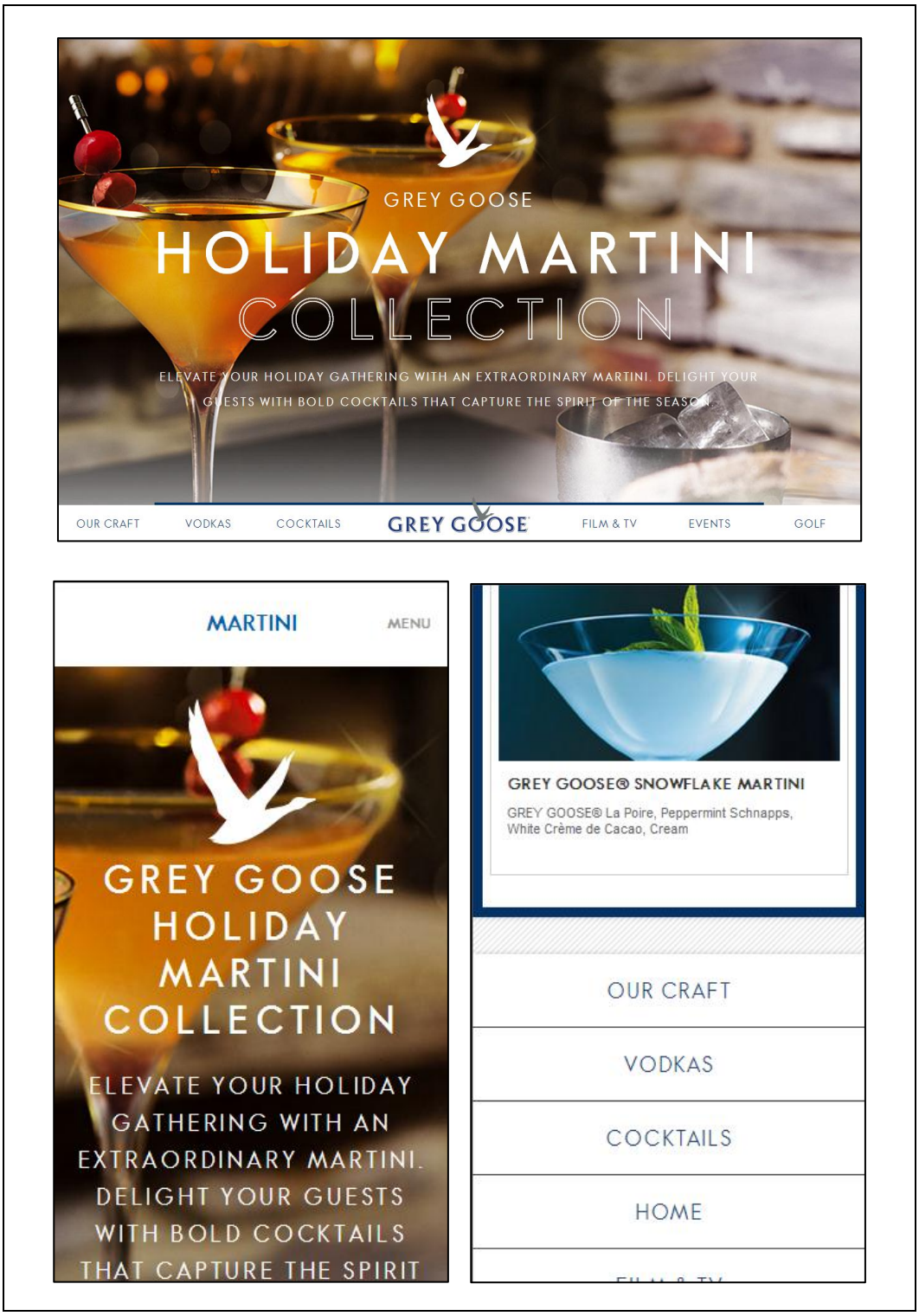

Abb. 2-14 Navigation im Fußbereich

Navigation per Dropdown-Liste

Auch mittels einer Dropdown-Liste, auch Select-Box genannt, kann ein Navigationsmenü gestaltet werden. Jeder in der aufklappbaren Liste existierende Eintrag stellt einen Navigationsmenüpunkt dar. Abb. 2-15 zeigt links das Menü als Dropdown-Liste auf einem Smartphone. Das rechte Bild zeigt eine Desktop-Ansicht mit ständig sichtbaren Menüpunkten. (http://www.fivesimplesteps.com/, 10.11.2013)

Abb. 2-15 Dropdown-Liste als Navigationsmenü

Da eine solch platzsparende Anzeige eines Menüs sicherlich nur auf kleinen Geräten verwendet werden wird, muss die raumgreifendere Navigation der Desktop-Anzeige, meist eine ungeordnete Liste, per „<ul>“ ausgezeichnet, für kleine Geräte aufwändig in das Formular-Element „<select>“ umgewandelt werden.[35] Da dieses Vorgehen zumindest in Anfangszeiten von responsivem Webdesign gebräuchlich war, existiert hierzu ein jQuery-Plugin auf der Seite „tinynav.viljamis.com“ (23.01.2014). Nachteil dieser Art des Menüs ist, dass abhängig vom Betriebssystem und dem Browser die Darstellung des Formularelements sehr unterschiedlich ist. Abb. 2-16 zeigt das unterschiedliche Aussehen im Safari auf Apples iOS und im Dolphine und Chrome auf Googles Android System. Es besteht seitens des Entwicklers keine Möglichkeit, das Aussehen zu beeinflussen.

[35] *Vgl. Müller, Flexible Boxes, Seite 272*

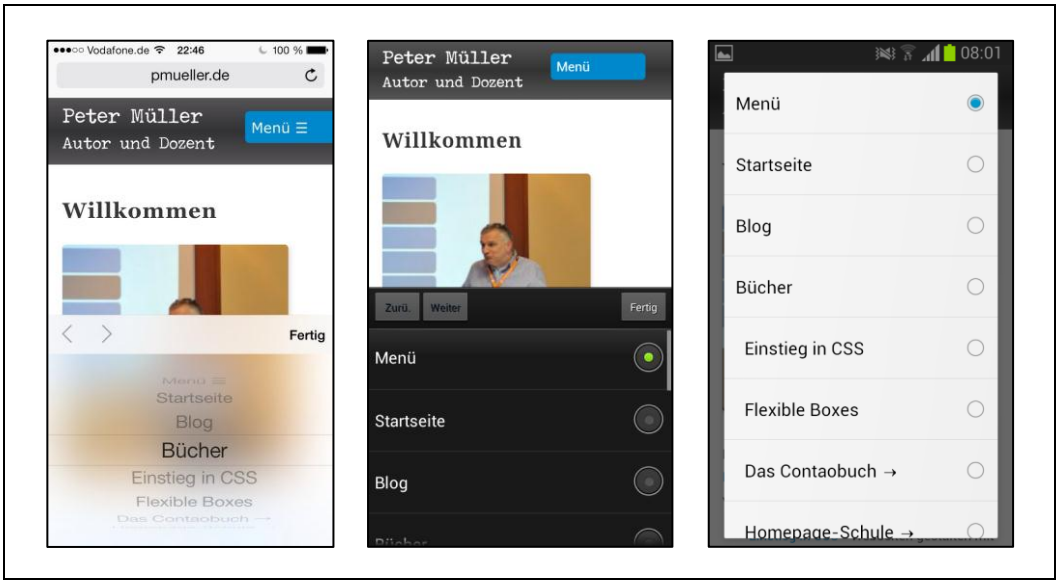

Abb. 2-16 Unterschiedliche Optik des Select-Menüs

Des Weiteren könnten Anwender, die das Dropdown-Menü meist aus Formularen zur Datenerfassung kennen, irritiert sein und kein Navigationsmenü hinter dem Element vermuten. Zusätzlich ist zur Realisierung JavaScript notwendig, das gegebenenfalls nicht auf allen aufrufenden Geräten zur Verfügung steht. Beachtet man das Prinzip „Progressive Enhancement“, so ist ein zusätzliches Menü für Geräte ohne JavaScript zu implementieren, da ansonsten ein Navigieren auf der Seite nicht möglich ist.

Navigation per Toggle-Menü

Das Toggle-Menü ist ebenfalls eine Art Dropdown-Menü. Im Kopf der Webseite gibt es einen Menübutton, der als Umschalter (englisch: toggle) dient und die Navigation bei Betätigung ein- beziehungsweise ausblendet. Der Button ist entweder beschriftet oder mit einem Symbol versehen. Ein eindeutiges Symbol für ein Menü hat sich noch nicht etabliert, sodass dem Anwender gegebenenfalls bei Verwendung eines Symbols nicht klar ist, was sich hinter dem Button verbirgt. Zurzeit wird teilweise ein auf der Spitze stehendes Dreieck, analog dem Symbol der Bedienfläche einer Select-Box, zum Aufklappen des Menüs verwendet. Häufiger sieht man mittlerweile jedoch Schaltflächen mit drei waagerechten Linien, die sich laut Andy Clarke zu einem Favoriten beim responsiven Webdesign herauskristallisieren.[36] Ein

[36] *Vgl. Clarke, We need a standard show navigation icon…, 23.01.2014*

Artikel auf der Webseite „GetElastic“ weist darauf hin, dass unbeschriftete Buttons eindeutig als Schaltfläche erkennbar sein sollen. Erreicht werden kann dies durch einen Rahmen, eine gleiche Gestaltung wie andere Buttons, die Anordnung neben anderen Bedienelementen und durch Abhebung vom Logo der Webseite.[37]

Ein Toggle-Menü erscheint meist als ein den Inhalt überlagerndes Menü unterhalb der Schaltfläche und haftet an dieser an. Das optische Aufklappen kann verlangsamt dargestellt werden, sodass für den Benutzer eindeutig zu erkennen ist, was passiert. Er wird nicht durch schnelle Sprünge an andere Stellen, wie den Fuß der Webseite, verwirrt. Ein solches Menü lässt sich frei gestalten, wodurch es dem Layout der Seite angepasst werden kann, wie in Abb. 2-17 erkennbar ist (http://fringewebdevelopment.com, 23.01.2014).

Abb. 2-17 Inhalt überlagerndes Toggle-Menü

Auch kann ein überlagerndes Menü von der Seite her über den eigentlichen Inhalt geschoben werden. Dies ist bei einer komplexen Navigationsstruktur, zum Beispiel mit Untermenüpunkten, sinnvoll, da die gesamte Höhe der Anzeigefläche genutzt werden kann. Auf der Webseite des „Content Management Systems“ (CMS) „Contao“ wird das für Desktop-Monitore horizontale Menü mit Unterpunkten in aufklappbaren Bereichen für kleine Anzeigegeräte in ein untergliedertes Toggle-Menü umgewandelt. Zusätzlich wird noch eine Scrollleiste notwendig, da die Höhe des Displays nicht ausreicht, alle Menüpunkte in einer per Touch-Bedienung auswählbaren

[37] *Vgl. Bustos, Don't Make These Mobile Menu Mistakes, 23.01.2014*

Größe aufzulisten. Abb. 2-18 zeigt die Darstellung des Menüs auf Smartphpones und Desktops (contao.org, 23.01.2014).

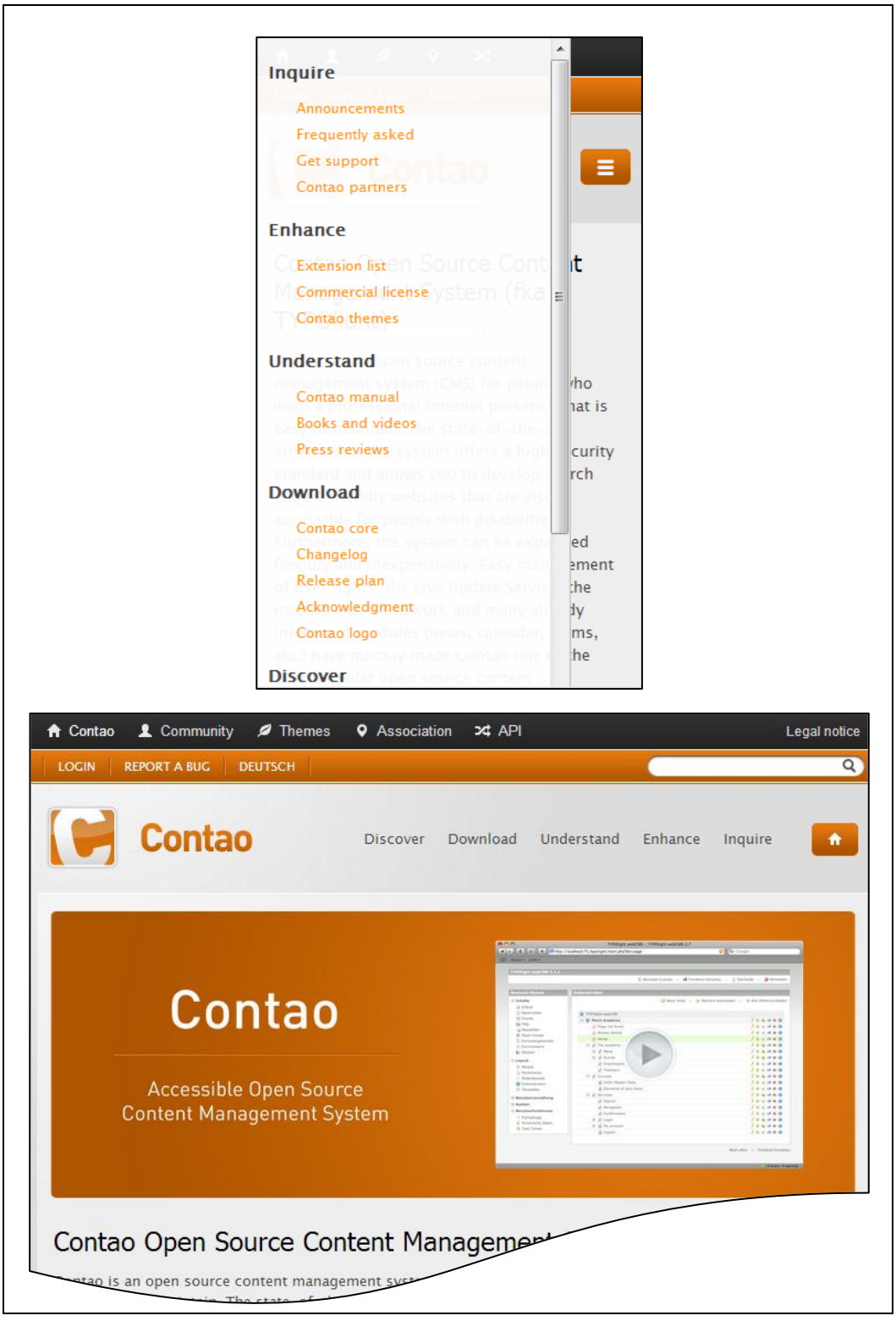

Abb. 2-18 Toggle-Menüs in der Mobil- und Desktop-Ansicht

Wie an der Homepage von „Contao“ erkennbar ist, sind der Gestaltung der Toggle-Menüs kaum Grenzen gesetzt. Durch diese Möglichkeit wird ein

solches Menü hinsichtlich der Entwicklung wesentlich aufwändiger. Ein weiterer Nachteil bei überlagernden Menüs ist, dass JavaScript notwendig ist. Dadurch muss zusätzlich ein Menü auch für nicht JavaScript-fähige Browser beziehungsweise Geräte angeboten werden. Auf Geräten mit geringer Prozessorleistung kann das Öffnen des Menüs ruckeln, da es rechenintensiv ist.[38]

Ohne die Verwendung von JavaScript lässt sich ein Toggle-Menü gestalten, das per CSS ein- und ausgeblendet wird. Eine per CSS gestaltete Animation läuft flüssiger als eine per JavaScript erstellte.[39] Meist wird ein solches Menü unterhalb des Kopfbereiches per CSS sichtbar geschaltet. Der nachfolgende Inhalt verschiebt sich bei Betätigen des Navigationsbuttons nach unten. Ein solches Menü sollte sich mit demselben Schalter auch wieder ausblenden lassen, damit ein Benutzer, der das Menü versehentlich geöffnet hat, wieder den eigentlichen Inhalt der Seite sieht, ohne scrollen zu müssen. Die Kaffeekette „Starbucks“ verwendet die zuvor erwähnten drei horizontalen Linien als Symbol auf dem Navigationsmenübutton. Nach dem Öffnen des Menüs verwandelt sich das Symbol in das zum Schließen von Anwendungen bekannte Kreuzchen („x“), wie Abb. 2-19 zeigt. Damit lässt sich die Navigation wieder ausblenden. (http://www.starbucks.de/, 23.01.2014).

[38] Vgl. Zillgens, Responsive Webdesign, Seite 243

[39] Vgl. Zillgens, Responsive Webdesign, Seite 252

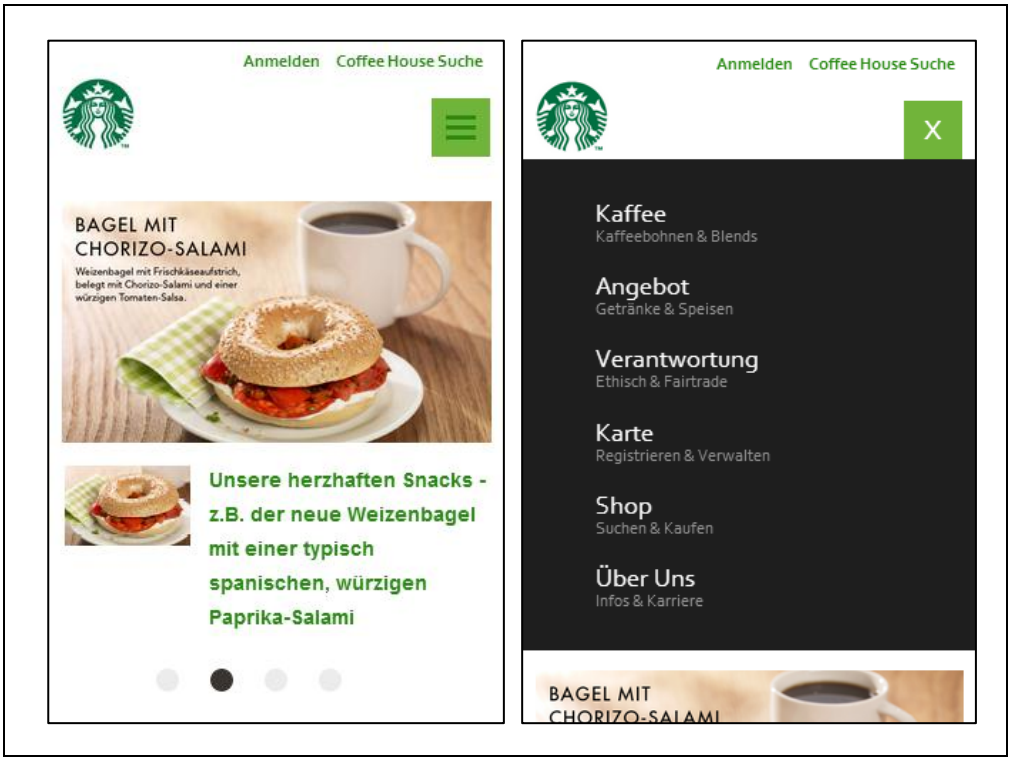

Abb. 2-19 Hauptinhalt verschiebende Toggle-Navigation

Navigation „Off Canvas"

Das vorangehend im Abschnitt „Flexible Inhalte" erwähnte Verfahren „Off Canvas" eignet sich ebenfalls für die Gestaltung von Navigationsmenüs. Meist links vom standardmäßig im Sichtbereich befindlichen Inhalt der Seite ist ein Bereich mit der Navigation vorhanden. Dieser Bereich wird bei Betätigen eines Buttons in den sichtbaren Bereich verschoben. Damit der Benutzer versteht, wo er sich gerade befindet, wird oft ein schmaler Streifen des Inhaltes bei sichtbarem Menü beibehalten. Das Beispiel von Christoph Zillgens verdeutlicht dieses Verfahren anschaulich (Abb. 2-20).[40]

[40] *Vgl. Zillgens, Responsive Webdesign, Seite 246*

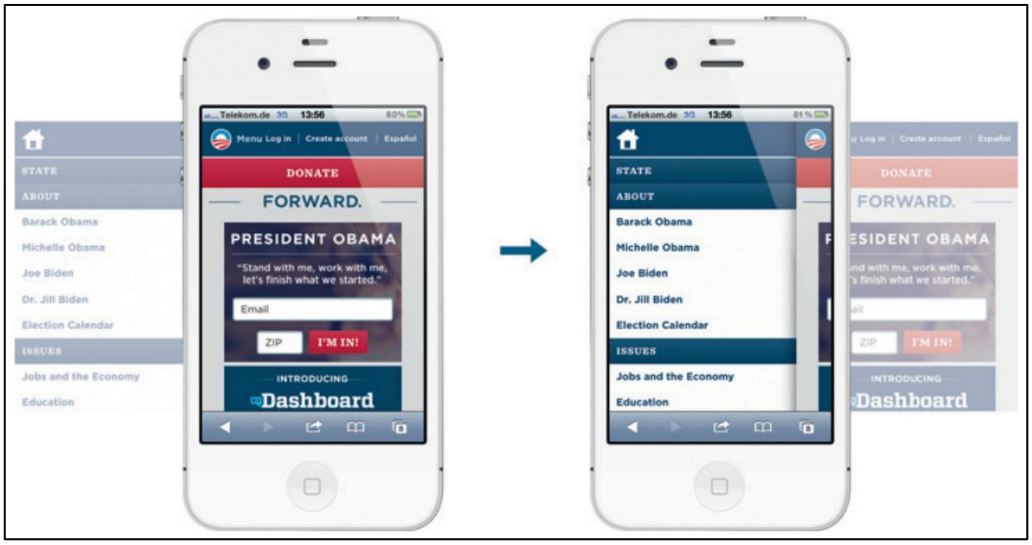

Abb. 2-20 „Off Canvas“ Navigation

Unzählige weitere Möglichkeiten

Die zuvor vorgestellten Verfahren, inklusive denen zum Ein- und Ausblenden von Inhalten, lassen sich natürlich kombinieren. Dadurch ergibt sich eine unbegrenzte Anzahl an Möglichkeiten zur Navigationsgestaltung. So kann man zum Beispiel in Toggle-Menüs Akkordeons mit Untermenüpunkten einfügen. Aus einem Karussell-Bereich, der zur Navigation dient, können Toggle-Untermenüs ausgeklappt werden. Aus einem Toggle-Menü, das den eigentlichen Inhalt nach unten verschiebt, lassen sich „Off Canvas“-Untermenüs in den sichtbaren Bereich schieben, die zunächst nicht sichtbar waren. Diese und andere Möglichkeiten beschreibt Brad Frost in seinen Artikeln „Responsive Navigation Patterns“ und „Complex Navigation Patterns for Responsive Design“.[41]

2.4 Semantisches HTML

HTML5 bietet die Möglichkeit, den Inhalt der Webseite semantisch auszuzeichnen. Betrachtet eine Person eine fertige Webseite, so geht für diese aus dem Layout hervor, welche Bedeutung welcher Text besitzt. Zum Beispiel geht der Name der Webseite beziehungsweise ein Firmenname aus dem gestalteten Kopf der Seite hervor, Überschriften sind größer dargestellt und an unterschiedlichen Überschriftgrößen lässt sich eine Unterstruktur

[41] *Vgl. Frost, brad frost web – Responsive Navigation Patterns… , 20.01.2014*

ausmachen. Für Geräte oder Anwendungen wie Screenreader, Suchmaschinen und Browser ist die Struktur der Seite ohne weitere technische Markierungen nicht zu erkennen. Ebenso fällt es einem Entwickler, der nur den Quellcode sieht, wesentlich schwerer, eine Struktur des Inhaltes auszumachen. In den HTML Versionen kleiner fünf war es üblich, eine Webseite in unterschiedliche „div“-Elemente zu strukturieren. Diesen Elementen wurden Klassen (englisch: class) oder identifizierende Attribute (englisch: Identifier (ID)) zugewiesen, die den Inhalt der Elemente beschrieben. So waren die IDs „header“ und „footer“ sehr gebräuchlich. Man kann sich jedoch vorstellen, dass deutsche Entwickler auch „kopf“ und „fuss“ als ID verwendeten. Solche nicht klar definierten Kennzeichnungen machen eine automatische Auswertung der Seite für Suchmaschinen, Browser und Screenreader natürlich unmöglich.

Mit HTML5 wurden Elemente vom W3C eingeführt, deren eindeutige Namen den einzelnen Bereichen einer Webseite eine Bedeutung zuweisen. Neu spezifiziert wurden die Elemente „header“, „footer“, „nav“, „main“, „article“, „section“, „aside“ und weitere. Die exakte Spezifikation dieser Strukturelemente ist auf den Internetseiten des W3C zu HTML 5.1 zu finden.[42]

<header>

Im Kopfbereich „header“ wird typischerweise der Titel der Webseite, eventuell ein Untertitel und ein Logo platziert. Auch die Hauptnavigation kann, muss sich aber nicht im “header“ wiederfinden. Der Anzeigebereich dieser Elementgruppe muss allerdings nicht waagerecht am oberen Rand einer Webseite positioniert sein. Eine senkrechte Anordnung am linken Rand ist zum Beispiel auch möglich. Ebenso kann ein „header“-Element mehrfach auf einer Seite vorkommen. Innerhalb des Hauptinhaltes beispielsweise kann es ein oder mehrere weitere „header“ geben.[43]

[42] Vgl. W3C, Spezifikation HTML 5.1, 23.01.2014

[43] Vgl. Zillgens, Responsive Webdesign, Seite 118

<footer>

Das Pendant zum Kopfbereich ist der Fußbereich „footer“. Hier können ergänzende Informationen, meist am untersten Rand der Webseite, aufgeführt werden. Typisch sind Copyright-Informationen, Links zum Impressum oder Soziale-Netzwerke-Buttons. Ebenso wie beim „header“ kann es auch beim „footer“ mehrere dieser Elemente auf einer Seite geben.

<nav>

Mit dem Element „nav“ sollte nur die Hauptnavigation ausgezeichnet werden, damit die Navigation nicht zu komplex wird und Nutzer von Screenreadern und Anwender, die zur Navigation die Tastatur benutzen, wichtige Navigationspunkte schnell ansteuern können. Die Navigation kann im „header“ enthalten sein. Eine Anordnung unterhalb des Kopfbereiches ist ebenfalls denkbar.

<main>

Mit dem Element „main“ wird der Hauptinhalt einer Seite ausgezeichnet. Dieses Element hat nach langen Diskussionen im W3C, das die HTML Spezifikation festlegt, Einzug in die HTML Version 5.1 gefunden.[44] Bis zu diesem Zeitpunkt war in HTML kein Element für den eigentlichen Inhalt der Seite vorhanden. Das Element „main“ darf auf einer Seite nur einmal vorkommen. Eine Platzierung innerhalb von „header“, „footer“, „nav“, „article“ oder „aside“ ist nicht erlaubt.

<article>

Ein für sich stehender, abgeschlossener Text wird in HTML5 in einem Element „article“ untergebracht. Der Inhalt dieses Bereiches muss auch herausgelöst aus seinem Umfeld verständlich sein. Dies könnte zum Beispiel ein Artikel einer Onlinezeitung sein, wenn auch die Namensverwandtschaft nicht darauf hinweist, dass in „article“ nur Zeitungsartikel eingefügt werden. In dem Element können zum Beispiel wiederum die Elemente „header“ und „footer“

[44] Vgl. Clark, html5 Doctor – The main element, 23.01.2014

verwendet werden, um den Titel und Kommentare zum Artikel unterzubringen.[45]

<section>

Das Element „section“ wird innerhalb des „main“-Elements verwendet, um Inhalte zu gliedern. Es können mehrere Elemente „article“ innerhalb von „section“ vorkommen. Auf das zuvor erwähnte Zeitungsartikel-Beispiel bezogen bedeutet dies, dass „section“ zum Beispiel für den Themenbereich Wirtschaft steht, worunter es mehrere Artikel gibt.

<aside>

Inhalte, die in ein Element „aside“ platziert werden, haben keine Bedeutung für den Hauptinhalt einer Seite. Wie „header“ und „footer“ sagt „aside“ nichts über die optische Anordnung dieses Bereiches aus. So wird das „aside“-Element zum Beispiel für hervorgehobene Texte, sogenannte Pull-Quotes, verwendet. Pull-Quotes sind aus dem Text entnommene Zitate, die zusätzlich zum fließenden Text ein weiteres Mal innerhalb des Textes platziert werden. Sie werden hervorgehoben und dekorativ dargestellt. Somit besitzen sie für das Verständnis des Hauptinhaltes keine Bedeutung und können zum Beispiel auf kleinen Smartphones ausgeblendet werden.

<figure>

Das Element „figure“ wurde eingeführt, um Bilder, die einen Hauptinhalt ergänzen, einzufügen. In „figure“ können Inhalte wie Bilder, Grafiken, Videos oder auch Code-Beispiele angezeigt werden. Ihre Position sollte für das Verständnis des Hauptinhaltes jedoch nicht von Bedeutung sein. Bei Umstrukturierungen auf responsiven Webseiten können diese Inhalte somit umplatziert oder zu einem späteren Zeitpunkt geladen werden. Zur Beschriftung eines Inhalts des „figure“-Elements kann vor oder nach diesem ein Element „figcaption“ eingefügt werden.

---

[45] Vgl. Gauchat, HTML5, CSS3, JavaScript, Seite 43

Die Grafik in Abb. 2-21 stellt eine mögliche Verwendung der genannten Elemente visuell dar. Juan Diego Gauchat führt das Schaubild in seinem Buch „HTML5, CSS3 & JavaScript“ auf.[46]

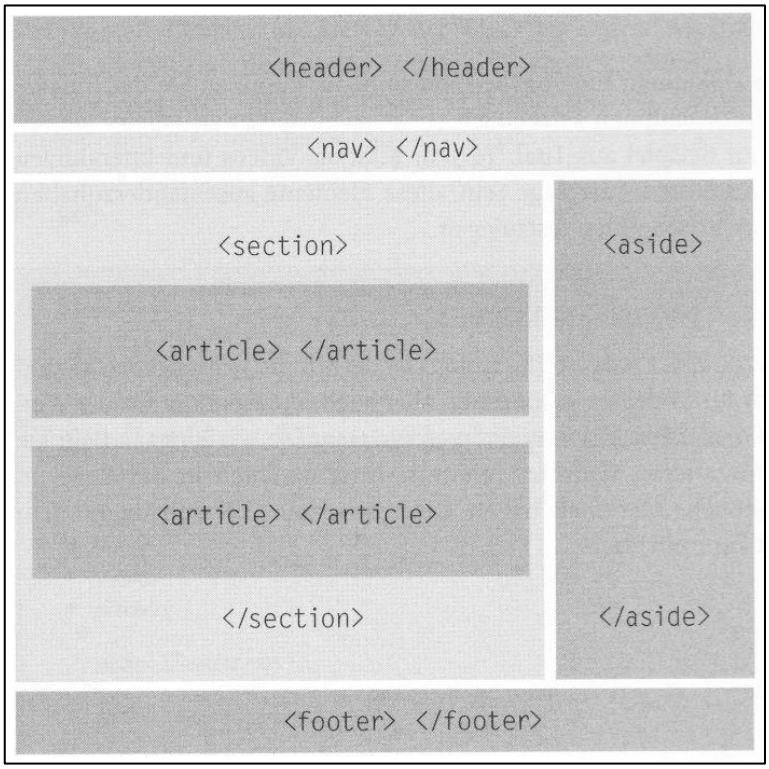

Abb. 2-21 Beispielhafte Darstellung der Verwendung der Elemente

Die semantische Auszeichnung bietet einige Vorteile für die folgenden Bereiche, die auch Peter Müller in seinem Buch identifiziert.[47]

Screenreader

Screenreader können Tasten oder Funktionen implementieren, die sofort zur Hauptnavigation springen und diese zur Auswahl vorlesen. Ebenso kann direkt zum Hauptinhalt gesprungen werden, ohne zuvor andere Informationen anzubieten.

Entwickler

Den Entwicklern ist es möglich, HTML unabhängig von Kommentaren zu gliedern. Dies ist besonders hilfreich für die Entwicklung von responsiven

[46] *Gauchat, HTML5, CSS3 & JavaScript, Seite 42*

[47] *Vgl. Müller, Flexible Boxes, Seite 78*

Webseiten, vor allem wenn es um eine Neuordnung der Inhalte des Layouts für verschieden große Anzeigeflächen geht. So kann ein im „aside“-Element befindlicher Text für kleine Geräte direkt als nachrangig identifiziert und zum Beispiel am Ende der Webseite platziert werden.

Browser

Browser könnten die semantischen Elemente nutzen, um Funktionen ähnlich denen der Screenreader anzubieten. Einen Button, der eine „Reader“-Funktion aufruft, bietet zum Beispiel der Browser Safari. Der Hauptinhalt der Webseite wird leserfreundlich angezeigt und Unwichtiges, wie schmückendes Beiwerk, Bilder oder Werbung, werden weggelassen.

Suchmaschinen

Eine präzisere Bewertung der Webseiten wird durch die kategorisierenden Elemente möglich. So können Begriffe aus dem „header“ oder „main“-Bereich für die Suchergebnisse höher bewertet werden als Begriffe in einem nachrangigen Bereich „aside“.

Nachteil dieser neuen Elemente ist jedoch, dass man zurzeit noch nicht davon ausgehen kann, dass jeder Browser diese Elemente verarbeiten kann. Generell gilt für Browser, dass unbekannte Elemente ignoriert werden und den restlichen Aufbau der Seite nicht beeinflussen sollen. Die Versionen 7 und 8 des Internet Explorers verstehen jedoch die neuen HTML5-Elemente noch nicht und erzeugen „HTML-Müll“, wie Peter Müller es nennt.[48] Als Entwickler kann man auf verschiedene Art und Weise auf diese Browser reagieren. Zum einen kann ein Hinweis, dass der Browser zu alt ist und der Inhalt nicht korrekt wiedergegeben wird, einhergehend mit der Bitte um Aktualisierung des Browsers, angezeigt werden. Um möglichst wenige Nutzer der Seite mit solchen Meldungen zu verärgern, sollte dieses Mittel jedoch nur bei sehr veralteten Browsern, die kaum noch benutzt werden, verwendet werden. Sollen eine Webseite oder –applikationen jedoch für einen Benutzerkreis erstellt werden, deren Standardbrowser zum Beispiel der Internet Explorer in der Version 8 ist, so sollte gegebenenfalls auf die neuen Elemente verzichtet werden. Um die Seite jedoch zukunftsfähig zu gestalten, ist es besser,

[48] Vgl. Müller, Flexible Boxes, Seite 98

HTML5-Elemente zu verwenden und für die nicht HTML5 fähigen Browser JavaScript Programme zu verwenden, die diese Browser HTML5 fähig machen. Ein solches JavaScript-Programm ist zum Beispiel „html5shiv“ (https://github.com/aFarkas/html5shiv, 23.01.2014).

2.5 Flexible Raster

Um ein in der Planung einer responsiven Webseite oder Webapplikation vorgesehenes flexibles Raster (englisch: „fluid grid“) beim Design im Browser und bei der späteren Implementierung umzusetzen, arbeitet man mit prozentualen Breitenangaben für die verschiedenen Elemente. Um die Anordnung von HTML-Elementen und deren Platzausnutzung zu verdeutlichen, folgt eine Beschreibung des CSS Boxmodells.

Ein Browser betrachtet jedes HTML-Element als einen separaten Kasten (englisch: box). So besteht eine Webseite aus vielen Boxen, die nach der HTML-Struktur und den vom Browser angewandten Regeln die eigentliche Webseite ergibt. Eine solche vom Browser zur Anzeige der Seite verwendete Regel ist zum Beispiel der hinterlegte Standardstyle für ein Element. Weiter gibt es die Regel, dass fast alle Strukturelemente, wie zum Beispiel „div“, „p“ und die zuvor erwähnten neuen Elemente wie „header“, „section“ und „nav“, als „Flow Content“ (vor HTML5: „Block-Element“) interpretiert werden. Dies bedeutet, dass jedes dieser Elemente in einer neuen Zeile platziert wird. Dabei nutzen die Elemente den maximal zur Verfügung stehenden Platz komplett aus und die Höhe der Box wird dem Inhalt angepasst. So würde ein „aside“ Element ohne weitere Formatierungsinformationen unterhalb des davor definierten Elementes über die volle Breite der Seite angezeigt.

Im Gegensatz zum „Flow Content“ werden Elemente der Kategorie „Phrasing Content“ (vor HTML5: Inline-Element) horizontal angeordnet. Bietet eine Zeile nicht ausreichend Platz für alle aufeinander folgenden Elemente, so erfolgt ein Zeilenumbruch. Beispiele für „Phrasing Content“ sind „button“, „img“, „label“ und „textarea“.

Die Einordnung der HTML-Elemente in die Content-Kategorien kann auf den Seiten des W3C nachgesehen werden.[49]

Dieses Verhalten der Elemente, wie auch weitere Eigenschaften zur Anzeige, lassen sich für die im HTML definierten Elemente per CSS definieren. Um „Flow Content“ Elemente horizontal anzuordnen, gibt es den Befehl „display:inline-block“. Soll ein waagerechtes Menü zum Beispiel als ungeordnete Liste (Element „ul“) implementiert werden, so würden die einzelnen Elemente der Liste (Element „li“) untereinander angeordnet. Per Befehl „display:inline-block“ werden die einzelnen Menüpunkte horizontal aufeinanderfolgend dargestellt.

Möchte man die in HTML5 neu eingefügten Strukturelemente verwenden und sicherstellen, dass auch ältere Browser, die diese Elemente nicht kennen, sie korrekt behandeln, so sollte man im CSS den Elementen explizit die Blockanordnung zuweisen, da sie sonst wie „Phrasing content“ angeordnet werden.

```
main, header, footer, aside, section, article <…> {
        display: block;
}
```

Möchte man einzelne Boxen nebeneinander anordnen, so gibt es hierfür eine im CSS festlegbare Eigenschaft „float“, zu der die Ausrichtung der Box innerhalb des zur Verfügung stehenden Bereiches angegeben werden kann. Die Syntax hier sieht folgendermaßen aus:

```
main {
        float: left;
}
aside {
        float: left;
}
```

Wird auf diese Art und Weise der Fluss der Elemente gesetzt, so würde ein im HTML nach dem „main“-Element vorhandenes „aside“-Element in der Webseite als zweite Spalte rechts von dem Hauptinhalt angezeigt. Voraussetzung ist, dass diese beiden Elemente nebeneinander ausreichend Platz finden. Ist dies nicht der Fall, werden sie untereinander angeordnet.

---

[49] Vgl. W3C, Spezifikation HTML 5.1, 23.01.2014

Damit ein nachfolgendes Element, wie zum Beispiel eine Fußzeile, korrekt unterhalb der beiden „float“-Elemente angeordnet wird, muss die Fließeigenschaft folgendermaßen initialisiert werden:

```
footer {
        clear: both;
}
```

Boxmodell

Am einfachsten lässt sich ein flexibles Raster umsetzen, indem man das mit CSS3 neu eingefügte Boxmodell „border-box“ für alle Elemente einstellt.

```
* {
        box-sizing: border-box;
}
```

Dabei wird die Breite eines Elementes entsprechend der angegebenen Prozentwerte oder auch Pixelwerte angezeigt, auch wenn für den Rahmen der Box („border“) und den Abstand des Inhaltes zum Rand der Box („padding“) Werte festgelegt wurden. Diese Werte werden von der Gesamtgröße ihrer Box abgezogen.

Dies war bei dem einzigen Boxmodell, das vor CSS3 existierte („content-box“), nicht der Fall. Die Breite einer Box definierte dort die Größe des eigentlichen Inhaltes. Der Abstand zum Rand und die Dicke des Rahmens mussten zur Ermittlung der Gesamtbreite noch aufaddiert werden. Dies war eine fast unmögliche Aufgabe, wenn sich die Art der Breitendefinition unterschied, zum Beispiel wenn die Dicke des Rahmens und der Abstand vom Rand in Pixel, die Breite der Box jedoch in Prozent des zur Verfügung stehenden Bereiches angegeben wurden.

Abb. 2-22 zeigt die beiden Boxmodelle im Vergleich.

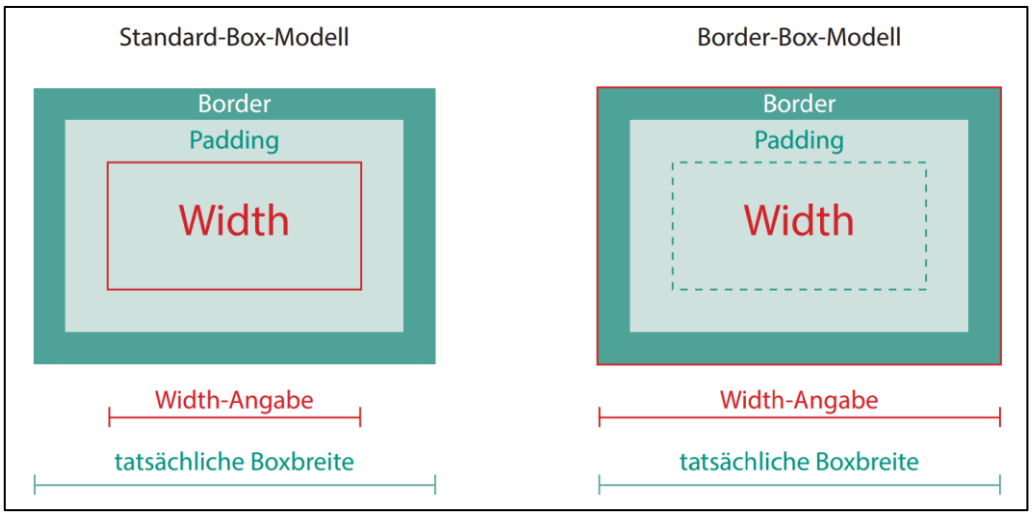

Abb. 2-22 Boxmodelle „content-box“ und „border-box“[50]

Eine mit diesen Mitteln aufgebaute Webseite mit Kopfbereich, Navigation, Hauptinhalt, Zusatzinformationen und Fußbereich würde sich im Browser so wie in Abb. 2-23[51] darstellen.

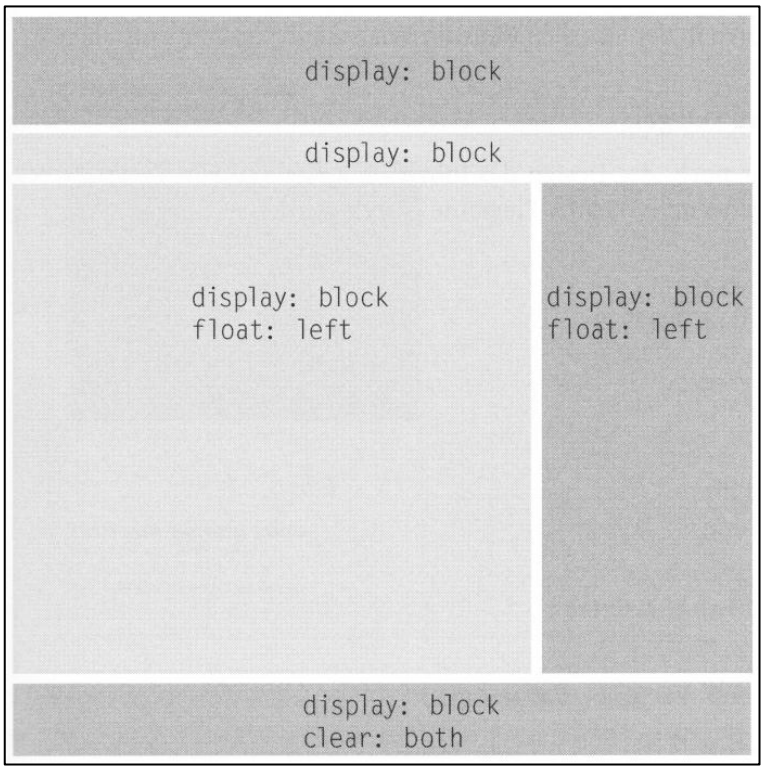

Abb. 2-23 Beispielhafte Darstellung der „float“-Eigenschaft

[50] *Zillgens, Responsive Webdesign, Seite 42*

[51] *Gauchat, HTML5, CSS3 & JavaScript, Seite 81*

2.6 Umbruchpunkte

Ein mit flexiblem Raster erstelltes Layout einer Webseite passt sich der zur Verfügung stehenden Anzeigefläche an. Für bestimmte Größen ist das daraus resultierende Layout einer Seite auch akzeptabel. Bei sehr großen oder kleinen Anzeigeflächen wird ein Layout jedoch unbrauchbar, das flexible Raster stößt an seine Grenzen. Während des Designs im Browser kann man die Größen, von denen an ein anderes Layout notwendig wird, durch Anzeige der Seite in einem Hilfsprogramm (englisch: tool) ermitteln. Hierzu kann man das Entwicklertool „Bildschirmgrößen testen“ des Browsers Firefox verwenden. Alternativ nutzt man eine die Fenstergröße simulierende Webseite (zum Beispiel http://www.active-value.de/responsive-design-viewer/, 23.01.2014). Bookmarklets, wie „Viewport Resizer“, bieten sich ebenfalls an. (http://lab.maltewassermann.com/viewport-resizer/, 23.01.2014). Alle genannten Tools teilen die Funktion, dass die Anzeigefläche (englisch: viewport) stufenlos per Maus verkleinert oder vergrößert werden kann. Auch lassen sich fest voreingestellte Größen auswählen. Ab Firefox in der Version 26 steht zusätzlich eine Simulation der Touch-Bedienung zur Verfügung.

Die Abb. 2-24 zeigt das Bookmarklet „Viewport Resizer“ mit der Seite des „Bundesministerium der Finanzen“ in verschiedenen Anzeigegrößen.

Ist man beim Design nach dem Prinzip „Mobile First“ vorgegangen, so lässt man sich die zu erstellende Webseite in einem der Tools in Smartphone-Größe anzeigen. Da man dieses Design entworfen hat, dürfte die Anzeige für kleine Geräte in Ordnung sein. Vergrößert man nun langsam die Anzeigefläche in dem gewählten Tool, so sieht man zeitgleich die Veränderung der Seite. An dem ersten Punkt, an dem die Seite unansehnlich wird, muss ein Umbruchpunkt (englisch: breakingpoint) festgelegt werden. Das bedeutet, dass ab dieser Bildschirmgröße, meist festgelegt durch die Breite in Pixel, ein anderes Layout verwendet werden muss. Umgesetzt werden kann dies per Mediaqueries, die im Anschluss erläutert werden. Zunächst betrachten wir jedoch, wo Umbruchpunkte notwendig und sinnvoll sind.

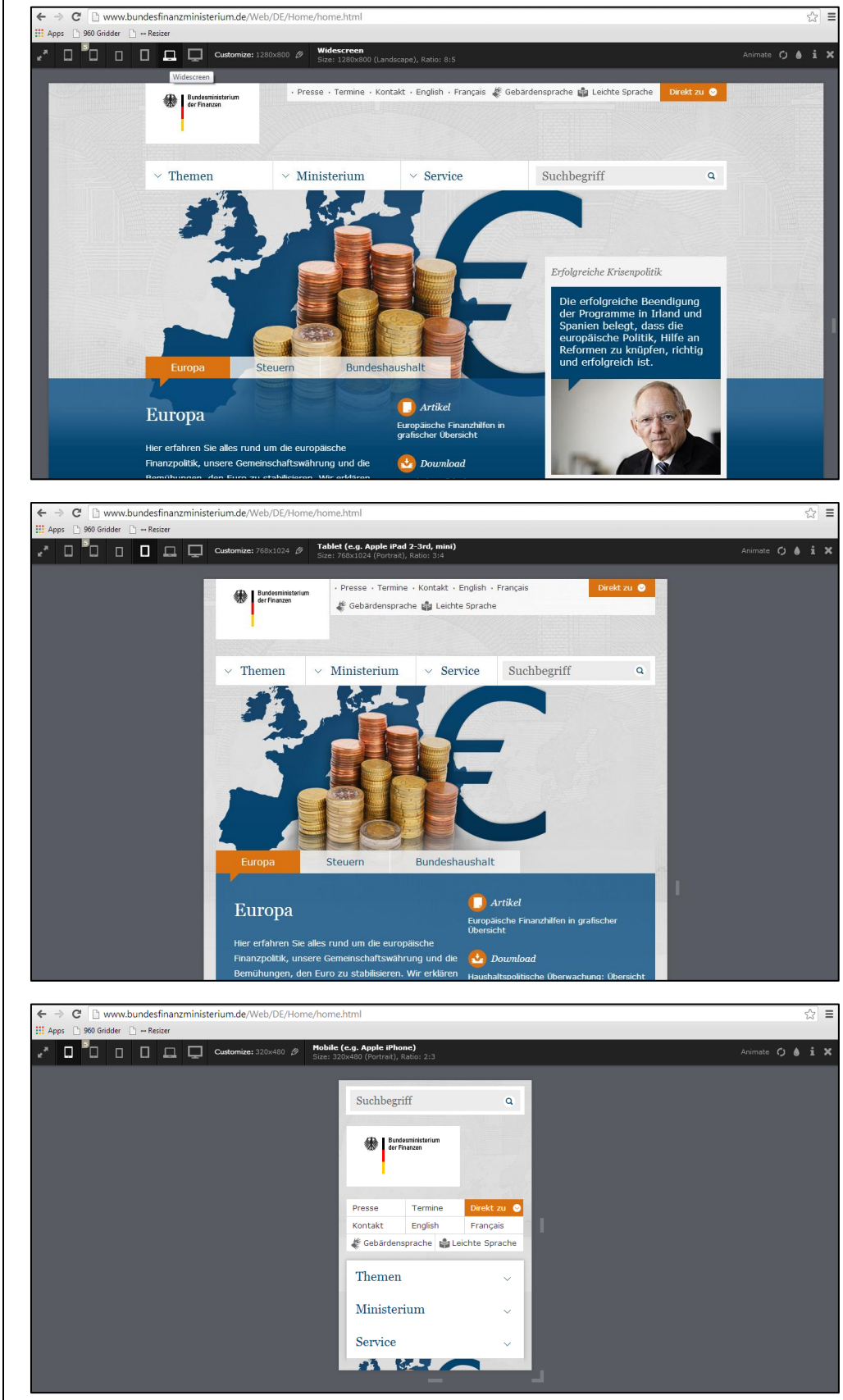

Abb. 2-24 „Viewport Resizer“ – Widescreen-, Tablet- und Mobile-View

Zum einen findet man Umbruchpunkte, die ein anderes Layout per CSS notwendig machen, indem man das Gesamtdesign der Webseite, wie gerade beschrieben, bei verschiedenen Anzeigegrößen betrachtet und darauf bezogene Umbruchpunkte festlegt. Weitere Hinweise für die Notwendigkeit das Layout zu verändern, ergeben sich aus der Länge der Schriftzeilen. Wird Schrift in sehr langen Zeilen angezeigt, so fällt es dem Leser schwer, vom Ende der Zeile an den Anfang der nächsten Zeile zu springen. Sehr kurze Zeilen machen häufige Sprünge mit den Augen notwendig. Beides stört den Lesefluss und sollte verhindert werden. Eine angenehm zu lesende Zeilenlänge liegt zwischen 45 und 65 Zeichen, wie Ryan Baker an der Wichita State Universität analysierte.[52] Um diesen optimalen Zeilenumbruchbereich zu markieren, kann man an den Zeichenpositionen 45 und 65 ein Sternchen („*“) einfügen, wie Trent Walten es beschreibt.[53] Verändert man nun die Größe der Seite mit den oben beschriebenen Tools, erkennt man, wann die Zeilenlänge nicht mehr optimal ist. Der Umbruch erfolgt nicht mehr zwischen den beiden Sternchen. Mit Umbruchpunkten lassen sich an solchen Stellen die Zeilenlängen optimieren. Zum Beispiel kann bei zu langen Zeilen mittels Mediaqueries in ein mehrspaltiges Layout gewechselt werden. Auch kann man die Darstellungsbreite der Webseite nach oben hin begrenzen und somit die maximale Zeilenlänge beschränken. Alternativ kann man die Schriftgröße anpassen. Jedoch ist auch dabei auf eine weiterhin gute Lesbarkeit zu achten.

Für die effiziente Entwicklung einer Webseite oder –applikation ist es wichtig, so wenig wie möglich Umbruchpunkte zu definieren. Denn jeder Wechsel auf ein anderes Layout macht die Gestaltung und Implementierung eines weiteren Layouts notwendig. Somit fällt bei jedem weiteren Umbruchpunkt zusätzlicher Aufwand an. Die für die Benutzerfreundlichkeit notwendigen und nicht mit anderen Mitteln des responsiven Designs zu realisierenden Anpassungen müssen jedoch umgesetzt werden.

---

[52] Vgl. Baker, Is Multiple-Column Online Text Better? It Depends!, 23.01.2014

[53] Vgl. Walton, Fluid Type, 23.01.2014

2.7 Mediaqueries

Die Umsetzung der sich an bestimmten Umbruchpunkten verändernden Layouts, können mit Hilfe der in CSS3 definierten Mediaqueries realisiert werden.

Zunächst einmal war es in CSS2 möglich, den Medientyp des aufrufenden Systems abzufragen und darauf mit unterschiedlichen Layouts zu reagieren. Dabei konnten folgende Werte geprüft werden: „screen“, „print“, „projection“, „tv“ und „handheld“. Wirklich genutzt, das heißt von den Browsern auf verschiedensten Geräten als Information zurückgegeben, wurden allerdings nur „screen“ und „print“. „screen“ bedeutet, die Anzeige findet auf einem Display statt. „print“ sagt aus, dass der Inhalt auf dem Drucker ausgegeben werden soll. In der Druckansicht eines Browsers lässt sich ein für den Druck definiertes Layout betrachten. Der Medientyp „handheld“ wäre für Smartphones sicherlich passend, jedoch setzte sich dieser Medientyp nicht durch. Das erste Smartphone, das iPhone, gab den Typ „screen“ zurück.[54] Dies hat sich auch für die heutigen Smartphones nicht geändert. Somit lässt sich mit der Mediatyp-Abfrage allein nicht auf unterschiedliche Anzeigegrößen reagieren.

Mit CSS3 wurden die Mediaqueries eingeführt. Hiermit ist nun unter anderem eine Abfrage der Breite des Browserfensters möglich. Im Stylesheet kann dann abgefragt werden, ob ein aufrufendes System eine Mindest- oder Maximalbreite besitzt.

Hat man bei der Ermittlung der Umbruchpunkte festgestellt, dass zum Beispiel ab einer Breite von 800 Pixeln und zusätzlich ab 1060 Pixeln andere Layouts notwendig werden, so kann man die Verwendung unterschiedlicher CSS-Layouts im HTML folgendermaßen veranlassen:

```
<link rel=“stylesheet“ media=“screen and (max-width:799px)“
href=“mini.css“>
<link rel=“stylesheet“ media=“screen and (min-width:800px) and
```

[54] *Vgl. Müller, Flexible Boxes, Seite 300*

```
(max-width:1059px)“ href=“mid.css“>
<link rel=“stylesheet“ media=“screen and (min-width: 1060px)“
href=“max.css“>
```

In einer CSS-Datei selbst können ebenfalls die Mediaqueries eingesetzt werden, um Eigenschaften bei bestimmten Bildschirmgrößen zu setzen. Der Befehl „@media“ kann in einem Stylesheet mehrfach vorkommen, darf jedoch nicht verschachtelt werden.

```
@media screen and (max-width:799px) {
  /*CSS Befehle*/
}
@media screen and (min-width:800px) and (max-width:1059px){
  /*CSS Befehle*/
}
@media screen and (min-width: 1060px){
  /*CSS Befehle*/
}
```

Um den korrekten Aufruf der unterschiedlichen Stylesheets zu testen, kann man die Webseite zum Beispiel im Browser Firefox mit dem enthaltenen und schon erwähnten Entwicklertool „Bildschirmgröße testen“ in verschiedenen Größen betrachten. Schaltet man währenddessen die „Web-Entwickler“-„Werkzeuge“ ein und wählt „Inspektor“ und den Reiter „Regeln“ aus, so sieht man, wann welches CSS herangezogen wird. Dies funktioniert auch für die Seiten des „Bundesministerium der Finanzen“, wie in Abb. 2-25 zu sehen ist (http://www.bundesfinanzministerium.de, 23.01.2014).

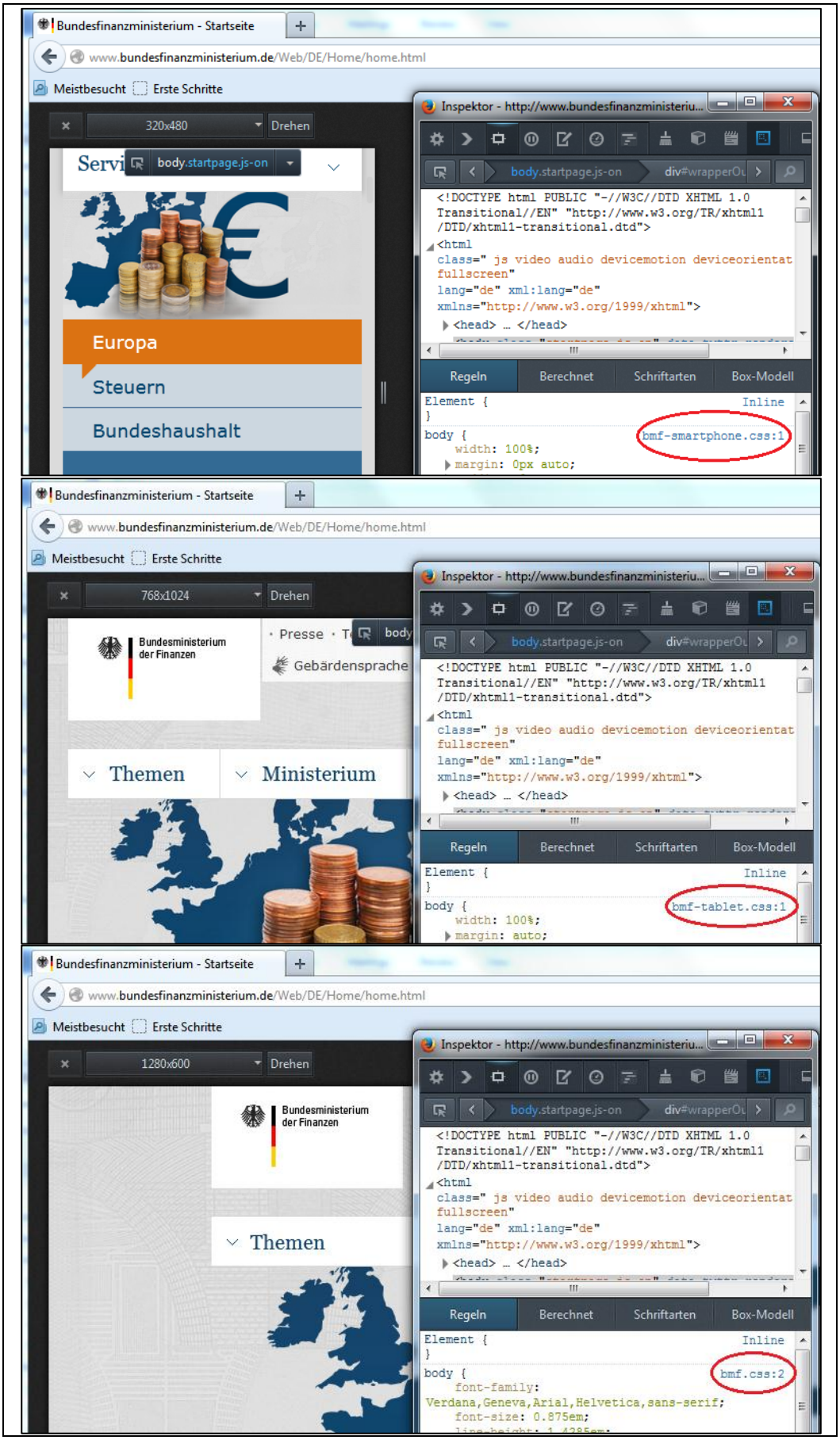

Abb. 2-25 Verwendung unterschiedlicher CSS

Auf den Seiten des „Bundesministerium der Finanzen“ werden korrekt, abhängig von der Darstellungsbreite, die unterschiedlichen CSS herangezogen. Dennoch könnte es sein, dass das Smartphone-Stylesheet auf einem realen Kleingerät nicht korrekt angezeigt wird, denn Smartphones geben ohne spezielle Befehle im HTML eine andere als die reale Breite zurück. Möchte man die tatsächliche Breite der Anzeigefläche erhalten, so muss dies im „head“-Element des HTML-Dokumentes explizit gesagt werden. Folgender „meta“-Befehl realisiert dies:

<meta name=“viewport“ content=“width=device-width“>

Dass nicht die reale Breite des Bildschirms von Smartphones zurückgegeben wird, hat historische Gründe. Als das erste Smartphone, das iPhone, 2007 eingeführt wurde, gab es kaum Webseiten, die auf kleine Displays angepasst waren. Das iPhone meldete eine Breite von 980 Pixeln und verkleinerte die so gerenderte Seite auf die reale Bildschirmbreite von 320 Pixeln. Dieses Verfahren hat sich bei allen Smartphones durchgesetzt und wird auch heute noch verwendet. Damit die Abfragen per Mediaquery funktionieren, muss also der Viewport („width“) auf die reale Bildschirmbreite („device-width“) gesetzt werden.[55]

Alle gängigen Browser unterstützen in der aktuellsten Version CSS3 und somit die Media-Abfragen, wie „min-width“ und „max-width“.[56] Browser, die CSS3 nicht unterstützen, werden kaum noch genutzt. Auf einfache Weise lässt sich für diese alten Programme jedoch sicherstellen, dass das Layout brauchbar dargestellt wird. Zunächst legt man fest, welches CSS für nicht CSS3 kompatible Browser verwendet werden soll. Hier ist es sinnvoll, das Layout, das auf Desktop-Monitoren zum Tragen kommt, zu verwenden. Denn die Browser auf Smartphones beherrschen meist CSS3 und auf einem größeren Gerät, wie zum Beispiel einem Fernseher, wird wahrscheinlich kein veralteter Browser verwendet werden. Um sicherzustellen, dass bezogen auf das Beispiel das mittlere Layout verwendet wird, fügt man in den anderen Mediaqueries den Befehl „only“ wie folgt ein:

---

[55] Vgl. Müller, Flexible Boxes, Seite 305

[56] Vgl. Deveria, Can I use – CSS3 Media Queries?, 23.01.2014

```
@media only screen and (max-width:799px) {
  /*CSS Befehle*/
}
```

Ein Browser, der nicht CSS3 fähig ist, kennt den Befehl „@media only“ nicht und ignoriert somit den restlichen Teil des Befehls. Das enthaltene CSS wird nicht herangezogen. Ohne das Einfügen des Befehls „only“ würde das enthaltene CSS gegebenenfalls angewendet, da „@media screen“ ein bekannter Befehl ist und lediglich das Prüfen der maximalen Breite nicht berücksichtigt würde.[57] So lässt sich die Webseite nach dem Prinzip „Progressive Enhancement“ implementieren.

Leider benutzen zurzeit noch 7 Prozent der Internetnutzer den Internet Explorer 8[58], der CSS3 zwar teilweise, jedoch CSS3-Mediaqueries nicht unterstützt. Ist eine Abfrage zwingend auch für ältere Browser umzusetzen, so kann man Polyfills verwenden. Ein Polyfill ist nachladbarer Code, der fehlende Browserfunktionalität durch JavaScript nachrüstet. Das Polyfill „Respond“ von Scott Jehl besteht aus JavaScript-Funktionen und rüstet unter anderem die Mediaquery-Optionen „min-width“ und „max-width“ nach (https://github.com/scottjehl/Respond, 23.01.2014). Auch im Browser nicht zur Verfügung stehende CSS Befehle kann man mit Polyfills nachrüsten. Auf der Internetseite „HTML5 please“ ist zu sehen, welche Polyfills es gibt und für welche Browser sie funktionieren (http://html5please.com/, 23.01.2014).

Meist werden Mediaqueries zur Abfrage der Breite des Viewports, also des inneren Anzeigebereiches des Browsers, verwendet. Die folgenden weiteren Abfragen sind jedoch auch möglich:

„height“
Die Höhe des Viewports. Auch als „min-height“ und „max-hight“ abfragbar.

„orientation“
Gibt an, ob das Gerät im Hochformat („portrait“) oder Querformat („landscape“) verwendet wird.

---

[57] Vgl. Müller, Flexible Boxes, Seite 302

[58] Vgl. StatCounter Global Stats, Browser Version, 23.01.2014

„device-width“

Gibt die Breite der Anzeige oder beim Medientyp „print“ des druckbaren Bereiches an. Auch als „min-device-width“ und „max-device-width“ abfragbar.

„device-height“

Gibt die Höhe des Mediums an. Auch als „min-device-height“ und „max-device-height“ verwendbar.

„device-aspect-ratio“

Gibt zum Beispiel „16/9“ zurück.

„color“

Gibt zurück, ob es sich um ein Farbdisplay handelt.

3 Performance-Optimierung

Die Effizienz einer Webseite oder einer Webapplikation spielt für die Akzeptanz beim Anwender eine große Rolle. Für die Nutzung vom Desktop aus gibt es kaum Performance-Probleme, da die Breitbandanbindung sehr verbreitet ist und hierüber Webseiten schnell geladen werden. Durch die schnelle Internetanbindung sind in den letzten Jahren Webseiten entstanden, bei denen nicht mehr, wie zu Zeiten des Modems, auf geringen Datenumfang geachtet wurde. Auch wenn man dieses hohe Datenvolumen beim Laden der Seite kaum bemerkt, so hat die Größe doch einen negativen Einfluss. Die Suchmaschine Google berücksichtigt die Ladezeit einer Webseite bei der Auswertung der Suchergebnisse. Schnelle Seiten werden weiter oben im Suchergebnis gelistet als langsamere. Ein geringer Datenumfang hat auch Vorteile für die Webseite auf Geräten mit schneller Datenverbindung. Im mobilen Kontext laden Seiten mit großem Datenvolumen jedoch nur sehr langsam. Im Rahmen der Erstellung einer responsiven Seite ist besonders auf eine gute Performance zu achten, damit die Seite auch auf Mobilgeräten zügig geladen werden kann und die Benutzerfreundlichkeit gegeben ist. Denn zu der langsameren Netzanbindung kommen dort weitere Verzögerungen durch die mobilen Übertragungsverfahren, wie „General Packet Radio Service“ (GPRS) und „Universal Mobile Telecommunications System“ (UMTS) hinzu. Dies wird nachfolgend erläutert.

Möchte man sicher gehen, dass die zu erstellende Webseite oder -applikation mit allen Übertragungstechniken möglichst schnell zu laden ist, zeigt sich wieder ein Vorteil des Vorgehens nach dem Prinzip „Mobile First“. Startet man mit d er Erstellung für Smartphones oder sonstige mobil verwendete Geräte, so leitet dies automatisch die Aufmerksamkeit auf die Performance. Für die Darstellung und Funktionalität auf größeren Anzeigegeräten kann man zwar weitere, Datenvolumen intensivere Elemente einbinden, jedoch sollte dies nur erfolgen, wenn es unbedingt notwendig ist. Eine möglichst schlanke Seite ist generell und nicht nur bei kleinen Geräten vorteilhaft, denn ein Anwender mit einer großen Anzeigefläche könnte zum Beispiel auch mit einem Laptop per mobiler Datenübertragung zugreifen.

Doch welche Faktoren spielen für die Performance eine Rolle? Die Höhe des Datenvolumens, das zum Aufbau einer Seite benötigt wird, ist hier das

Offensichtlichste. Eine hohe Anzahl und hochauflösende Bilder sind sicherlich mit verantwortlich, wenn eine Webseite träge geladen wird. Eine wichtigere Rolle spielt jedoch die Gesamtanzahl der Dateien, die übertragen werden muss, um die Seite aufzubauen. Werden Ressourcen aus vielen einzelnen Bild-, CSS- und JavaScript-Dateien herangezogen, iFrames und Buttons zu sozialen Netzwerken verwendet, so wird leicht eine große Anzahl zu ladender Dateien erreicht.

Die Bedeutung der Anzahl der Dateien für die Ladezeit einer Webseite ist groß. 1999 wurde in der Spezifikation des Hyper Text Transfer Protocols (HTTP) Version 1.1 festgelegt, dass von einem Endanwendersystem lediglich zwei Verbindungen zu einem Server parallel aufgebaut sein durften.[59] Heute können die meisten Browser zumindest sechs, der Internet Explorer 10 sogar acht parallele Requests stellen.[60] Diese Begrenzung der parallelen Anfragen soll dem Schutz der Server vor einer extrem hohen Zugriffszahl dienen und eine Überlastung vermeiden. Es hat jedoch zur Folge, dass nur sechs bis acht Dateien parallel beim Server abgerufen werden können. Erst wenn die ersten Dateien komplett übertragen wurden, können Anfragen für weitere Dateien gestartet werden. So entsteht zeitlich gesehen ein wasserfallartiger Ladeprozess. Diesen Prozess kann man sich in verschiedenen Browsern, wie zum Beispiel dem Chrome oder dem Firefox, unter den Entwicklertools auf dem Reiter „Netzwerk“ ansehen. Auch die Erweiterung zum Firefox namens Firebug bietet eine solche Ansicht. Diese Ansicht wird beispielhaft für das Laden der Webseite des „Bundesministerium der Finanzen“ in Abb. 3-1 gezeigt (http://www.bundesfinanzministerium.de, 20.01.2014).

[59] *Vgl. W3C, Hypertext Transfer Protocol -- HTTP/1.1, 26.01.2014*

[60] *Vgl. Seitenreport, Grundlagen der Website Performance , 26.01.2014*

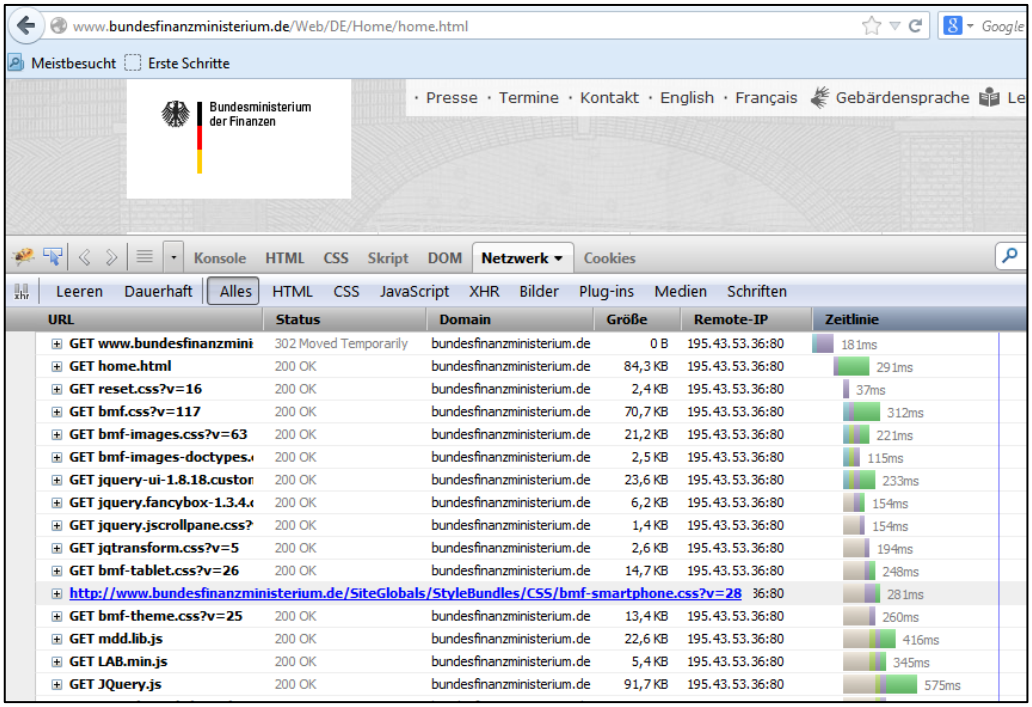

Abb. 3-1 HTTP Request- und Ladezeit-Diagramm

Zunächst wird die Datei „home.html“ geladen, anschließend die darin eingebundenen CSS-Dateien. Darauf folgen JavaScript-Dateien, später, außerhalb des hier sichtbaren Bereiches, Bilder im „gif“-, „png“- und „jpg“-Format und Weiteres. Man erkennt, dass nach dem Laden der „home.html“ in der ersten Zeile der Grafik zeitgleich der Request von 6 weiteren Dateien startet. Die darauf folgenden Dateien warten zunächst auf freie Verbindungen, gekennzeichnet wird dies durch den grauen Bereich des Balkens. Bewegt man im Firebug die Maus über den Balken einer Datei, so werden zusätzliche Informationen angezeigt. Abb. 3-2 zeigt die Daten zu der letzten Zeile des vorangehenden Beispiels, der Zeile für das JavaScript JQuery.js. 237 Millisekunden nach dem Aufruf der Webseite startet der Verbindungsaufbau für die Datei JQuery.js. Es folgt der Domain Name System (DNS) -Lookup zur Ermittlung der IP-Adresse aus der Uniform Resource Locator (URL), der Verbindungsaufbau zum Zielserver, das Senden der Anfrage, die Wartezeit bis zur Antwort und letztendlich das Empfangen der angeforderten Daten.

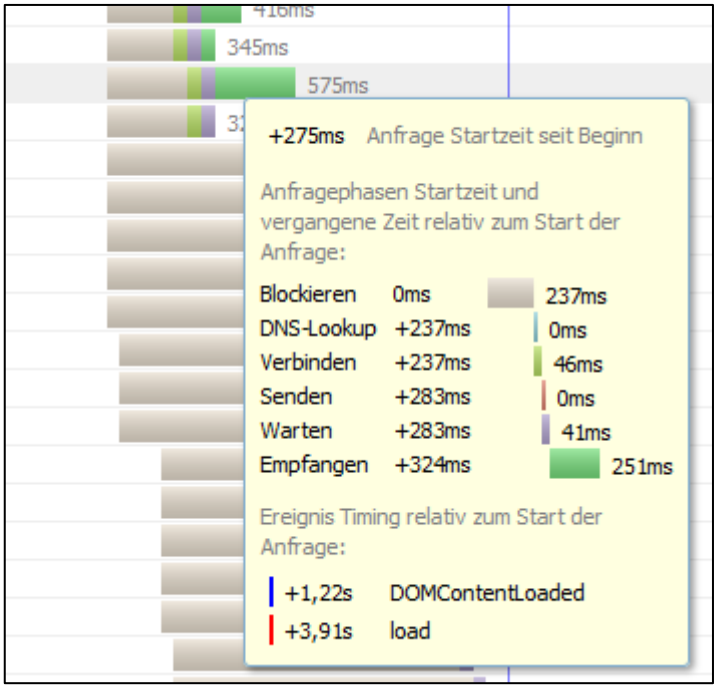

Abb. 3-2 Detailansicht der Ladezeit der Datei JQuery.js

Der in beiden Abbildungen (Abb. 3-1 und Abb. 3-2) sichtbare blaue senkrechte Balken zeigt den Zeitpunkt an, ab dem die zur Anzeige der Seite notwendigen Daten geladen sind. Zu diesem Zeitpunkt wird vom Browser das Event „DOMContentLoaded“ geworfen. Je größer die Zeitspanne bis zum blauen Balken ist, desto länger muss der Nutzer auf die Anzeige der Seite warten. Zusätzliche Daten werden auch hinter der blauen Linie noch geladen. Außerhalb des Bereiches der Abbildungen gibt es rechts von der blauen Linie noch eine rote. Diese zeigt den Zeitpunkt an, zu dem der Seiteninhalt vollständig geladen ist und das Ladesymbol verschwindet. Das Event „onload“ wird geworfen.

Der Verbindungsaufbau, der bis zum Start der Übertragung eines Dateiinhalts notwendig und in Abb. 3-2 beispielhaft dargestellt ist, fällt für jede Datei an. Die Zeit, die für diesen Verbindungsaufbau benötigt wird, nennt man „Round Trip Time“ (RTT). Darin enthalten ist noch nicht die Übertragung des Dateiinhaltes selbst. Die „Round Trip Time“ wird unter anderem durch die folgenden Faktoren beeinflusst:

- die physische Distanz vom aufrufenden System hin zum Server
- die Anzahl der Knotenpunkte bis hin zum Server
- die physikalische Grenze bezüglich der Geschwindigkeit durch die Art des Übertragungsmediums (Kabel, optisches Kabel, Drahtlosnetzwerk, „Digital Subscriber Line“ (DSL), UMTS…)

- die Anzahl der Anfragen am Server

Die Datenübertragungsrate jedoch spielt kaum eine Rolle für die „Round Trip Time“. Erst bei der Übertragung der eigentlichen Daten führt eine Breitbandverbindung zur schnelleren Übertragung, da dabei mehrere Verbindungen parallel zur Übertragung des Dateiinhaltes genutzt werden, wie der Name schon erkennen lässt. Das Übertragungsmedium beeinflusst im Gegensatz dazu die „Round Trip Time“ stark. Christoph Zillgens führt in seinem Buch folgende Round-Trip-Verzögerungen bezüglich der Leitungstechniken an:[61]

Kabelnetzwerk	1 ms
Drahtlosnetzwerk	10 ms
DSL	40 ms
UMTS	350 ms
GPRS/EDGE	850 ms

Hierdurch wird deutlich, dass die Anzahl der Dateien einer Webseite gerade für die Übertragung auf mobile Geräte von größter Bedeutung ist. Zwar liegt die „Round Trip Time“ für eine Datei meist im Millisekunden-Bereich, aber nimmt man alle Dateien einer Seite zusammen, so kann die aufaddierte Zeit durchaus mehrere Sekunden betragen, ohne dass in dieser Zeit die eigentliche Datenübertragung enthalten wäre. Betrachtet man hierzu Auswertungen über den Dateiumfang von Webseiten, so wird einem die Bedeutung der „Round Trip Time“ bewusst. Die Top 1.000 Webseiten im Jahr 2012 benötigten im Durchschnitt etwa 100 Dateien. Diese enthielten 1.114 Kilobyte (kB). Die Entwicklung des Datenumfangs der Webseiten über die letzten Jahre zeigt folgende Grafik der Seite „Website Optimization“ (Abb. 3-3)[62].

[61] *Vgl. Zillgens, Responsive Webdesign, Seite 320*

[62] *Vgl. Website Optimization - Average Web Page Size Triples…, 23.01.2014*

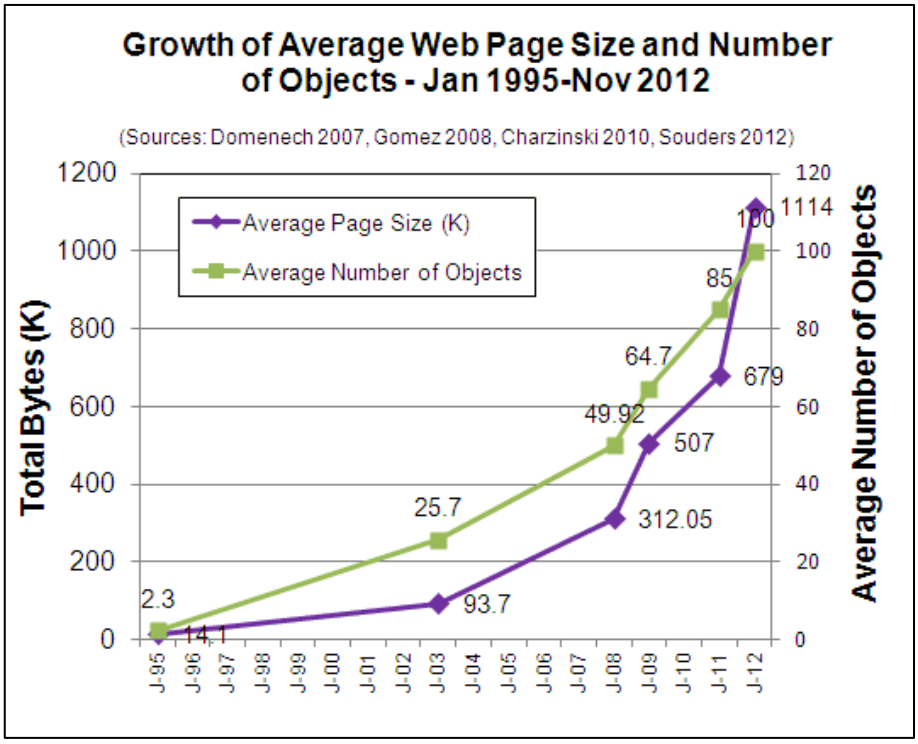

Abb. 3-3 Anstieg der Anzahl der Dateien und des Datenvolumens

Ursächlich für eine hohe Anzahl an Dateien einer Webseite sind zum Teil sicherlich die eingebundenen Bilder. Des Weiteren werden CSS-Inhalte und auch JavaScripts gerne modularisiert in Dateien gespeichert, da sich so die Wiederverwendbarkeit und die Wartbarkeit erhöht. Oft bringen integrierte Plugins, wie z.B. jQuery für Ligthboxen, zusätzlich eigene Stylesheet-Dateien mit ein. Werden für die Seite Polyfills benötigt, um die Anzeige auch für ältere Browser zu ermöglichen, so kommen hierdurch weitere Dateien hinzu.

Es ist daher beim responsiven Webdesign aufgrund der hohen „Round Trip Time“ auf Mobilgeräten besonders wichtig, die Anzahl der Dateien möglichst gering zu halten. Wie dies zu realisieren ist, ohne auf Funktionalität und Layout verzichten zu müssen, wird in den folgenden Kapiteln erläutert.

3.1 JavaScript-Dateien zusammenfassen

Werden mehrere JavaScripts für die Webseite oder –anwendung benötigt, so können diese einfach in einer Datei zusammengefasst und in einem Befehl eingefügt werden. Dieses Zusammenfügen kann manuell erfolgen, indem man das bis dahin existierende JavaScript aus Einzeldateien in eine zusammenfassende Datei hineinkopiert. Dabei muss auf die identische Reihenfolge der Skripte geachtet werden, da sie gegebenenfalls aufeinander aufbauen.

Möchte man auf den Komfort der besseren Auffindbarkeit und die Möglichkeit, einzelne Skripte komfortabel auch in andere Seiten einbinden zu können, nicht verzichten, kann man auch ein serverseitiges Skript, das die JavaScripts zusammengefasst zurück gibt, verwenden. Christoph Zillgens führt hierfür in seinem Buch ein PHP-Skript an.[63]

3.2 CSS-Dateien zusammenfassen

Stylesheets werden ebenfalls gerne modularisiert und in mehreren Dateien abgespeichert. Bei Verwendung von Mediaqueries für die Auswahl von Stylesheets könnte der Eindruck entstehen, dass nur die Dateien mit zutreffender Abfrage geladen werden. Dies ist jedoch nicht der Fall. Alle in einer Seite eingebundenen Stylesheets werden beim Aufruf der Webseite geladen. Lediglich wird ihr Inhalt nicht zur Aufbereitung der Anzeige herangezogen. Deswegen können auch CSS-Dateien händisch vor der Bereitstellung auf dem Server zusammengefasst und mit einem Import-Befehl in das HTML eingebunden werden. Jedoch leidet auch hier die Wartbarkeit unter diesem Verfahren. Muss ein CSS aktualisiert oder korrigiert werden, so muss dies in der zusammengefassten Datei erfolgen. Oder die einzelnen Dateien werden weiterhin für die Entwicklung genutzt, wobei dann vor einem Test immer wieder die CSS-Dateien zu einer Datei zusammengefasst werden müssen.

Somit bietet es sich auch bei CSS Dateien an, die Entwicklung in separaten Dateien durchzuführen und lediglich einen Aufruf eines Skriptes, das die Dateien serverseitig zusammenfasst, für das Abrufen der Seite zu verwenden. Bei diesem Verfahren wird das Datenvolumen der CSS-Befehle nicht optimiert, lediglich die Anzahl der Dateien reduziert sich.

Um zusätzlich das Datenvolumen positiv zu beeinflussen, hat Scott Jehl das JavaScript „eCSSential“ (https://github.com/scottjehl/eCSSential, 23.01.2014) erstellt. Anstatt wie bei einer Bildschirmgrößen-Abfrage per Mediaquery alle möglichen CSS-Dateien zu laden, ermöglicht sein Skript eine Abfrage der Bildschirmgröße per JavaScript. Anschließend wird nur das CSS geladen, für das die Bedingung gültig ist. „eCSSential“ fasst jedoch noch nicht automatisch

[63] *Zillgens, Responsive Webdesign, Seite 323*

einzelne CSS-Dateien, die gegebenenfalls in dem ausgewählten CSS verwendet werden, zusammen. Man kann jedoch zusätzlich einstellen, dass „file concatenation“ verwendet werden soll, wodurch dann die Dateien zusammenführt werden. Hierfür wird in „eCSSential“ auf Server-Seite das Tool „QuickConcat“ (https://github.com/filamentgroup/quickconcat, 23.01.2014) verwendet.

Bei der Verwendung des Skriptes „eCSSential“ im Rahmen von „Progressive Enhancement“ ist jedoch darauf zu achten, dass auch Browser, die nicht JavaScript fähig sind, mit den notwendigen Informationen zur Aufbereitung der Seite versorgt werden.

3.3 JavaScript- und CSS-Dateien minifizieren

Unter minifizieren versteht man, Dateien vor der Produktionseinführung so kompakt und zeichensparend wie möglich zu gestalten. Hierbei werden alle im JavaScript oder CSS enthaltenen Kommentare und überflüssige Zeichen, wie Einrückungen und Zeilenumbrüche, entfernt. So wird das Datenvolumen auf dem Server und vor allem für die Datenübertragung reduziert.

Direkt bei der Entwicklung macht es wenig Sinn, datenvolumensparend zu arbeiten, da der Code dadurch seine Wartbarkeit verliert.

Vor dem Einsatz der Software in der produktiven Umgebung bietet es sich jedoch an, ein Tool zum Minifizieren der Dateien einzusetzen. Nicholas C. Zakas beschreibt in zwei Artikeln bei „A List Apart“ das Tool „YUI Compressor“ (http://yui.github.io/yuicompressor/, 23.01.2014) zur Minifizierung.[64] Dieses Tool entfernt nicht nur überflüssige Zeichen, sondern es verkürzt auch Variablennamen auf ein bis zwei Zeichen, um so eine größere Volumeneinsparung zu erreichen.

3.4 Dateien für die Übertragung komprimieren

Zusätzlich zu den voran genannten Verfahren, lassen sich Dateien generell für die Übertragung serverseitig komprimieren. Dies kann mit dem GZIP-Verfahren realisiert werden, das gängige Webserver, wie Apache Tomcat

[64] *Vgl. Zakas, Better JavaScript Minification und -Part II, 23.01.2014*

oder WebSphere Application Server, als auch Browser beherrschen. Auf dem Server werden Dateien oder Datentypen, für die man eine Komprimierung konfiguriert hat, per GZIP gepackt, so komprimiert übertragen und im Browser wieder entpackt. Gerade bei ASCII-Dateien, wie es HTML-, CSS-, XML- oder JavaScript-Dateien sind, lassen sich gute Komprimierungsergebnisse erzielen und somit das Übertragungsvolumen reduzieren.

3.5 Caching optimieren

Am Webserver lässt sich einstellen, wie lange die heruntergeladenen Dateien im Browser oder auf einem Proxyserver gespeichert werden sollen. Der Server gibt diese Information in der Antwort, genauer im HTTP-Response, an Proxyserver und Browser zurück. Ist bei einem erneuten Aufruf einer Datei, zum Beispiel beim Navigieren zu einer Unterseite, die Datei noch gültig, so wird sie wiederverwendet. Eine lange Vorhaltedauer im Cache spart HTTP-Requests für wiederholt aufgerufene Seiten ein. Dieses Ablaufdatum kann für alle Dateitypen eingestellt werden. Besonders sinnvoll ist dies für Bilder, die sich nicht oft ändern und ein verhältnismäßig großes Dateivolumen besitzen. So machen Vorhaltezeiten von 14 Tagen bis hin zu einem Jahr oder länger Sinn. Auch eingebundene JavaScripts einer ausgereiften Webseite werden sich nicht so häufig ändern und müssen nicht ständig neu geladen werden. Das Gleiche gilt für die Stylesheets.

Möchte man sicher gehen, dass Dateien definitiv neu geladen werden, sobald sie angepasst wurden, so kann man beim Einbinden der Datei im HTML der URL eine Versionsnummer mitgeben und diese hochzählen, wenn die zu verwendende Datei verändert wurde.

```
<link rel="stylesheet" type="text/css" href="style.css?v=1.1" />
```

Auf diese Weise kann man, ohne Sorge um eine inkonsistente Seite haben zu müssen, lange Caching-Zeiten ausnutzen.

Allerdings ist dieses manuelle Verfahren fehleranfällig. Leicht kann das Hochzählen der Dateiversion vergessen werden. Um dieses Problem zu umgehen, schlägt Christoph Zillgens in seinem Buch die Verwendung des folgenden PHP-Skripts vor, das anstatt einer Versionsnummer automatisch

den Timestamp der Datei in die URL einfügt. So wird die Datei definitiv neu geladen, sobald sie aktueller als die Version im Browsercache ist.[65]

```
<?php
echo ‘<link rel=“stylesheet“ type=“text/css“
href=“style.css?timestamp=‘.filemtime(‘style.css‘).‘“ />‘
?>
```

Steve Souders, ehemals Performance Chef von Yahoo! und Google, zeigt jedoch auf seiner Webseite warum diese „Query Strings“ zum Erweitern der URL nicht verwendet werden sollten. Der weit verbreitete Proxy Server „Squid“, der unter anderem auch von Wikipedia eingesetzt wird, kann die so aufgebauten URLs nicht cachen. Somit würde unter Verwendung der Versionierung der Datei per „Query String“ jeder Anwender, der über einen dieser Proxy Server zugreift, die Seite neu vom eigentlichen Server abrufen. Das Einsparpotenzial würde nicht genutzt werden.[66]

Steve Souders beschreibt im Yahoo! Developer Network eine Alternative. Bei Yahoo! wird im Build-Prozess automatisch eine Versionsnummer in jeden Dateinamen und in den Code für die URL eingefügt. Durch die neue URL kann auf keinem Server oder in keinem Browser die Information zwischengespeichert sein. Sie muss neu vom Server abgerufen werden.[67]

3.6 Positionierung der Stylesheets und Skripte

Das Ladeverhalten von Webseiten und –applikationen lässt sich auch mit Hilfe der Positionierung des Einbindens von Stylesheet- und JavaScript-Dateien optimieren.

Die im HTML definierte Seite wird vom Browser erst gerendert, wenn er alle eingebundenen Stylesheets geladen hat. Denn erst dann steht ihm die Information bezüglich der exakten Darstellung der Seite zur Verfügung. Somit ist es wichtig, dass dem Browser frühestmöglich alle benötigten CSS-Dateien mitgeteilt werden. Dies bedeutet, dass das Einbinden der Stylesheets in der

---

[65] Vgl. Zillgens, Responsive Webdesign, Seite 345

[66] Vgl. Souders, Revving filenames: don’t use querystring, 26.01.2014

[67] Vgl. Souders, High Performance Web Sites: Rule 3, 26.01.2014

HTML-Datei so früh wie möglich, zum Beispiel im „head“-Element des Dokuments, stattfinden sollte.[68]

Stößt der Browser beim Aufbau einer Seite jedoch auf einen Befehl zum Aufruf eines JavaScripts, so wird der weitere Seitenaufbau gestoppt. Und zwar so lange, bis das JavaScript, das normalerweise in einer separaten Datei gespeichert ist, geladen und ausgewertet wurde. Diese Unterbrechung ist notwendig, da mit JavaScript HTML geändert werden kann. Zum Beispiel können zusätzlich Felder hinzugefügt werden. Deshalb kann erst nach Ausführung des enthaltenen Skripts die Seite weiter aufgebaut werden. JavaScript, das nicht für den initialen Seitenaufbau benötigt wird, sollte deswegen ans Ende der HTML-Datei platziert werden. So wird dem Anwender durch das progressive Rendern des Browsers schneller eine Seite angezeigt, auch wenn noch nicht alle JavaScripts geladen wurden.[69]

3.7 Bilder optimieren

Ein weiterer Ansatzpunkt, den Datenumfang von Webseiten zu minimieren, ist das Volumen der Bilder zu optimieren. Eine solche Optimierung wirkt sich am meisten auf Nutzer mit langsamen Datenverbindungen aus. Die grundlegendste Überlegung diesbezüglich führt wieder auf das Prinzip „Content First“ zurück. Welche Bilder werden wirklich benötigt, um den Inhalt der Seite zu übermitteln? Eine Überladung einer Webseite mit Hintergrund- und anderen Bildern macht eine Seite unruhig und unübersichtlich. Zusätzlich würden unnötige Bilder die Ladezeit negativ beeinflussen.

Per Mediaqueries können für verschiedene Anzeigeformate unterschiedlich große Dateien geladen werden, was das Datenvolumen reduziert. Doch auch an den Dateien selbst können Optimierungen vorgenommen werden. So sollte das richtige Dateiformat gewählt werden. Es eignen sich GIFs und PNGs für Symbole mit wenig Farben und JPGs für Fotos. Für letztere sollte immer eine Auflösung gefunden werden, bei der die Anzeige akzeptabel ist, aber auch ein höchstmögliches Komprimierungsergebnis erzielt wird. Bei größeren JPGs ist es sinnvoll, die Datei progressiv zu speichern. So wird das Bild, auch wenn es

---

[68] Vgl. Zillgens, Responsive Webdesign, Seite 331

[69] Vgl. Souders, High Performance Web Sites: Rule 6, 26.01.2014

zunächst noch nicht komplett geladen ist, schon einmal angezeigt, wenn auch verschwommen. Das Bild baut sich während des weiteren Ladeverlaufs weiter auf und wird immer deutlicher.

Auch werden Grafiken oft verwendet, um das Layout der Webseite zu gestalten. Buttons mit Farbverläufen oder Eingabefelder mit runden Ecken sind typische Elemente, die mit Hilfe von Grafiken erstellt werden. Solche Layouts lassen sich mittlerweile mit CSS gestalten. Nutzt man CSS3 hierfür, werden wesentlich weniger Bilder zum Aufbau einer Seite benötigt.

3.7.1 CSS Sprites

Da die Anzahl der für eine Webseite benötigten Dateien einen sehr großen Einfluss auf die Ladezeit hat, ist es sinnvoll, die Anzahl der Bilddateien zu reduzieren. Werden auf einer Seite viele Icons und Symbole verwendet, so kann man diese in einem Bild zusammenfassen, indem man sie neben- und untereinander anordnet. Solche Bilder, in denen viele Grafiken zusammengefasst werden, nennt man „CSS Sprites“. Hauptsächlich sollten die Bilder nebeneinander angeordnet werden, weil hierdurch die Dateigröße geringer ist als bei vertikaler Anordnung.[70] Verwendet werden die einzelnen Icons, indem nur ein Bildausschnitt der Gesamtdatei im Anzeigebereich positioniert wird.

Diese CSS Sprites manuell zu erstellen und die jeweiligen Bildausschnitte zu ermitteln, wäre sehr aufwendig und rechenintensiv. Daher gibt es Tools, die diese Arbeit übernehmen. Sie fügen die Bilder aneinander und geben CSS-Code, der die einzelnen Bildpositionen enthält, zurück.

Hier ein einfaches Beispiel, das die Sprites-Technik verdeutlicht.[71]

Folgendes Sprite namens „img_navsprites.gif“ (Abb. 3-4) enthält drei Teilbilder: Ein Bild für den Home-Button und jeweils eine Grafik für einen Vor- und Zurück-Button.

[70] *Vgl. Yahoo!, Best Practices for Speeding Up Your Web Site, 26.01.2014*

[71] *Vgl. W3Schools, CSS Image Sprites, 26.01.2014*

Abb. 3-4 Überschaubares Sprite

Möchte man das erste Bild verwenden, so kann man es folgendermaßen im HTML einbinden:

```
<img class="home" src="img_trans.gif">
```

Das Bild erhält die Klasse „home“ und ihm wird zunächst eine transparente GIF-Datei als Bildquelle zugewiesen, da der „img“-Befehl eine solche verlangt. Die Anzeige des benötigten Bildbereiches von 46 mal 44 Pixel wird im CSS folgendermaßen definiert, wodurch das transparente Bild überschrieben wird:

```
img.home
{
  width:46px;
  height:44px;
  background:url(img_navsprites.gif) 0 0;
}
```

Das Bild wird hiermit als Hintergrundbild in einen 46 mal 44 Pixel großen Bereich eingefügt. Da das gesamte Sprite an der Position 0, 0 beginnen soll, ist somit exakt das Haus auf dem Home-Button sichtbar.

Die Positionierung der beiden Pfeile, die jeweils 43 Pixel breit sind, geschieht wie folgt:

```
img.prev
{
  width:43px;
  height:44px;
  background:url(img_navsprites.gif) -47px 0;
}

img.next
{
  width:43px;
  height:44px;
  background:url(img_navsprites.gif) -91px 0;
}
```

Die Höhe aller Teilbilder ist identisch, somit sind die Angaben für die Höhe im CSS identisch. Das Bild für den Vor-Button befindet sich jedoch erst ab Pixel 48, denn die ersten 46 Pixel sind das Home-Bild und ein weiterer Pixel folgt als senkrechte Trennlinie zwischen den Bildern. Um das Pixel 48 an den

linken Rand des sichtbaren Bereichs zu rücken, wird das Sprite unterhalb des Rahmens für die Anzeige 47 Pixel nach links verschoben und damit an die korrekte Position gerückt. Ebenso verhält es sich bei dem Zurück-Button. Es muss lediglich das Sprite um 91 Pixel nach links verschoben werden.

3.7.2 Data URI

Eine andere Möglichkeit, die Anzahl der Dateien für Bilder zu reduzieren, ist, die Bilder inline in den CSS-Befehl und somit der CSS-Datei einzubinden. Hierzu verwendet man „Uniform Resource Identifier“ (URI). In CSS ist dies eine spezielle Form des „url“-Befehls, denn es wird nicht eine Adresse oder eine Datei angesprochen, sondern es werden die Daten, in unserem Falle die Daten eines Bildes, im Befehl mit eingefügt. Anstatt auf ein Bild mit Pfadangabe und Dateinamen zu verweisen, wird dem „url“-Befehl die Information mitgegeben, dass Daten inline enthalten sind. Anschließend folgt die Information, welcher Datentyp eingebunden ist und mit welchem Verfahren er kodiert ist. Als letzter Parameter folgen die Daten. Zur Konvertierung von Bildern in eine Zeichenfolge gibt es Tools. Ein Beispiel hierfür ist „data: URI-Generator“ von David Wilkinson (http://dopiaza.org/tools/datauri/index.php, 26.01.2014). Dirk Weber hat ein Werkzeug entwickelt, das für eine CSS-Datei alle darin angesprochenen externen Medien per Data URI in die Datei einfügt (http://www.spritebaker.com, 26.01.2014). Im Folgenden ein Beispielcode für die Einbindung eines Bildes per Referenz auf eine Datei und anschließend per Data URI:

```
background: url(„img/bild1.jpg“);
background: url(„…“);
```

Da die CSS-Dateien durch die eingebundenen Bilder größer werden, sollte man laut Dirk Weber darauf achten, dass GZIP zur Komprimierung der Daten für die Übertragung aktiviert ist. Auch sollte sichergestellt werden, dass es auf ein und dasselbe Bild nur eine Referenz gibt. Denn würde mehrfach auf dasselbe Bild verwiesen, so würde das Bild an jeder dieser Stellen in das CSS eingefügt. Die Datei würde unnötig groß. Ebenso sollte man Data URIs nicht für CSS Sprites verwenden. Das gesamte Bild mit allen enthaltenen Bildern würde überall im CSS enthalten sein, wo ein im Sprite enthaltenes Bild verwendet wird. Data URIs werden von allen aktuellen Browsern korrekt verarbeitet. Lediglich beim Internet Explorer 8 gibt es eine relevante Einschränkung. Er kann inline Bilder nur bis zu einer Größe von 32kB

auswerten.[72] Theoretisch wäre es möglich, Bilder inline auch in das HTML einzufügen, wodurch ebenfalls die Anzahl der Dateien reduziert würde. Das hätte jedoch den Nachteil, dass beim Aufbau der Seite im Browser das HTML inklusive der enthaltenen Bilder heruntergeladen sein muss, bevor der Browser mit dem Rendern beginnen kann. Wird im HTML lediglich auf eine Bilddatei referenziert oder ist das Bild per CSS eingebunden, so kann der Browser die Seite schon anzeigen, bevor die verhältnismäßig großen Bilddaten geladen wurden.[73]

3.8 Content Distribution Network

Da die „Round Trip Time“ abhängig von der Entfernung des Servers zum aufrufenden System ist, spielt die Distanz zum die Webseite bereitstellenden Server eine Rolle für die Performance. Eine Webseite, die aus vielen Regionen aufgerufen wird, kann man zur Performance-Optimierung in einem Content Distribution Network (CDN) bereitstellen. Ein CDN ist ein räumlich verteiltes Serversystem mit einem Hauptserver und mehreren Replikat-Servern. Bei der Anfrage auf dem Hauptserver wird geprüft, wie die Auslastung der Server und wie die Entfernung vom aufrufenden System zu den einzelnen Servern ist. Die Anfrage wird dann an den geeignetsten Server des CDN weitergeleitet und das aufrufende System von dort bedient.

Für größere Webseiten oder –applikationen, die weltweit abgerufen werden, ist ein Hosting verbunden mit CDN auf jeden Fall sinnvoll. Die Seite wird beim Benutzer schneller geladen.[74]

Aber auch für kleinere Webseiten oder –anwendungen macht die Nutzung eines CDNs Sinn. So sollten allgemeine JavaScript- oder CSS-Bibliotheken, die auf einer Webseite eingebunden sind, immer von einem CDN genutzt werden und nicht auf dem die eigene Webseite bereitstellenden Server platziert werden. Dies hat den Vorteil, dass die Bibliothek immer über dieselbe URL aufgerufen wird. Wurde die Bibliothek schon auf einer vorhergehenden Webseite genutzt und im Browsercache gespeichert, so wird sie durch diese

---

[72] *Vgl. Deveria, Can I Use – Data URIs?, 26.01.2014*

[73] *Vgl. Zillgens, Responsive Webdesign, Seite 341*

[74] *Vgl. Müller, Content Delivery Network (CDN) von Google…,.26.01.2014*

Art der Einbindung erkannt und wiederverwendet. Ähnlich verhält es sich auf Proxies. Greifen mehrere Internetseiten auf die gleichen Bibliotheken zurück, so können diese von einem Proxy als identisch identifiziert und ohne Abrufen am Server zurückgegeben werden. Natürlich sorgt auch beim Einbinden von Bibliotheken aus einem CDN die meist geringere Entfernung zum abrufenden System für eine Performancesteigerung.

Ein Beispiel eines CDN für Bibliotheken ist „Google Hosted Libraries“ (https://developers.google.com/speed/libraries/, 26.01.2014). Dort werden beliebte JavaScript Bibliotheken, wie zum Beispiel jQuery und AngularJS, bereitgestellt.

3.9 Lazy Loading

Unter „Lazy Loading“ versteht man das zeitlich nach hinten verschobene Laden von Daten, die zunächst nicht benötigt werden. Bei Webseiten können dies Daten sein, die sich zunächst nicht im sichtbaren Bereich der Webseite befinden. Des Weiteren eignen sich Daten, die erst infolge von Benutzerinteraktionen benötigt werden, für das „Lazy Loading“. Auch Daten, die im Rahmen des „Progressive Enhancement“ zur Basisversion einer Webseite hinzugefügt wurden, können nachgeladen werden. Dies hat den Vorteil, dass ein geringerer Datenumfang bis zur ersten Anzeige der Webseite heruntergeladen werden muss. Die Seite kann schneller angezeigt werden.

Um „Lazy Loading“ zu realisieren, gibt es verschiedene Plugins und Tools. Das „Lazy Load Plugin for jQuery“ von Mika Tuupola (http://www.appelsiini.net/projects/lazyload, 26.01.2014) veranlasst ein Laden der Bilder erst, wenn sie in den sichtbaren Bereich kommen. JavaScript eignet sich ebenfalls zum Nachladen, beispielsweise wenn als zusätzliche Bedienmöglichkeit „Drag and Drop“ Funktionalitäten angeboten werden sollen. Auch für JavaScript gibt es Plugins, zum Beispiel „Lazy“ von Jan Jarfalk (http://www.unwrongest.com/projects/lazy/, 26.01.2014).

3.10 Weitere Ansatzpunkte

Zusätzlich zu den bisher aufgeführten Punkten gibt es viele weitere Dinge, die eine Optimierung der Performance bewirken. Es würde jedoch den Umfang dieser Arbeit überschreiten, alle möglichen Optimierungen aufzuführen. Die bisher genannten sind die Techniken, die die Performance am meisten

beeinflussen. Weitere Punkte, die die Lade- und Ausführungsgeschwindigkeit positiv beeinflussen, sind:

- Rechenintensive CSS-Befehle, wie Expressions, vermeiden
- Webfonts ggf. nicht auf Mobilgeräten verwenden
- Anzahl der DNS Lookups reduzieren
- Weiterleitungen (englisch: redirects) vermeiden
- Asynchronous JavaScript and XML (AJAX) -Content cachebar machen
- Anzahl der Document Object Model (DOM) -Elemente reduzieren
- „Off Canvas“ Bereiche per AJAX nachladen[75]
- Animierte Bereiche, wie „Off Canvas“-Menüs, per CSS anstatt per jQuery animieren[76]
- CSS-Animationen JavaScript-Animationen vorziehen[77]
- Social Media Buttons nicht per Plugin einbinden, sondern lediglich Links verwenden[78] oder Button per 2-Klick-Verfahren zunächst vom Anwender aktivieren lassen[79]

Informationen hierzu und zu weiteren Möglichkeiten der Performance-Optimierung findet man im Artikel „Best Practices for Speeding Up Your Web Site“ im Yahoo! Developer Network[80] oder im Smashing Magazine unter „How To Make Your Websites Faster On Mobile Devices“[81].

3.11 Test der Performance

Sicherlich kann man sehr viel Aufwand in die Performance-Optimierung einer Webseite oder –anwendung investieren. Werkzeuge, die die Performance testen, zeigen jedoch konkret die Bereiche auf, an denen Optimierungen notwendig sind.

---

[75] *Vgl. Wroblewski, Off Canvas Multi-Device Layouts, 20.01.2014*

[76] *Vgl. Bushell, Implementing Off-Canvas Navigation…, 26.01.2014*

[77] *Vgl. Zillgens, Responsive Webdesign, Seite 252*

[78] *Vgl. Jonathan, ZURBlog – Small, Painful Buttons…, 26.01.2014*

[79] *Vgl. Schmidt, 2 Klicks für mehr Datenschutz, 26.01.2014*

[80] *Vgl. Yahoo!, Best Practices for Speeding Up Your Web Site, 26.01.2014*

[81] *Vgl. Johansson, How To Make Your Websites Faster …, 26.01.2014*

Ein solches Tool ist „YSlow“ von Yahoo! (http://developer.yahoo.com/yslow/, 26.01.2014). Es ist ein Firebug AddOn, das einer aufgerufenen Webseite Noten für verschiedene Kategorien, die zur Performance-Optimierung genutzt werden können, vergibt. Es werden Noten nach dem Prinzip der amerikanischen Schulnoten vergeben. Somit ist eine Bewertung mit „A“ das Optimum. Folgender Screenshot des YSlow-Tests der Seite des „Bundesministerium der Finanzen“ (http://www.bundesfinanzministerium.de, 23.12.2013) zeigt die Bewertung und Verbesserungsvorschläge.

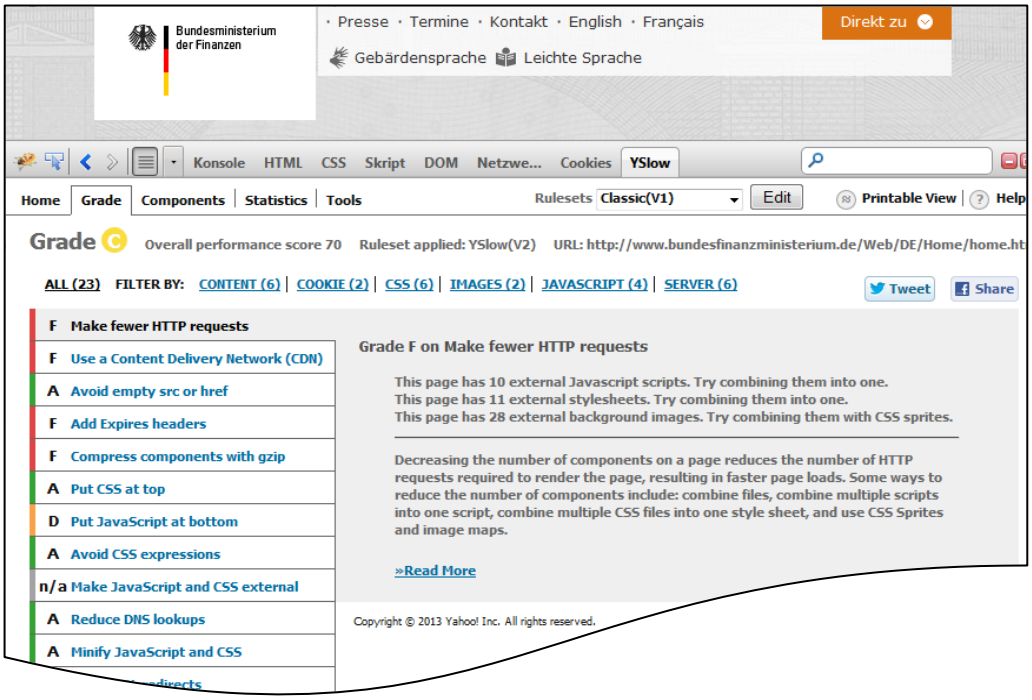

Abb. 3-5 Ergebnis eines YSlow-Tests

Das Werkzeug „Page Speed“ von Google, ebenfalls ein AddOn für Firebug, geht einen Schritt weiter als YSlow und bietet optimierten Code und komprimiertere Bilder zum Download an (http://code.google.com/speed/page-speed/, 26.01.2014).

„WebpageTest“ von Google, ursprünglich entwickelt von AOL, ist eine Webseite, von der ausgehend die Performance von Webseiten getestet werden kann (http://www.webpagetest.org/, 26.01.2014). Einstellen kann man bei diesem Tool, für welchen Browser und von welchem Ort aus der Test durchgeführt werden soll. „WebpageTest“ kann auch lokal installiert werden, wodurch ein Test von Webseiten oder Webapplikationen, die nicht öffentlich zugänglich sind, möglich wird.

4 Webapplikationen

Die bisher aufgeführten Grundlagen des responsiven Webdesigns beziehen sich hauptsächlich auf den Aufbau und die Darstellung von Webseiten, die Informationen zur Performance-Optimierung auf die Ladezeiten und die benötigte Rechenleistung für die Ausführung der Seiten. Für die Erstellung von responsiven Webapplikationen (englisch: Responsive Web Application (RWA)) ist dies alles ebenfalls von Bedeutung. Jedoch spielen bei Applikationen Funktionalitäten und Techniken, mit denen sie zu realisieren sind, eine weitere wichtige Rolle.

Diese Techniken werden im folgenden Kapitel erläutert. Denn immer häufiger besteht die Anforderung, Programme, die bisher lokal installiert wurden, als Webanwendung umzusetzen. Dies bedeutet, dass die Anwendung auf einem Server bereitgestellt und vom Browser abgerufen und ausgeführt wird. Solche Applikationen können zum Beispiel E-Mail-Clients, Textverarbeitung, Tabellenkalkulation, Bildverarbeitung, Auskunftssysteme oder Kundenverwaltungssysteme (englisch: Customer Relationship Management (CRM)) sein. Vorteil solcher Webanwendungen ist die ständige Verfügbarkeit aktueller Daten, sofern diese online abgelegt werden. Wie das aktuelle Schlagwort „Cloud“ zeigt, ist die Speicherung von Daten auf entfernten Systemen momentan ein Trend. Voraussetzung für den Zugriff auf die Daten ist jedoch, dass ein Internetzugang verfügbar ist und sonstige technische Probleme bezüglich der Datenverbindung nicht auftreten. Ein weiterer Vorteil ist, dass eine Webapplikation nur auf einem oder mehreren Servern installiert wird, wodurch Installationen, Updates, Wartung und Support auf den einzelnen Anwendersystemen nicht mehr notwendig sind.

Die zur Erstellung einer Webapplikation benötigten Techniken sind HTML, CSS und JavaScript. Lange Zeit war der Funktionsumfang hierbei sehr beschränkt. Um ein wenig Funktionalität einbinden zu können, wurde Java, Flash oder Silverlight genutzt, die alle drei als zusätzliches Plugin für den Browser installiert werden. Wird eine Anwendung in einem dieser Plugins gestartet, so ist die Kommunikation daraus mit dem HTML-Dokument, in das die Anwendung eingebunden war, nicht möglich. Es bestand keine Integration,

wodurch sich Java zum Beispiel nicht durchsetzte und auch Flash nicht das Potenzial für Webanwendungen hat.[82]

Im Jahr 2005 schon veröffentlichte die „Web Hypertext Application Technology Working Group“ (WHATWG) den Entwurf einer Spezifikation für „Web Applications 1.0“.[83] Dieser Entwurf beschreibt Eigenschaften für HTML und DOM, die die Erstellung von webbasierten Anwendungen vereinfachen sollen. Enthalten waren damals schon inline Popup-Fenster, Kontext-Menüs und eine direkte Grafikleinwand beziehungsweise -fläche (englisch: canvas). Aus diesem Entwurf ging die Weiterentwicklung der Auszeichnungssprache HTML, der Stylesheets CSS und des JavaScripts zu HTML5 hervor. Diesem liegt somit ein neues Konzept zu Grunde.

Unter HTML5 wird nicht nur die Auszeichnungssprache HTML in der Version 5 verstanden, sondern man fasst darunter das Gesamtprodukt aus HTML5, CSS3 und JavaScript zusammen. Dabei kommt dem HTML die Aufgabe des Aufbaus der Struktur einer Webseite oder –anwendung zu, CSS ist für die optische Aufbereitung zuständig und JavaScript bietet die Funktionalität, die es ermöglicht, Webapplikationen zu erstellen. Ein Beispiel für umfassende Webapplikationen sind die Online-Anwendungen von Google, zusammengefasst unter dem Namen „Drive“ (https://drive.google.com – „Erstellen“-Button, 26.01.2014). Dazu gehören unter anderem die Anwendungen zur Erstellung von Textdokumenten, Präsentationen, Tabellen, Formularen und Zeichnungen. Diese Office Anwendungen sind rein in HTML, CSS und JavaScript erstellt.[84]

Welche Funktionalitäten die Erstellung von Webapplikationen unterstützen, wird im Folgenden erläutert.

---

[82] Vgl. Gauchat, HTML5, CSS3 & JavaScript, Seite 17

[83] Vgl. WHATWG, Web Applications 1.0, 26.01.2014

[84] Vgl. Müller, Flexible Boxes, Seite 62

4.1 Formulare und Validierung

Formulare sind Bestandteil vieler Anwendungen. Vor allem, wenn es darum geht, Daten zu erfassen und daraus ein Ergebnis zu ermitteln oder die Daten persistent zu speichern. HTML5 bietet hierzu neue Elemente, Eingabetypen und Attribute an, die von den Browsern spezifisch angezeigt und validiert werden. Zuvor mussten solche client-seitigen Funktionen und Prüfungen mit JavaScript implementiert werden. Für responsive Webanwendungen besteht der Vorteil dieser neuen Elemente und Attribute darin, dass auf Geräten mit kleinen Anzeigeflächen die Eingabemethode angepasst werden kann. Beispielsweise kann für ein Formularfeld, das numerische Werte erwartet, direkt die Ziffern-Tastatur auf einem Handy eingeblendet werden. Dies macht eine Anwendung auf Geräten mit Bildschirmtastatur (englisch: On-Screen-Keyboard) wesentlich bedienungsfreundlicher.

Leider werden viele dieser Eigenschaften noch nicht von allen Browsern umgesetzt. Auf der Internet-Seite der W3Schools (http://www.w3schools.com, 26.01.2014) kann man nachsehen, in welchem Browser welche Funktionalitäten schon umgesetzt sind.

Dort wird auf der Unterseite „HTML5 Form Input Types“ (http://www.w3schools.com/html/html5_form_input_types.asp, 26.01.2014) für alle Eingabetypen die Browserunterstützung aufgeführt.

Im Folgenden werden einige dieser neuen Elemente, wie Datumswähler oder Schieberegler, erläutert. Basis-Element dieser neuen Möglichkeiten ist das auch in vorhergehenden HTML-Versionen vorhandene „form“-Element. Ausprobieren kann man die Elemente unter dem oben angeführten Link der W3Schools. Es wird ein Beispiel-Code angezeigt und ausgeführt. Den Beispielcode kann man anpassen und umsetzen lassen. Die folgenden Screenshots stammen, falls nicht anders angeführt, von den Seiten der W3Schools.

4.1.1 Attribute des „form“-Elements

Zwei neu im Standard aufgenommene Attribute gibt es für das „form“-Element, das die einzelnen Formularelemente umfasst.

„novalidate“

HTML5 bietet die Möglichkeit, Prüfungen für die einzelnen Formularfelder durchzuführen. Diese Prüfungen sind abhängig von den Typen oder den Attributen der Eingabefelder. Im folgenden Abschnitt werden diese beschrieben. Die Validierung des Formulars wird im Standardfall durchgeführt. Um eine automatische Validierung zu unterbinden, kann im „form“-Element das Attribut „novalidate“ eingefügt werden. Es unterbindet für das gesamte Formular die Prüfung. Wie man in der folgenden beispielhaften Code-Zeile sieht, reicht das Attribut zum Deaktivieren der Validierung aus, es muss kein Wert zugewiesen werden.

```
<form name=“form“ methode=“get“ novalidate>
   …
</form>
```

„autocomplete“

HTML5 ermöglicht eine Autovervollständigung. Sie ist ohne Angaben zur Vervollständigung aktiviert. Setzt man das Attribut „autocomplete“ auf den Wert „off“, so ist die automatische Vervollständigung deaktiviert. Ein explizites setzen auf „on“ ist ebenfalls möglich.

4.1.2 Eingabetypen des „input“-Elements

Das wichtigste Eingabefeld war vor HTML5 das „input“-Element mit dem Typ „text“. Neu eingeführt wurden Typen, wie „number“, „date“ und „email“. Diese Angabe der Typen ermöglicht nun mit HTML5 eine Validierung im Browser. Browser, für die die Typen noch nicht implementiert sind, stufen sie auf den Wert „text“ herab, wodurch das Element wie ein normales Textfeld funktioniert und die Prüfungen entfallen.

„number“

Ein so gekennzeichnetes „input“-Element erlaubt nur die Eingabe von numerischen Werten. Mit den Attributen „min“ und „max“ kann ein erlaubter Wertebereich definiert werden. Der Wert des Attributs „value“ wird als Vorbelegung in das Feld eingetragen. Dieser wird gelöscht, sobald der Anwender den Cursor in das Feld positioniert. Mit dem Attribut „step“ kann eine Schrittweite festgelegt werden, wie das folgende Codebeispiel zeigt.

<input type="number" name="quantity" min="1" max="10" step="2" value="2">

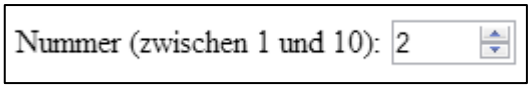

Abb. 4-1 Eingabefeld vom Typ „number“

Obenstehende Abbildung (Abb. 4-1) zeigt, wie manche Browser, in diesem Falle Chrome, Pfeile in das Formularfeld einfügen, um den eingetragenen Wert um die Schrittweite zu erhöhen oder zu verringern. Browser für Geräte mit eingeblendeter Tastatur zeigen für Felder des Typs „number“ meist direkt die Zifferntastatur an, wodurch ein Umschalten auf die Ziffern für den Anwender entfällt.[85]

„date“

Der Input-Typ „date“ fordert die Eingabe eines Datums. Browser, die diesen Datentyp umgesetzt haben, bieten eine Kalenderoberfläche zur Auswahl eines Datums an, das dann in das „input“-Feld übernommen wird (Abb. 4-2). Ein solcher „Datepicker“ musste vor HTML5 per JavaScript umgesetzt werden. Im Browser Safari wird auf dem iPhone die Datumsauswahl per beschrifteter, drehbarer Rollen angezeigt.

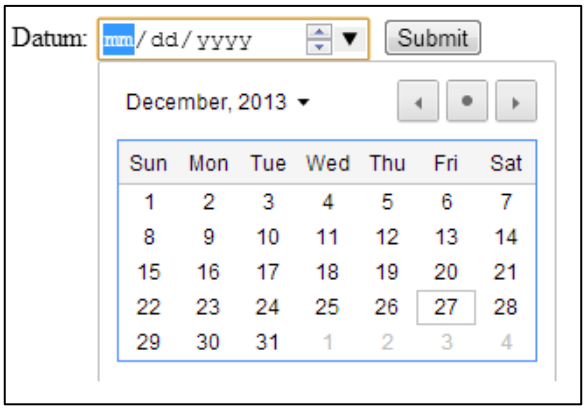

Abb. 4-2 Eingabefeld vom Typ „date“„email“

Ein HTML5 fähiger Browser prüft bei einem Eingabefeld des Typs „email“, ob der eingegebene Wert der Syntax einer E-Mail-Adresse entspricht. Browser auf Desktop-Geräten zeigen ein normales Textfeld zur Eingabe an. Auf

[85] *Vgl. Zillgens, Responsive Webdesign, Seite 135*

Smartphones oder Tablets wird zum Teil die Bildschirmtastatur angepasst, sodass zum Beispiel das @-Zeichen direkt auf der Tastatur verfügbar ist.

„tel“

Der Datentyp „tel“ geht nicht mit einer Prüfung der eingegebenen Daten einher, wie es bei „email“ der Fall ist. Der Typ „tel“ dient zur semantischen Auszeichnung eines Eingabefeldes. So ist ein auf HTML5 angepasster Browser in der Lage, für dieses Eingabefeld die Nummer-Tastatur zu öffnen (Abb. 4-3).

Abb. 4-3 Eingabefeld vom Typ „tel“ auf einem Smartphone

„url“

Ein „input“-Element vom Typ „url“ prüft, ob das Format einer eingegebenen Zeichenfolge dem einer URL entspricht.

„search“

Mit dem Datentyp „search“ kann ein Eingabefeld, das für Suchen genutzt werden kann, ausgezeichnet werden. Dies ermöglicht zum Beispiel einem Screenreader, sobald ein Befehl zum Suchen abgesetzt wird, sofort in das Suchfeld zu springen. Browser stellen das Suchfeld zum Teil anders dar als andere Eingabefelder. Auf dem Betriebssystem Mac OS von Apple werden die Suchfelder in Opera, Safari und Chrome mit runden Ecken dargestellt, so wie es beim Betriebssystem selbst auch der Fall ist. Einen Löschen-Button zum Leeren des Feldes fügen Safari und Chrome ein.[86] Auf anderen

[86] *Vgl. Zillgens, Responsive Webdesign, Seite 139*

Betriebssystemen sieht das Feld vom Typ „search“ wie ein normales Textfeld aus.

„range“

Der Datentyp „range“ ist für die Eingabe eines Wertes aus einem Wertebereich gedacht, wenn der Wert nicht exakt benötigt wird. Meist wird ein Element dieses Typs als Schieberegler oder als Pfeile, mit denen man den Wert hinauf oder hinunter regeln kann, dargestellt. Einen Standard gibt es zur Anzeige noch nicht.[87] Chrome auf einem Windows-Rechner zeigt einen Schieberegler an (Abb. 4-4).

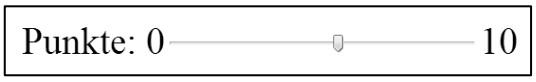

Abb. 4-4 Eingabefeld vom Typ „range“

Mit den Attributen „min“ und „max“ wird der Wertebereich des Feldes festgelegt. Die Werte werden nicht automatisch am Anfang und am Ende der Skala angezeigt. Möchte man die Werte anzeigen lassen, so muss man sie, wie der folgende Code zu Abb. 4-5 zeigt, zusätzlich einfügen. Der Wert im Attribut „value“ legt die Ausgangsposition des Schiebereglers fest. Im Attribut „step“ kann eine Schrittweite angegeben werden.

```
Punkte: 0<input type="range" name="punkte" min="1" max="10">10
```

Beherrscht ein Browser den Typ „range“ noch nicht, so wird in ihm lediglich ein Texteingabefeld angezeigt. Der Internet Explorer 8 gestaltet das Textfeld aus oben stehendem Code folgendermaßen (Abb. 4-5).

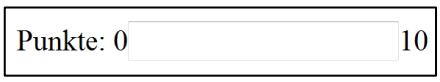

Abb. 4-5 Eingabefeld „range“ im Internet Explorer 8

4.1.3 Attribute des „input“-Elements

Verschiedene Attribute zu den „input“-Elementen wurden neu in HTML5 aufgenommen. Zuvor wurden die Funktionalitäten, die sie erfüllen, oftmals per

[87] *Vgl. Gauchat, HTML5, CSS3 & JavaScript, Seite 177*

JavaScript realisiert. Die vorangehend bei den neuen Elementen aufgeführten Attribute „min“, „max“ und „step“ sind ebenfalls neu in HTML5, werden hier jedoch nicht mehr erwähnt, da sie schon erläutert wurden. Im Folgenden werden beispielhaft einige zusätzliche neue Attribute genannt.

„formnovalidate“

Das Attribut „formnovalidate“, ohne Wertzuweisung, wird auf ein Eingabeelement angewendet, um die Validierung auf diesem Feld auszuschalten. Wird das Attribut nicht verwendet, so ist die Prüfung aktiv. Es entspricht dem Attribut „novalidate“, das auf „form“-Ebene verwendet wird und die Prüfung für alle Eingabefelder des Formulars beeinflusst.

„autocomplete“

Das Attribut „autocomplete“, dem der Wert „on“ oder „off“ zugewiesen werden kann, wird auf ein Eingabeelement angewendet, um die automatische Vervollständigung auf diesem Feld ein oder aus zu schalten. Wird das Attribut nicht explizit verwendet, so ist die Vervollständigung aktiviert. Auch dieses Attribut ist analog dem „autocomplete“-Befehl, der auf „form“-Ebene verwendet wird. Dort schaltet es die Vervollständigung für alle Eingabefelder des Formulars ein oder aus.

„placeholder“

Auf die Eingabefelder der Typen „text“, „search“, „url“, „tel“, „email“ und „password“ kann ein Attribut „placeholder“, dessen Wert ein Wort oder einen kurzen Text enthält, zugewiesen werden.[88] Der Text dient zur Erläuterung dessen, was als Eingabe erwartet wird. Ein Literal zur Beschriftung des Feldes sollte zusätzlich vorhanden sein. Erhält das mit einem Platzhalter versehene Eingabefeld den Fokus, so wird der zuvor meist gegraute Schriftzug gelöscht. Folgender Code ergibt das Eingabefeld mit Platzhaltertext (Abb. 4-6):

```
Name: <input type="text" name="name" placeholder="Nachname">
```

[88] *Vgl. W3Schools, HTML5 Form Attributes, 26.01.2014*

Abb. 4-6 Attribut „placeholder“

„required“

Das Attribut „required“, dem kein Wert zugewiesen wird, markiert Pflichtfelder. Das Absenden eines Formulars ist nicht möglich, wenn ein so gekennzeichnetes Feld nicht ausgefüllt wurde.

„autofocus“

Zur Platzierung des Fokus in ein Eingabefeld eines Formulars beim Aufruf der Seite kann das „autofocus“-Attribut verwendet werden. Auch dieses Attribut ist ein boolscher Wert und benötigt somit keine Wertzuweisung. Es sollte natürlich auf einer Seite der Anwendung nur einmal verwendet werden.

„multiple“

Auf die Elemente des Typs „email“ und „file“ kann das Attribut „multiple“ gesetzt werden.[89] Es bewirkt, dass mehrere E-Mail-Adressen eingegeben oder mehrere Dateien ausgewählt werden können.

„pattern“

Mit Hilfe des Attributs „pattern“ kann ein Regulärer Ausdruck definiert werden, gegen den die Eingabe, die in das Feld erfolgt, geprüft wird. Gültig ist dieses Attribut für Elemente der Typen „text“, „search“, „url“, „tel“, „email“ und „password“.[90] Zum Beispiel kann per „pattern“-Attribut geprüft werden, ob eine abgefragte Postleitzahl eine fünfstellige Nummernfolge enthält.

<input type=“text“ pattern=“[0-9]{5}“>

Eine Postleitzahl darf auf diese Weise nur geprüft werden, wenn es sich um eine Adresse in Deutschland handelt. Sollen weltweite Adressen eingegeben werden können, so darf der Anwender nicht derartig eingeschränkt werden.

---

[89] Vgl. W3Schools, HTML5 Form Attributes, 26.01.2014

[90] Vgl. W3Schools, HTML5 Form Attributes, 26.01.2014

Validierung der Eingabe

Werden für Eingabeelemente Validierungen per Attribut „pattern“ definiert, so erfolgt zurzeit die Validierung nicht einheitlich. Die Browser Opera und Chrome prüfen nicht während der Eingabe der Daten oder beim Verlassen des Feldes, sondern erst, wenn das Formular abgeschickt wird. Firefox hingegen validiert die Daten beim Verlassen des Feldes. Der Internet Explorer 8 prüft bei Angabe eines Patterns gar nicht.

Fehlerhafte Eingaben werden in den verschiedenen Browsern auch unterschiedlich gekennzeichnet. Chrome zum Beispiel markiert die erste fehlerhafte Eingabe mit einem orange Rahmen um das Feld und gibt einen Text aus, der besagt, dass das Format eingehalten werden muss. Wurde dem Element das Attribut „title“ mitgegeben, so wird der dem Attribut „title“ zugewiesene Text in der Fehlermeldung mit angezeigt. Der Text wird auch angezeigt, wenn die Maus sich über dem Feld befindet. Somit ist es sinnvoll, dort das erwartete Format zu beschreiben. Firefox gibt eine sinngemäß identische Meldung zu dem ersten fehlerhaften Feld beim Absenden aus, markiert jedoch alle Fehler, indem er das Eingabefeld beim Verlassen rot umrahmt. Der folgende Code ergibt in den beiden Browsern Chrome und Firefox beim Absenden ein unterschiedliches Ergebnis. Abb. 4-7 zeigt die unterschiedliche Darstellung des folgenden Codes. Der Cursor befindet sich dabei über dem Feld für die Postleitzahl, wodurch der Text des Attributs „title“ angezeigt wird.

```
Straße: <input type="text">
Hausnr.: <input type="text" pattern="[0-9]{4}"
                title="numerisch 1 bis 1000"><br>
Postleitzahl: <input type="text" name="plz" pattern="[0-9]{5}"
                title="Fünfstellige Ziffernfolge">
Ort: <input type="text">
```

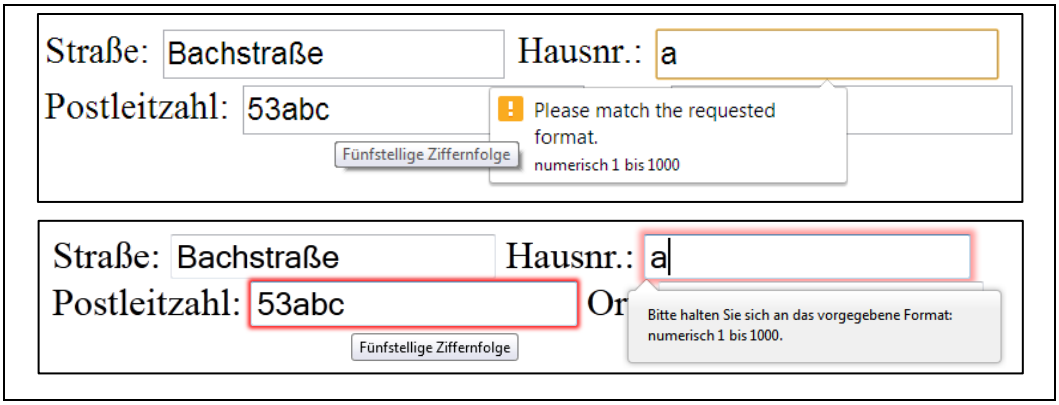

Abb. 4-7 Validierungsergebnis im Chrome (oben) und Firefox (unten)

4.1.4 Elemente

Neue Elemente sind ebenfalls mit HTML5 eingeführt worden, allerdings bietet die Umsetzung dieser in den einzelnen Browsern noch Verbesserungspotenzial.

„datalist“

In dem neuen HTML-Element „datalist“ können Vorschläge für die Werte eines Feldes definiert werden. Die Werte sollen zukünftig als Dropdown-Liste angezeigt werden. Firefox und Chrome zeigen zum aktuellen Zeitpunkt jedoch die vorgegebenen Werte nur an, sobald man zu schreiben beginnt. Tippt man ein erstes Zeichen ein, so zeigt Chrome in der Version 31 alle Werte an, die damit beginnen. Firefox in der Version 26 bietet alle Werte an, die das eingegebene Zeichen oder die Zeichenfolge enthalten. Abb. 4-8 zeigt das unterschiedliche Verhalten der Browser für den folgenden Beispielcode.

```
<input list="browsers" name="browser">
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

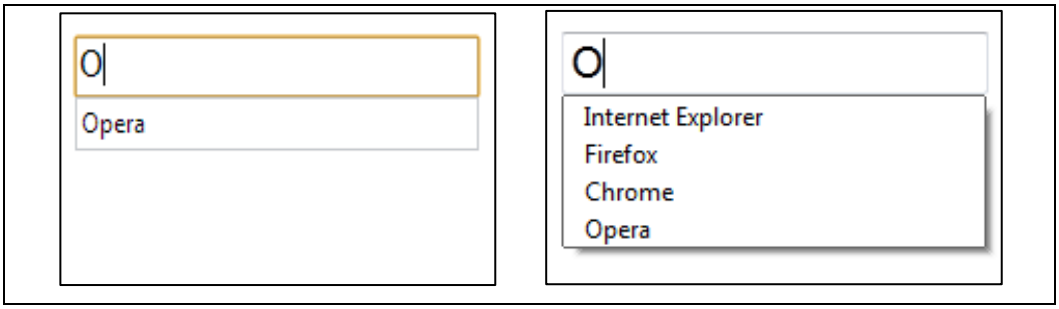

Abb. 4-8 „datalist“ im Chrome (links) und Firefox (rechts)

„output“

Das Element „output“ dient zur semantischen Auszeichnung eines Rechenergebnisses. In dem zugehörigen Attribut „for“ kann man die IDs der Eingabefelder, aus denen das Ergebnis berechnet wird, vermerken. Im Element „output“ kann man das Ergebnis, meist mit Hilfe eines JavaScripts ermittelt, anzeigen, wodurch es semantisch ausgezeichnet wird.

4.2 JavaScript APIs

JavaScript APIs sind Schnittstellen für die Anwendungsprogrammierung (englisch: Application Programming Interface (API)). Über ein API greift man auf Funktionalitäten einer Bibliothek zu, um diese zu nutzen. In der Spezifikation von HTML5 wurden einige Bibliotheken direkt integriert, sodass die Browser sie implementieren und die API für Entwickler von Webseiten und –anwendungen zur Verfügung stellen. Der Stand der Umsetzung in den verschiedenen Browsern ist sehr unterschiedlich. Bei Verwendung sollte zuvor geprüft werden, wie weit die Umsetzung in den bevorzugten oder vorgegebenen Browsern ist. Eine Webseite, auf der die Browserunterstützung nachgesehen werden kann, ist „Can I Use“ (www.caniuse.com, 26.01.2014) Im Folgenden werden verschiedene HTML5 APIs erläutert.

4.2.1 Forms-API

Einfache Validierungen lassen sich in HTML5 mit den unterschiedlichen Eingabetypen, wie „email“ oder „date“ realisieren. Attribute, wie „min“ und „max“, geben Grenzwerte an und als Pflichtfeld kann ein Eingabefeld mit dem Attribut „required“ gekennzeichnet werden. Möchte man komplexere Prüfungen durchführen, die zum Beispiel von anderen Feldern abhängig sind,

so kann dies mit der in HTML5 enthaltenen JavaScript Forms-API realisiert werden.

Meldungen zu einem Element ausgeben per „setCustomValidity()“

So lassen sich eigene Fehlermeldungen, die beim Absenden eines Formulars im Fehlerfall angezeigt werden, per „setCustomValidity(message)“ für ein Formularelement setzen. Dies kann zum Beispiel für Prüfungen, ob mindestens ein Feld von zweien gefüllt ist, verwendet werden. Per JavaScript ruft man bei jeder Eingabe in eines der beiden Felder eine JavaScript Funktion auf, in der geprüft wird, ob nicht beide Felder leer sind. Diese Prüfung muss initial auch beim Aufruf der Seite erfolgen. Sind beide Felder leer, so setzt man mit „setCustomValidity(message)“ eine entsprechende Meldung, dass mindestens eine Eingabe benötigt wird. Ist mindestens ein Feld ausgefüllt, so setzt man die Meldung auf einen Leerstring zurück. Beim Absenden des Formulars wird, falls ein Literal gesetzt ist, eine Meldung ausgegeben.

Event „invalid“ beim Absenden eines Formulars

Tritt beim Absenden eines Formulars ein Fehler auf, so wird automatisch das Event „invalid“ für das Formular erzeugt. Per Event-Listener, der auf das Formular-Element platziert wird, kann das Event, das die Information über das verursachende fehlerhafte Element enthält, abgefangen werden. Mit Hilfe des Events „invalid“ kann somit zum Beispiel die Hintergrundfarbe des Fehler verursachenden Feldes verändert werden.

Per Objekt „ValidityState“ bei jeder Eingabe prüfen

Möchte man das Feld bei jeder vom Anwender getätigten Eingabe prüfen, so kann man dies mit Hilfe des Objektes „ValidityState“ erreichen. Man setzt auf das Formular einen Event-Listener des Typs „input“. Dieses Event wird automatisch geworfen, wenn ein Eingabefeld des Formulars editiert wird und löst dann den Aufruf einer Prüffunktion aus. Darin wird zunächst geprüft, ob das Feld valide ist oder nicht (if (element.validity.valid) {…}). Das Attribut „valid“ gibt dabei „true“ oder „false“ zurück. Möchte man zum Beispiel den Hintergrund des fehlerverursachenden Feldes direkt während der Eingabe auf Rot setzen, so entnimmt man dem Event das auslösende Feld und ändert die Farbe.

Zusätzlich kann das „ValidityState“-Objekt auch gefragt werden, wie der Status exakt ist (element.validity.status). Der Status gibt zurück, dass zum Beispiel ein Typ wie „numeric“ nicht eingehalten wurde („typeMismatch“). Folgende Beispiele selbsterklärender Fehler können ebenfalls zurückgegeben werden: „tooLong“, „patternMissmatch“, „valueMissing“, „rangeUnderflow“, …

4.2.2 Canvas-API

Die Canvas-API ist der Versuch in HTML5, unabhängig von bis dahin genutzten Plugins wie Flash oder Java-Applets zu werden. Mit der Canvas-API können Grafiken gezeichnet, wiedergegeben, Bilder und Texte animiert und verarbeitet werden.[91] Das Ergebnis ist ein Bitmap, das heißt eine pixelbasierte Grafik.

Mit dem HTML-Element „canvas“ legt man eine Grafikfläche an, deren Größe mit „width“ und „height“ bestimmt wird. Auf dieses Element muss man die Methode „getContext( )“ ausführen, um den Zeichenkontext zu erhalten. Als Parameter erwartet die Methode den Wert „2d“ oder „webgl“ (Web Graphics Library) mit deren Hilfe auch dreidimensionale Grafiken gezeichnet werden können.[92] Der Zeichenkontext besteht aus Pixeln, die mit verschiedenen Methoden angesprochen werden können. So kann zum Beispiel mit dem Befehl „strokeRect(x, y, width, height)“ ein Rahmen in Form eines Rechtecks gezeichnet werden, das an Position x/y beginnt und die angegebene Breite und Höhe annimmt. Per „strokeStyle“ kann die Farbe des Rahmens gesetzt werden. Komplexere Formen lassen sich mit Hilfe von Zeichenpfaden vorbereiten und anschließend auf die Fläche zeichnen. Zum Beispiel können mit der Methode „arc( )“ Kreise oder Teilbögen entwickelt werden. Ebenfalls kann man Texte in seine Grafik platzieren und Grafikelemente mit Schatten versehen. Animationen sind möglich, Bilder und Videos können eingefügt werden.

Für Webanwendungen bietet die Canvas-API die Möglichkeit, Grafiken, wie zum Beispiel Diagramme, dynamisch zur Laufzeit zu erstellen, sodass sie abhängig von eingegebenen oder ermittelten Werten sein können. Möchte

[91] Vgl. Gauchat, HTML5, CSS3 & JavaScript, Seite191

[92] Vgl. WHATWG, HTML – Living standard - The canvas element, 26.01.2014

man in seiner Anwendung die Bearbeitung von Grafiken und Videos anbieten, so ist dies nun ebenfalls möglich.

4.2.3 Drag&Drop-API

In Desktop-Anwendungen ist es möglich, Elemente anzuklicken und zu verschieben. Dieses „Drag and Drop"-Verfahren kann genutzt werden, um Elemente umzusortieren, Dateien einzufügen oder sie im Programm zu öffnen und so weiter. Die Drag&Drop-API macht dies nun für Grafiken, Links, Dateien und Daten auch bei Webanwendungen möglich. Hierzu wurden verschiedene Events neu eingeführt. Das Element, das verschoben werden soll, löst beim Aufnehmen das Event „dragstart", während des Bewegens „drag" und beim Loslassen „dragend" aus. Beim Aufnehmen wird in dem Ausgangselement der zu verschiebende Inhalt in das Objekt „dataTransfer" gespeichert. Das Zielelement, das zuvor für das Aufnehmen von Elementen aktiviert werden muss, löst beim Loslassen das Event „drop" aus. Das im Zielobjekt befindliche Objekt „innerHTML" wird mit dem Inhalt von „dataTransfer" befüllt und der Inhalt wird angezeigt. Ebenso ist es möglich, Dateien per „Drag and Drop" in eine im Browser laufende Anwendung hineinzuziehen oder eine im Browser angezeigte Datei in das Dateisystem des Rechners zu ziehen.

4.2.4 Web-Storage-API

Für die Funktionsfähigkeit von Anwendungen ist es notwendig, dass Daten temporär zwischengespeichert oder langfristig festgeschrieben werden können. Im Web standen hierfür vor HTML5 lediglich die Cookie-Technik und die API „Web SQL Database" zu Verfügung. „Web SQL Database" war für das W3C nicht zukunftsträchtig und wurde aus der Spezifikation entfernt. Das Cookie-Verfahren ermöglicht lediglich eine Speicherung von kurzen Strings. Zudem stellen Anwender im Browser oft ein, dass Cookies nur eine Sitzung (englisch: session) lang erhalten bleiben sollen. Somit ist eine Speicherung, wie sie für Anwendungen benötigt wird, damit meist nicht zu realisieren. Um das Ziel des W3C, das Web bereit für Anwendungen zu machen, umzusetzen, wurde die Web-Storage-API entwickelt. Web-Storage ist unterteilt in zwei Bereiche, der „sessionStorage" und der „localStorage". Die beiden Verfahren speichern die Daten als Schlüssel/Wert-Paare (englisch: key/value pairs) und werden mit denselben Methoden und Eigenschaften umgesetzt. Schlüssel und Wert werden als Zeichenkette (englisch: string) abgelegt. Zugriff auf die Daten

hat nur die Webseite oder -anwendung, die sie auch gespeichert hat. Das W3C empfiehlt den Browserherstellern, fünf Megabyte (MB) für jede Webseite zur Verfügung zu stellen und bei Erreichen des Limits den Benutzer um die Erlaubnis zu bitten, den Speicherplatz zu erweitern. Im Folgenden werden die Besonderheiten für „sessionStorage“ und „localStorage“ erläutert.

sessionStorage

Mit Hilfe von „sessionStorage“ können Daten nur zur Laufzeit einer Sitzung gespeichert werden. Dies ähnelt dem Verfahren bei Cookies, jedoch sind Cookies immer für den gesamten Browser gültig. Per „sessionStorage“ abgelegte Informationen sind nur in einem Fenster oder Reiter verfügbar.

localStorage

Per „localStorage“ werden Daten persistent auf dem Rechner gespeichert. Die Daten stehen allen Reitern oder Browser-Fenstern zur Verfügung, vorausgesetzt, sie befinden sich auf derselben Webseite. Werden Daten über ein Fenster geändert, so wird das Event „storage“ ausgelöst. Horcht man auf dieses Event und löst bei Auftreten einen Neuaufbau des Bereiches auf, in dem die Daten angezeigt werden, so können die verschiedenen Fenster synchron gehalten werden.[93]

Für beide Speicherarten erfolgt das Ablegen der Daten als Schlüssel/Wert-Paar mit der Methode „setItem(key,value)“, ausgeführt auf das Objekt „sessionStorage“ beziehungsweise „localStorage“. Die Methode „getItem(key)“ liefert den Wert zu einem Schlüssel zurück. Verändert werden kann der Wert per „sessionStorage/localStorag[key]=newValue“. Ebenso einfach kann die Anzahl der gespeicherten Paare abgefragt oder könnenDaten gelöscht werden.

4.2.5 Indexed-Database-API

Indexed-Database (IndexedDB) ist ein im Browser integriertes Datenbanksystem, mit dem größere Datenmengen strukturiert persistent gespeichert werden können. Dieses System soll laut W3C die Web-SQL-

[93] *Vgl. Gauchat, HTML5, CSS3 & JavaScript, Seite 272*

Database-Funktionalität ersetzen, die bei HTML5 nicht mehr in der Spezifikation vorhanden ist. Die „Web Applications Working Group“ des W3C entwickelt zur Speicherung die Indexed-Database-API weiter. Zur Umsetzung im Browser empfiehlt das W3C das B-Baum-Konzept. Die Indexed-Database-API befindet sich zurzeit noch im Status „Candidate Recommendation“.[94] Dies bedeutet, dass die Entwicklung aus Sicht des W3C abgeschlossen ist, jedoch noch Rückmeldungen von nutzenden Entwicklern einfließen können. Die Unterstützung ist in vielen aktuellen Browserversionen gegeben. Lediglich Safari, iOS Safari und Opera mini unterstützen IndexedDB nicht. Der Android Browser wird ab der Version 4.4 IndexedDB implementiert haben, wie auf der Internetseite „Can I Use“ ersichtlich ist (Abb. 4-9).[95]

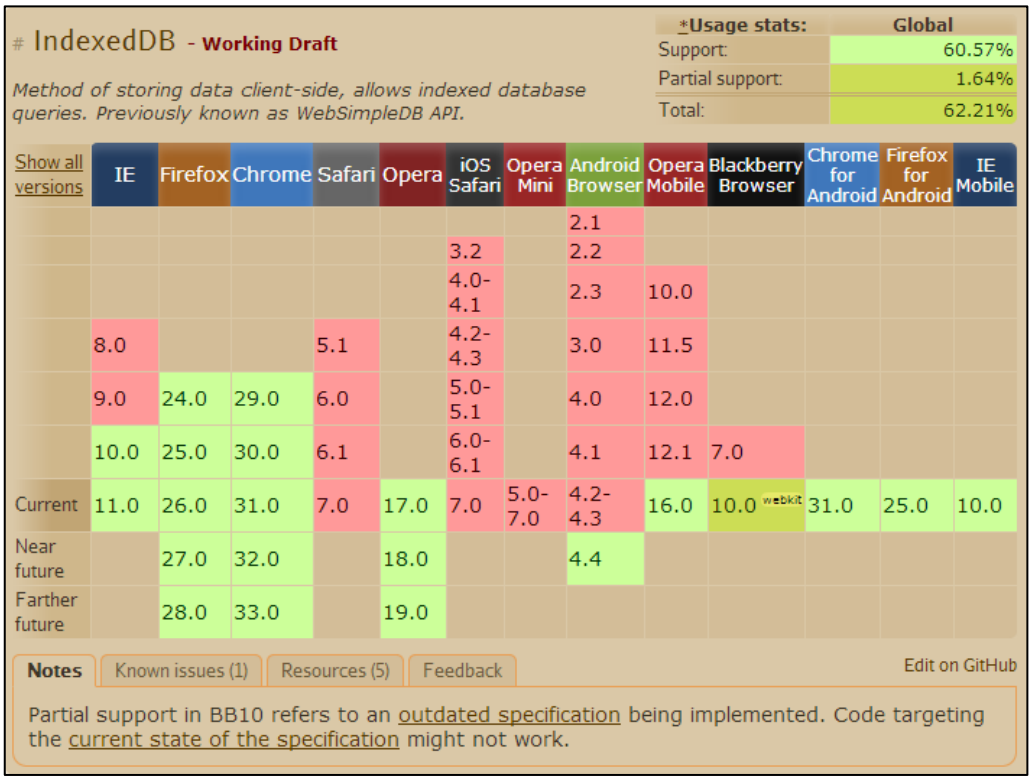

Abb. 4-9 Browserunterstützung - IndexedDB

Die IndexedDB ist für Anwendungen geeignet, die Daten, die über Key/Value-Paare hinausgehen, speichern und effizient wieder auffinden müssen. Es ist möglich, mehrere Objekte mit demselben Index abzuspeichern. Auch wenn die Reihenfolge der Daten in der Anwendung eine Rolle spielt, bietet sich die

[94] *Vgl. W3C, Web SQL Database, 26.01.2014*

[95] *Vgl. Deveria, Can I Use – IndexedDB, 26.01.2014*

IndexedDB an.[96] Ebenso kann die Datenbank verwendet werden, um Webanwendungen und ihre Daten offline verfügbar zu machen. Dazu wird zusätzlich die Offline-API benötigt, die später noch erläutert wird.

Die IndexedDB ist eine Datenbank auf Client-Seite, auf die nur von der Webseite oder –anwendung zugegriffen werden kann, von der sie angelegt wurde. Da die Datenbank lokal gespeichert wird und nur die entsprechende Anwendung, identifiziert in Abhängigkeit von der URL, Zugriff darauf hat, ist bei der IndexedDB keine Authorisierung oder Identifizierung per Benutzerkennung und Passwort notwendig. Die Daten werden in Objekten in einem Objektspeicher (englisch: object store) abgelegt. Für einen Objektspeicher muss mindestens ein Index definiert werden, der in jedem zu speichernden Objekt vorhanden sein muss. Der Wert des Index kann automatisch generiert werden, wodurch ein inkrementeller Index entsteht. Über die Indizes kann effizient nach Daten gesucht und auf sie zugegriffen werden. Die Objekte selbst können eine unbestimmte Anzahl beliebiger Schlüssel/Wert-Paare enthalten, sodass sich einzelne Objekte in einem Objektspeicher in der Anzahl der Schlüssel/Wert-Paare und in den Namen der Keys unterscheiden können. Zugriffe auf die Datenbank müssen in Transaktionen eingebettet werden. Eine Größenbegrenzung wird vom W3C nicht vorgesehen.

Um von einer Webanwendung aus auf eine Datenbank zuzugreifen, benötigt man lediglich die Methode „indexedDB.open(„dbName“)“. Dieser Befehl öffnet eine Datenbank mit dem angegebenen Namen, falls sie für die URL der Anwendung vorhanden ist. Falls keine Datenbank existiert, wird sie automatisch angelegt. Die Datenbank hat immer eine Versionsnummer. Diese kann man mit der Methode „setVersion(„versNumber“)“ vergeben. Default nach dem Anlegen einer Datenbank ist der Wert 0. Beim erstmaligen Anlegen der Datenbank oder beim Versuch per Methode „open(„dbName“, versNumber)“ auf eine bestimmte Version der Datenbank zuzugreifen, die die Datenbank selbst noch nicht besitzt, wird ein Event „upgradeneeded“ geworfen, auf das man dementsprechend reagieren kann. Ist alles in Ordnung, so erhält man das Event „success“ und kann aus dem Event „e“, je nach

[96] *Vgl. W3C, Indexed Database API, 26.01.2014*

Implementierung im Browser, per „e.result“ oder „e.target.result“ den Verweis auf die Datenbank erhalten.

Möchte man nun einen Objektspeicher anlegen, der in etwa einer Tabelle bei relationalen Datenbanken entspricht, jedoch keine festen Attribute besitzt, so kann man dies nur während eines „updateneeded“-Events. Dies bewirkt, dass bei jeder Strukturänderung der Datenbank die Versionsnummer angepasst werden muss. Auf die zuvor angelegte Datenbank kann nun die Methode „createObjectStore(„objStoreName“, {<indexDefinition>})“ zum Anlegen eines Objektspeichers aufgerufen werden. Den Index kann man in dem zweiten Parameter, der optional ist, festlegen. Er kann in der Methode „CreateObjectStore( )“ auf zwei Arten vergeben werden. Einen automatisch vergebenen, inkrementellen Index erhält man, indem man <indexDefinition> gleich „autoIncrement: true“ setzt. Setzt man jedoch „keypath: <indexKey>“, so wird der Index durch eine Eigenschaft der Objekte, in relationalen Datenbanken Attribut genannt, definiert. Die Eigenschaft <indexKey> muss dann in allen eingefügten Objekten enthalten sein.

Um nun Daten im Object Store zu speichern, muss man, wie bei jedem Zugriff oder jeder Änderung der Daten, eine Transaktion eröffnen.[97] Auf die Datenbank wird hierzu die Methode „transaction([„objStoreName], „readwrite“)“ unter Angabe des Namens des Objektspeichers und des Transaktionstyps ausgeführt. Mit dem Verweis auf den zurückgelieferten Object Store können nun Objekte, die inline oder vorangehend definiert werden, in den Objektspeicher per „objStoreName.add(„objName“)“ gespeichert werden. Um festzustellen, ob die Daten hinzugefügt wurden oder ob ein Fehler aufgetreten ist, kann wiederum ein Event abgefragt werden. Wurde beim Anlegen des Objektspeichers kein Index-Verfahren angegeben, so muss man hier der „add( )“-Methode als zweiten Parameter einen selbst generierten Index mitgeben. Zum Ändern oder Ersetzen von Daten gibt es analog eine „put(„objName“)“-Methode, Lesen kann man per „get(„key“)“ und Löschen per „delete(„key“)“. Um die Transaktion abzuschließen, fragt man das geworfene Event ab und fügt für den festgestellten Status den gewünschten Code ein.

---

[97] *Camden, net tuts+ - Working With IndexedDB, 01.01.2013*

Die IndexedDB-API bietet einen Cursor an, über den eine Gruppe von Objekten angesprochen werden kann. Alle Objekte eines Objektspeichers erhält man per „openCursor( )“. Das aus der Methode resultierende Objekt enthält einen Zeiger auf das erste Objekt und man kann unter Angabe des enthaltenen Schlüssels auf den dazugehörigen Wert zugreifen. Führt man auf dem Cursor „continue( )“ aus, so gelangt man zum nächsten Objekt. Beim Anlegen des Cursors kann noch festgelegt werden, ob die Daten auf- oder absteigend durchlaufen werden sollen. Auch kann man festlegen, ob doppelte Objekte, abhängig vom Index, zurückgegeben werden sollen oder nicht.

Um eine bestimmte Menge von Objekten zu lesen, bietet die API die Möglichkeit, der „openCursor( )“-Methode Bedingungen als Parameter mitzugeben. Diese werden in Form von „IDBKeyRange“-Objekten erzeugt. So kann man, wenn man zuvor den Index eines Objektspeichers auf einen bestimmten Schlüssel gesetzt hat, alle Objekte erhalten, die darin einen bestimmten Wert enthalten. Eine solche Bedingung legt man per „IDBKeyRange.only(<value>)“ an und gibt den Returnwert (range) dem „openCursor(range)“-Befehl mit. Auf ähnliche Weise kann mit der Methode „IDBKeyRange.bound( )“ ein Wertebereich generiert werden.

Die genaue Spezifikation und weiterführende Information zur Indexed-Database-API können auf den Seiten des W3C aufgerufen werden (http://www.w3.org/TR/IndexedDB/, 26.01.2014). Eine Anleitung von Raymond Camden, wie mit der IndexedDB zu arbeiten ist, findet man bei „net tuts+“ (http://net.tutsplus.com/tutorials/javascript-ajax/working-with-indexeddb/, 26.01.2014).

Den Inhalt der IndexedDB kann man in den Entwicklerwerkzeugen des Chrome im Reiter „Resources“ unter „IndexedDB“ ansehen. Dazu muss natürlich die Anwendung im Browser aufgerufen werden, da nur dies den Zugriff des Browsers auf die Daten erlaubt.

4.2.6 Offline-API

Gestaltet man eine Anwendung webbasiert, so besteht auch heutzutage die Gefahr, dass keine Internetverbindung verfügbar ist, wenn man die Anwendung nutzen möchte. Stationäre Verbindungen können aufgrund von Leitungs- oder Serverproblemen ausfallen, bei mobilem Netzzugang können zusätzlich schlechte Netzabdeckung oder fehlender Empfang aufgrund der

räumlichen Situation auftreten. Solche Probleme können mit der Offline-API für Anwendungen und Webseiten, die rein clientseitig ausgeführt werden, minimiert werden. Mit der API können bei bestehender Verbindung alle Daten, die für eine Anwendung erforderlich sind, heruntergeladen und so im Offline-Modus verfügbar gemacht werden. Ebenfalls kann man mit Hilfe dieses API die Ladezeiten effizienter gestalten, indem Daten dauerhaft auf dem System des Anwenders gespeichert werden. Bei Anwendungen oder Webseiten mit hohem Datenvolumen fällt die Downloadzeit damit nur beim ersten Aufruf an, erneute Zugriffe sind beschleunigt.

Um dem Browser mitzuteilen, welche Dateien offline gespeichert werden sollen, muss eine Manifestdatei mit der Dateierweiterung „.appcache“ zur Anwendung angelegt werden. Alle darin zu dem Eintrag „CACHE“ namentlich aufgeführten Dateien werden offline gespeichert. Im Eintrag „NETWORK“ kann man Dateien auflisten, die immer vom Server geladen werden sollen. Wurde die Anwendung aktualisiert, ist darauf zu achten, dass die Manifest-Datei angepasst wird. Ein neuer Kommentar, zum Beispiel mit dem Änderungsdatum, reicht aus, damit der Browser Anpassungen feststellt und alle Dateien neu herunterlädt.

Im HTML muss die Manifestdatei im <html>-Element unter dem Attribut „manifest“ angegeben werden, damit sie berücksichtigt wird.

Um zu prüfen, ob das Herunterladen komplett durchgeführt werden konnte, gibt es ein Event, worauf das Objekt „ApplicationCache“ horchen kann. Im Fehlerfall kann zum Beispiel ein Hinweis ausgegeben werden. Auch kann der Status des Cache-Prozesses abgefragt werden. Das „progress“-Event ermöglicht eine Fortschrittsanzeige zu implementieren, wodurch dem Anwender angezeigt werden kann, wie der Verlauf ist.

4.2.7 File-API

Vor HTML5 war kaum eine Dateiverarbeitung im Browser des Anwenders möglich. Es konnten lediglich Dateien von einem Server heruntergeladen und auf dem aufrufenden Rechner gespeichert werden. Die zweite Art der Dateiverarbeitung war das Hochladen einer Datei auf einen Server. HTML5 ermöglicht das Lesen, Erstellen und Verarbeiten von Dateien und bietet hierzu gleich drei APIs an. „File-API“, „File-API: Directories and System“ und „File-API: Writer“.

Die „File-API“ an sich kann dazu genutzt werden, um vom Anwender ausgewählte Dateien einzulesen. Wie zuvor schon erwähnt, kann eine Auswahl für Dateien per „input“-Element des Typs „file“ oder mit einer Drag&Drop-Aktion realisiert werden. Eine so vom Anwender spezifizierte Datei kann mit Hilfe des „FileReader“-Objektes als Text, als binärer String bei Bildern und Videos oder als „data:url“ eingelesen werden. Die so gewonnenen Daten können auf der Webseite oder -anwendung angezeigt werden. In dieser API existiert auch ein „file“-Objekt, mit dem der Name, die Größe und der Typ der Datei abgefragt werden kann. Events ermöglichen es, den Status des Ladens einer Datei in den Hauptspeicher zu prüfen. Beim Start des Einlesens wird das Event „loadstart“, während des Lesens regelmäßtig „progress“ und nach erfolgreichem Laden „load“ geworfen. Zusätzlich gibt es auch eine Schnittstelle, um „Binary Large Objects“ (BLOBs) behandeln zu können.

Die „File API: Directories and System“ ermöglicht es, Dateien oder Verzeichnisse anzulegen. Hierzu wird Speicherplatz auf der Festplatte des Anwenders reserviert. Auf diesen Bereich des Dateisystems kann nur die Anwendung, die den Speicherplatz angelegt hat, zugreifen. Zum Anlegen gibt es die Funktion „requestFileSystem( )“, der man den Typ „TEMPORARY“ oder „PERSISTENT“ und den benötigten Speicherplatz mitgeben kann. Es wird das Basisverzeichnis als „root“-Objekt zurückgegeben. Die Methoden „getDirectory( )“ und „getFile( )“ können darauf ausgeführt werden, um ein Verzeichnis beziehungsweise eine Datei zu erstellen. Auch Methoden zum Kopieren, Verschieben und Löschen von Dateien und Verzeichnissen sind vorhanden.

Um Daten in selbst angelegte Dateien schreiben zu können, wird die dritte API, die „File-API: Writer“ benötigt. Dazu erstellt man zunächst ein „FileWriter“-Objekt mit der Methode „createWriter( )“. Auf das „FileWriter“-Objekt können wiederum Methoden zum Schreiben eines Dateiinhaltes, auch ab einem bestimmten Aufsatzpunkt, zum Abfragen der Länge, zum Ändern der Dateilänge und vieles mehr ausgeführt werden.

Leider ist die Implementierung der File-APIs in den Browsern noch nicht weit fortgeschritten. „File-API: Directories and System“ und „File-API: Writer“ sind bisher lediglich in Chrome, Opera und dem Blackberry Browser umgesetzt.[98]

4.2.8 Web-Workers-API

JavaScript ist prinzipiell eine simple Sprache und läuft in einem Thread ab. Die Web-Workers-API ermöglicht es, JavaScript im Hintergrund in einem separaten Prozess laufen zu lassen, sodass es die restliche Webseite nicht beeinflusst. Komplexere Anwendungen und Webseiten können somit effizienter gestaltet werden. JavaScript wird mit Hilfe der Web-Workers-API multithread-fähig.

Zu einem HTML-Dokument und der Haupt-JavaScript-Datei muss der JavaScript-Code, der im Hintergrund ablaufen soll, in eine separate JavaScript-Datei geschrieben werden. Um auf diesen Code zugreifen zu können, wird im Hauptskript zum Beispiel bei der Initialisierung ein Worker-Objekt unter Angabe des Dateinamens generiert (new Worker(<filename>)). Damit die beiden Skripte untereinander kommunizieren können, horcht das Worker-Objekt auf das Event „message“. Aus dem Hauptskript kann man auf das Worker-Objekt die Methode „postMessage(<message>)“ aufrufen, da dieses ebenfalls auf Nachrichten horcht. Die Rückgabe einer Antwort ist aus dem Worker ebenfalls per „postMessage(<answre>)“ möglich. Diese Nachricht wird als Event durch das Hauptskript empfangen. Die verwendeten Nachrichten müssen aus Strings oder „JavaScript Object Notation“-Objekten (JSON-Objekten) bestehen. JSON ist eine Syntax, um Daten zu übertragen und zu speichern, ähnlich der „Extensible Markup Language“ (XML). Da die Syntax identisch der von JavaScript-Objekten ist, lassen sich diese einfach in JSON-Objekte umwandeln und umgekehrt.

Der Worker hat jedoch keinen Zugriff auf Elemente des HTML oder das Haupt-JavaScript. Weitere Funktionalitäten zum Abfangen von Fehlern, zum Abbrechen eines Workers, zum Importieren externer Skripte in die Worker und viele mehr stehen darüber hinaus zur Verfügung.

---

[98] *Deveria, Can I use - Filesystem and FileWriter API, 26.01.2014*

4.2.9 Sonstige APIs

Weitere APIs wurden für eine bessere Kommunikation in den HTML5-Standard aufgenommen.

Die „XML-HttpRequest-API Level 2“ realisiert mehr Möglichkeiten für AJAX. AJAX ermöglicht es, Anfragen an den Server zu stellen und Daten auf einer Seite zu verändern, ohne die gesamte Seite neu laden zu müssen. Der Level 2 dieser API besitzt mehr Events, mit denen der Übertragungsstatus überprüft werden kann und die Möglichkeit, Dateien mit AJAX hochzuladen oder Formulardaten zu übertragen.

Die „Cross-Document-Messaging-API“ ermöglicht es Anwendungen über die Grenzen eines Fensters, eines Tabs oder eines Frames hinweg zu kommunizieren, indem Nachrichten von einem Dokument in ein anderes geschickt werden. Hierbei muss das empfangende Dokument auf ein „message“-Event horchen.

Mit Hilfe der „History-API“ ist es möglich, aus JavaScript heraus in der Browser-Chronik vor und zurück zu springen. Diese Funktionalität kennt man als Anwender von den Pfeiltasten her. Doch auch aus Anwendungen heraus kann diese Funktion sinnvoll sein. Da bei der Verwendung von AJAX zum Nachladen von Daten sich die URL nicht ändert und somit auch nicht in der Browser-Historie vorhanden ist, gibt es nun die Möglichkeit über diese API künstliche Einträge in der Historie zu generieren.

Die „Geolocation-API“ kann verwendet werden, wenn der Standort des Anwenders ermittelt werden soll. Dies ist für Applikationen nützlich, deren Informationen sich auf den Standort beziehen. Dies können zum Beispiel Auskunftssysteme für öffentliche Verkehrsmittel oder Gastronomiesuchen sein. Aber auch bei Anwendungen für Außendienstmitarbeiter könnte die Standortermittlung eingesetzt werden, um Informationen zu Kunden anzuzeigen, die in der Nähe wohnen.

Fazit

Im Rahmen dieser Arbeit wurden Verfahren und Techniken zusammengetragen, die eine effiziente Implementierung von responsiven Webapplikationen im Rahmen von HTML5 ermöglichen.

Es existiert eine Vielzahl von Büchern und Internetartikeln zu responsivem Webdesign. In ihnen wird beschrieben, wie Webseiten responsiv gestaltet werden können. Viele der dort aufgeführten Punkte sind ebenfalls für das Design von Webanwendungen relevant und wurden im Rahmen dieser Arbeit erläutert.

Da die Performance einer Webanwendung besonders wichtig ist, wenn diese auch auf mobilen Geräten genutzt werden soll, wurde der Performance-Optimierung besondere Beachtung geschenkt. Bisher existierende Texte zur Performance beziehen sich wie beim responsiven Design meist auf Webseiten. Viele beschriebene Optimierungsmöglichkeiten lassen sich auf Webanwendungen übertragen und wurden für diese Arbeit berücksichtigt. Da prinzipiell für Anwendungen mehr Funktionalitäten implementiert werden müssen als für Webseiten, sind Optimierungen bezüglich des JavaScripts von größerer Bedeutung als bei informativen Webseiten. Gegenläufig verhält es sich bei Bildern. Diese werden bei Webanwendungen generell eher weniger benötigt. Deshalb wurden Optimierungsmöglichkeiten für einzelne Bilder in dieser Arbeit nicht betrachtet. Lediglich die Zusammenfassung von Bildern oder Grafiken in einer Datei wurde beschrieben, da dies beispielsweise für Symbole innerhalb einer Anwendung relevant ist.

Die HTML5-Spezifikation mit den darin enthaltenen JavaScript APIs soll laut W3C die Erstellung von clientseitigen Webapplikationen ermöglichen. Diese APIs, die eine Implementierung von Funktionalitäten wie man sie von lokal installierten Anwendungen kennt, ermöglichen, wurden in dieser Arbeit ebenfalls erläutert.

Die in dieser Arbeit zusammengetragenen Verfahren und Techniken zeigen, dass eine Implementierung einer responsiven Webapplikation mit dem heute zur Verfügung stehenden HTML5 zu realisieren ist.

Ein reaktionsfähiges Layout kann mit Hilfe von flexiblen Rastern, Mediaqueries zum Heranziehen von unterschiedlichen Styles und flexibel reagierenden Inhalten umgesetzt werden. Durch ein auf die Erstellung von responsiven Webanwendungen angepasstes Vorgehensmodell lässt sich die Effizienz des Entwicklungsprozesses steigern. Dies ist notwendig, da die Entwicklung einer responsiven Webapplikation mehr Aufwand verursacht als die Erstellung einer Webanwendung für eine feste Bildschirmgröße. Auch wenn ein responsives Design nicht für bestimmte aktuelle Anzeigeflächen entworfen werden soll, so hat es jedoch Vorteile für die Effizienz des Umsetzungsprozesses, wenn die Anzeige für ein paar wenige Formate optimiert ist. Möchte man für alle zurzeit oder in der Zukunft existierenden Bildschirmgrößen ein optimales Layout realisieren, so würde hoher Aufwand für die Erstellung von unterschiedlichen Styles anfallen.

Für eine Performance-Optimierung von responsiven Webapplikationen gibt es eine Vielzahl an Möglichkeiten. Eine möglichst geringe Anzahl an HTTP-Requests bis zur ersten Anzeige der Webanwendung ist sicherlich die wichtigste Option. Erreichen lässt sich dies, wie beschrieben, durch Zusammenfassen von CSS-, JavaScript- und Bilddateien und auch durch optimiertes Caching. Die Minimierung der Dateigröße sollte bei einer Umsetzung ebenfalls beachtet werden. Wie die Liste der weiteren Ansatzpunkte im Kapitel Performance-Optimierung zeigt, gibt es unzählige weitere Möglichkeiten, die Lade- und Reaktionszeiten von Webanwendungen zu reduzieren. Auch die Möglichkeit JavaScript in mehreren Threads ablaufen zu lassen kann zur Steigerung der Performance genutzt werden. Somit lassen sich für jede Art der Webapplikationen Optimierungsmöglichkeiten umsetzen. Es sollte jedoch auf das Beschleunigungspotenzial durch die einzelnen Optimierungen im Verhältnis zum hierfür zu investierenden Entwicklungsaufwand geachtet werden. Ineffizient würde die Umsetzung der Anwendung, wenn aufwandsintensive Optimierungen umgesetzt würden, die lediglich eine minimale Beschleunigung der Anwendung bewirken. Zur Ermittlung von Optionen für die Performance-Optimierung empfiehlt es sich, die genannten Testtools zu verwenden.

Mit den in HTML5 spezifizierten JavaScript APIs ist ein breites Spektrum an Funktionen möglich, wie sie für im Browser laufende Webanwendungen benötigt werden. Theoretisch steht der Realisierung einer Webapplikation nichts mehr im Wege. Leider ist die Umsetzung dieser APIs in den verschiedenen Browsern noch nicht so weit fortgeschritten, dass man von einer flächendeckenden Verfügbarkeit beim Anwender ausgehen kann. Das hat zur Folge, dass Webanwendungen, wenn sie jedem Anwender zur Verfügung stehen sollen, nur sehr begrenzte Funktionen der APIs nutzen können. Ist für eine Webanwendung eine Funktion unbedingt notwendig, diese jedoch noch nicht in allen gängigen Browsern umgesetzt, so bleibt den Entwicklern nichts anderes übrig als beim Aufruf die Browserkompatibilität zu prüfen und nötigenfalls auf die Verwendung eines anderen Browsers hinzuweisen. Auch stößt das Prinzip „Progressive Enhancement“ bei Webapplikationen an seine Grenzen, denn der Grundsatz, dass für alle Anwender der gesamte Inhalt dargestellt werden soll, lässt sich für Webapplikationen zum Beispiel bei deaktiviertem JavaScript nicht realisieren. Funktionen, wie sie für Applikationen benötigt werden, sind auf JavaScript angewiesen.

Somit zeigt diese Arbeit, dass prinzipiell ausreichend Techniken für die Erstellung von performanten, responsiven Webapplikationen zur Verfügung stehen. Die Problematik der Browserkompatibilität stellt sich bei jeder Technik, die neu im World Wide Web eingeführt wird. Die Probleme hierdurch werden durch die Etablierung der neuen Techniken geringer.

Eine Optimierung der Effizienz des Entwicklungsprozesses ist ebenfalls möglich. Es muss bei der Umsetzung jedoch darauf geachtet werden, dass nicht unnötig hoher Aufwand für die Optimierung von Layout und Performance investiert wird. Sicherlich ist die Umsetzung einer responsiven Anwendung zeitintensiver als die Realisierung einer Anwendung für eine feststehende Bildschirmgröße. Eine Implementierung von zwei oder mehr separaten Webanwendungen, zum Beispiel für Desktop-PCs und Smartphones, verursacht jedoch noch höhere Aufwände sowohl bei der Entwicklung als auch bei der Wartung.

Ausblick

Im Rahmen dieser Arbeit wurde die theoretische Umsetzung von Webanwendungen in reinem HTML5 betrachtet. Die mit HTML5 realisierten Webanwendungen werden clientseitig im Browser ausgeführt.

Eine Implementierung einer responsiven Webanwendung mit den hier beschriebenen Techniken würde auch in der Praxis zeigen, dass eine Umsetzung möglich ist. Durch die Umsetzung ließe sich der Aufwand, der für eine Optimierung der Performance anfällt, besser beurteilen. Ebenso wäre besser einschätzbar, wie hoch der Aufwand für die Umsetzung eines Layouts wäre, das auf Anzeigeflächen unterschiedlichsten Größe akzeptabel ist. Auch ließe sich besser beurteilen, wie die Kompatibilität der Browser bezüglich der HTML5 JavaScript APIs ist.

Um die Effizienz der Implementierung von responsiven Webapplikationen umfassender beurteilen zu können, könnte analysiert werden, ob die Verwendung von CSS-Präprozessoren wie SASS, LESS oder Stylus den Arbeitsaufwand reduzieren. Der Einsatz von CSS-Frameworks wie Twitter Bootstrap, Zurbs Foundation oder YAML könnte ebenso Entwicklungszeit einsparen wie auch JavaScript-Frameworks oder –Bibliotheken.

Auf Seiten des Servers könnten mit Hilfe von Skripten und Backend-Anwendungen weitere Optimierungen für eine bessere Reaktionsfähigkeit der Anwendung umgesetzt werden. Serverseitig lässt sich zum Beispiel implementieren, dass nur die Daten übertragen werden, die auf dem abrufenden System benötigt werden. Auch könnte die Umsetzung einer Speicherung der Daten auf einem zentralen Server betrachtet werden. Vorstellbar wäre zum Beispiel die temporäre Speicherung von im Browser erfassten Daten in Form von JSON-Objekten in einer IndexedDB und der zeitlich versetzten Übertragung der Daten in einem nachrangigen JavaScript-Thread an den Server.

Ein anderes interessantes Thema im Zusammenhang mit Webseiten oder –anwendungen ist deren ökologischer Fußabdruck. Die Nachhaltigkeit wird zurzeit für viele Dinge betrachtet. Serverbetreiber werben mit ihren Systemen, die zumindest zur Laufzeit wenig Energie benötigen. Einzelne Webseiten oder –Anwendungen werden diesbezüglich noch nicht analysiert.

Literaturverzeichnis

Allsopp, John. *A List Apart - A Dao of Web Design*. 07. 04. 2000. (20. 01. 2014). <http://alistapart.com/article/dao>.

Baker, Ryan. *Wichita State University - Is Multiple-Column Online Text Better? It Depends!* 07. 2005. (23. 01. 2014). <http://psychology.wichita.edu/surl/usabilitynews/72/columns.asp>.

Bundesverband des Deutschen Versandhandels e.V. *… Mobiler Einkauf legt 2013 noch einmal deutlich zu ...* 28. 05. 2013. (20. 01. 2014). <http://www.bvh.info/presse/pressemitteilungen/details/datum/2013/mai/artikel/aktuelle-studienergebnisse-mobiler-einkauf-legt-2013-noch-einmal-deutlich-zu-40-prozent-aller-sma/?cHash=b1be135a1948bcca55be758002b3ccbc>.

Bushell, David. *Smashing Magazine - Implementing Off-Canvas Navigation for a Responsive Website*. 15. 01. 2013. (26. 01. 2014). <http://coding.smashingmagazine.com/2013/01/15/off-canvas-navigation-for-responsive-website/>.

Bustos, Linda. *GetElastic - Don't Make These Mobile Menu Mistakes*. 10. 04. 2013. (23. 01. 2014). <http://www.getelastic.com/dont-make-these-mobile-menu-mistakes/>.

Camden, Raymond. *net tuts+ - Working With IndexedDB*. 13. 09. 2013. (26. 01. 2014). <http://net.tutsplus.com/tutorials/javascript-ajax/working-with-indexeddb/>.

Clark, Richard. *html5 Doctor - The main element*. 26. 06. 2013. (23. 01. 2014). <http://html5doctor.com/the-main-element/>.

Clarke, Andy. *Stuff & Nonsense - Walls Come Tumbling Down*. 28. 06. 2009. (20. 01. 2014). <http://www.stuffandnonsense.co.uk/blog/about/walls_come_tumbling_down_presentation_slides_and_transcript/>.

—. *Stuff & Nonsense - We need a standard show navigation icon for responsive web design*. 17. 03. 2012. (23. 01. 2014).

<http://stuffandnonsense.co.uk/blog/about/we_need_a_standard_show_navigation_icon_for_responsive_web_design>.

Deveria, Alexis. *Can I use - CSS3 Media Queries?* o. Datum. (23. 01. 2014). <http://caniuse.com/css-mediaqueries>.

—. *Can I use - Data URIs?* o. Datum. (23. 01. 2014). <http://caniuse.com/datauri>.

—. *Can I use - Filesystem and FileWriter API*. 15. 10. 2013. (26. 01. 2014). <http://caniuse.com/#feat=filesystem>.

—. *Can I Use - IndexedDB*. 15. 10. 2013. (26. 01. 2014). <http://caniuse.com/#feat=indexeddb>.

Frost, Brad. *brad frost web - Responsive Navigation Patterns*. 24. 02. 2012. (20. 01. 2014). <http://bradfrostweb.com/blog/web/responsive-nav-patterns/>.

Gauchat, Juan Diego. *HTML5, CSS3 & JavaScript*. 1. Auflage. Weinheim: Wiley-VCH Verlag, 2013.

Gläßner, Matthias. *Seitenreport - Grundlagen der Website Performance - 1. Requests*. 03. 04. 2013. (26. 01. 2014). <http://www.seitenreport.de/kb/-/grundlagen-der-website-performance-1-requests.html>.

Hahlbohm, Kerstin. *mobile zeitgeist - Responsive Webdesign Workflow – Sechs Praxistipps*. 24. 06. 2013. (20. 01. 2014). <http://www.mobile-zeitgeist.com/2013/06/24/responsive-webdesign-workflow-sechs-praxistipps/>.

Johansson, Johan. *Smashing Magazine - How To Make Your Websites Faster On Mobile Devices*. 03. 04. 2013. (26. 01. 2014). <http://mobile.smashingmagazine.com/2013/04/03/build-fast-loading-mobile-website/>.

Jonathan. *ZURBlog - Small, Painful Buttons: Why Social Media Buttons Might be Killing Your Mobile Web Site*. 17. 01. 2012. (26. 01. 2014). <http://zurb.com/article/883/small-painful-buttons-why-social-media-bu>.

Marcotte, Ethan. *A List Apart - Fluid Grids*. 03. 03. 2009. (20. 01. 2014). <http://alistapart.com/article/fluidgrids>.

—. *A List Apart - Responsive Web Design*. 25. 05. 2010. (20. 01. 2014). <http://alistapart.com/article/responsive-web-design>.

Müller, Peter. *Flexible Boxes - Eine Einführung in moderne Websites*. 1. Auflage. Bonn: Galileo Press, 2013.

Müller, Sergej. *Playground - Content Delivery Network (CDN) von Google für Eigenprojekte nutzen*. 31. 01. 2013. (26. 01. 2014). <http://playground.ebiene.de/google-app-engine-als-cdn/>.

Preuss, Sebastian. *//SEIBERT/MEDIA-Weblog - Wann sind Akkordeon-Effekte auf Websites sinnvoll?* 31. 03. 2010. (20. 01. 2014). <http://blog.seibert-media.net/2010/03/31/wann-sind-akkordeon-effekte-auf-websites-sinnvoll/>.

Schmidt, Jürgen. *c't - 2 Klicks für mehr Datenschutz*. 01. 09. 2011. (26. 01. 2014). <http://www.heise.de/ct/artikel/2-Klicks-fuer-mehr-Datenschutz-1333879.html>.

Siegel, Ronny. *My Webcheck - Weitenbreite von Websites 2013*. 07. 06. 2013. (20. 01. 2014). <http://www.mywebcheck.de/seitenbreite-websites-2013>.

Smith, Nathan. *960 Grid System*. o. Datum. (20. 01. 2014). <http://960.gs>.

Souders, Steve. *stevesouders.com - Revving Filenames: don't use querystring*. 23. 08. 2008. (23. 01. 2014). <http://www.stevesouders.com/blog/2008/08/23/revving-filenames-dont-use-querystring/>.

—. *Yahoo! Developer Network - High Performance Web Sites: Rule 3 – Add an Expires Header*. 24. 05. 2007. (26. 01. 2014). <http://developer.yahoo.com/blogs/ydn/high-performance-sites-rule-3-add-expires-header-7181.html>.

—. *Yahoo! Developer Network - High Performance Web Sites: Rule 6 - Move Scripts to the Bottom*. 12. 07. 2007. (26. 01. 2014). <http://developer.yahoo.com/blogs/ydn/high-performance-sites-rule-6-move-scripts-bottom-7200.html>.

StatCounter. *StatCounter Global Stats*. 12. 2013. (23. 01. 2014). <http://gs.statcounter.com/>.

Thelwell, Chris. *Design-in-browser, are you ready?* 26. 06. 2013. (14. 01. 2014). <https://medium.com/think-big-work-smart/afb0ed7cab4a>.

Tsonev, Krasimir. *How to Design Responsively*. 10. 06. 2013. (20. 01. 2014). <http://davidwalsh.name/design-responsively>.

W3C, World Wide Web Consortium. *HTML 5.1*. 29. 10. 2013. (23. 01. 2014). <http://www.w3.org/TR/html51/>.

—. *Hypertext Transfer Protocol -- HTTP/1.1*. 06. 1999. (26. 01. 2014). <http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8>.

—. *Indexed Database API*. 04. 07. 2013. (26. 01. 2014). <http://www.w3.org/TR/IndexedDB/>.

—. *Web SQL Database*. 18. 11. 2010. (26. 01. 2014). <http://dev.w3.org/html5/webdatabase/>.

W3Schools. *w3schools.com - CSS Image Sprites*. o. Datum. (26. 01. 2014). <http://www.w3schools.com/css/css_image_sprites.asp>.

—. *w3schools.com - HTML5 Form Attributes*. o. Datum. (26. 01. 2014). <http://www.w3schools.com/html/html5_form_attributes.asp>.

Walton, Trent. *Fluid Type*. 19. 06. 2012. (23. 01. 2014). <http://trentwalton.com/2012/06/19/fluid-type/>.

Warren, Samantha. *A List Apart - Style Tiles and How They Work*. 27. 03. 2012. (20. 01. 2014). <http://alistapart.com/article/style-tiles-and-how-they-work>.

—. *Style Tiles*. o. Datum. (20. 01. 2014). <http://styletil.es>.

Website Optimization. *Average Web Page Size Triples Since 2008*. 11. 2012. (23. 01. 2014). <http://www.websiteoptimization.com/speed/tweak/average-web-page/>.

WHATWG. *HTML - Living Standard - The canvas element*. 25. 01. 2015. (26. 01. 2014). <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#the-canvas-element>.

—. *Web Applications 1.0*. 01. 09. 2005. (26. 01. 2014). <http://www.whatwg.org/specs/web-apps/2005-09-01/>.

Wikipedia. *Apple iPhone*. 17. 01. 2014. (20. 01. 2014). <http://de.wikipedia.org/wiki/Apple_iPhone>.

—. *Design*. 14. 01 2014. (20. 01. 2014). <http://de.wikipedia.org/wiki/Design>.

WooThemes. *FlexSlider 2*. o. Datum. (20. 01. 2014). <http://flexslider.woothemes.com/>.

Wroblewski, Luke. *Mobile First*. 1. Auflage. New York: A Book Apart, 2011.

—. *Mobile First*. 03. 11. 2009. (20. 01. 2014). <http://www.lukew.com/ff/entry.asp?933>.

—. *Off Canvas Multi-Device Layouts*. 21. 06. 2012. (20. 01. 2014). <http://www.lukew.com/ff/entry.asp?1569>.

Yahoo! *Developer Network - Best Practices for Speeding Up Your Web Site*. o. Datum. (26. 01. 2014). <http://developer.yahoo.com/performance/rules.html>.

Zakas, Nicholas C. *A List Apart - Better JavaScript Minification*. 20. 04. 2010. (23. 01. 2014). <http://alistapart.com/article/better-javascript-minification>.

—. *A List Apart - JavaScript Minification Part II*. 20. 07. 2010. (23. 01. 2014). <http://alistapart.com/article/javascript-minification-part-II>.

Zeldman, Jeffrey. *Twitter - Zeldman*. 05. 05. 2008. (20. 01. 2014). <https://twitter.com/zeldman/statuses/804159148>.

Zillgens, Christoph. *Responsive Webdesign - Reaktionsfähige Websites gestalten und umsetzen*. 1. Auflage. München: Carl Hanser Verlag, 2013.

Anhang

Die im Literaturverzeichnis aufgeführten Artikel und Informationen aus dem Internet befinden sich in Form von PDF-Dokumenten auf der beiliegenden CD. Der Dateiname der PDFs setzt sich aus dem Nachnamen des Verfassers und dem Titel des Artikels zusammen, sodass die einzelnen Texte leicht auffindbar sind.

Ebenso ist diese Arbeit als Word- und PDF-Dokument auf der CD enthalten.

Erklärung über die selbstständige Abfassung der Arbeit

„Ich versichere an Eides statt, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt beziehungsweise in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.“

Bonn, 01.02.2014